INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Aplicação móvel para aconselhamento de treino de jovens atletas

DAVID DUARTE PINTO DE ABREU
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Aplicação móvel para aconselhamento de treino de jovens atletas

David Duarte Pinto de Abreu

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

Orientador: António Constantino Martins

Coorientador: Paulo Matos

Júri:

Presidente:

Vogais:

Porto, outubro de 2019

Aos meus pais por tudo o que me proporcionaram ao longo da vida e aos meus familiares e amigos que sempre me apoiaram ao longo do meu percurso profissional

Resumo

Numa sociedade cada vez mais repleta de informação, torna-se necessário facilitar o acesso a dados, desse modo, surgem os sistemas de recomendação. Atualmente, o uso destas ferramentas tem uma maior adesão nas plataformas de comercio online (*e-commerce*), no entanto, estas ferramentas podem ser aplicadas em diversas áreas. Assim sendo, é pretendido testar a usabilidade e utilidade de um destes sistemas na área desportiva, mais concretamente, num sistema de apoio a treino especializado de jovens atletas que praticam futebol.

Desse modo, o principal objetivo deste documento é, primeiramente, elaborar um estudo de investigação sobre a área em que o projeto se enquadra, nomeadamente, o setor do desporto, bem como, a área de inteligência artificial associada a sistemas de recomendação, e de seguida, dar a conhecer ao leitor todo o processo de desenvolvimento do projeto *SmartCoach*, desde o design elaborado, até à solução final.

Relativamente à solução *SmartCoach*, esta, consiste num protótipo de uma aplicação *web* multi-dispositivos, destinada a todos os membros de uma equipa de futebol, possibilitando assim, o treinador de gerir a sua equipa. E, por outro lado, permitindo que os jogadores tenham acesso a uma vasta lista de informações relevantes provenientes de um mecanismo de recomendação que proporcionará um aumento da qualidade da performance individual.

Como forma de validação do software desenvolvido foram elaborados inquéritos de satisfação, nestes os inquiridos avaliam as funcionalidades e verificam a utilidade geral da aplicação. Ao analisar as respostas do inquérito é possível verificar que todos os utilizadores concordam que a aplicação seria útil na sua equipa. Já em relação à classificação final dada à aplicação, pode-se verificar que 70% dos utilizadores inquiridos classificam a mesma como boa, já os restantes 30% classificam esta, como muito boa. Por fim, pode-se assim concluir que a solução obtida satisfaz todos os seus utilizadores respondendo com qualidade aos requisitos impostos.

Palavras-chave: Sistemas de recomendação, Desporto, Gestão de equipas, Treino especializado

Abstract

In an increasingly information-intensive society, it's necessary to facilitate access to data, thus is why recommender systems emerge. Currently, the use of these tools has a greater adherence in the online commerce platforms (*e-commerce*), however, these tools can be applied in several areas. Therefore, it's intended to test the usability and utility of one of these systems in the sports field, more specifically, in a support system for specialized training for young athletes practicing football.

Thus, the main objective of this document is firstly to prepare a research study on the area in which the project fits, namely the sports sector, as well as the area of Artificial Intelligence associated with recommendation systems, and then let the reader know the entire development process of the project, from the design to the final solution.

Regarding the developed solution, that implies the production of an application for all members of a football team, allowing the coach to manage his team. And, on the other hand, it allows the players to have access to a vast list of information coming from a recommendation mechanism, which will provide an increase in quality in their individual performance.

As a way of validating the developed software, satisfaction surveys were elaborated, in which the users evaluate the functionalities and verify the general utility of the application. By reviewing survey responses, one can see that all users agree that the app would be useful to their team. And regarding the final rating given to the application, 70% of users rate it as good, while the remaining 30% rate it as very good. Finally, it can be concluded that the solution satisfies all its users by responding with quality to the requirements imposed.

Keywords: Recommendation systems, Sport, Team management, Specialized training

Agradecimentos

Agradeço ao Professor António Constantino por ter-me atribuído a presente proposta de mestrado, e também por todo o seu apoio no acompanhamento dado ao longo do projeto, assim como, toda a disponibilidade na ajuda da construção deste documento.

Agradeço em especial ao Professor Paulo Matos por toda a disponibilidade e orientações relacionadas com o tema do projeto.

Agradeço ao meu colega Hugo Soares, pelo desenvolvimento do projeto complementar intitulado *SmartCoachManager*.

Agradeço à minha família e amigos pelo apoio durante esta fase da minha vida.

E, por fim, agradeço a todos aqueles que, de alguma forma, contribuíram direta ou indiretamente para a realização deste projeto

Índice

Lista de Figuras

Lista de Tabelas

Lista de Gráficos

Acrónimos e Símbolos

Lista de Acrónimos

AI	*Artificial Intelligence*
AJAX	*Asynchronous Javascript And XML*
BPMN	*Business Process Model and Notation*
CAGR	*Taxa de crescimento anual composta*
CAPSI	*Conferência da Associação Portuguesa de Sistemas de Informação*
CSS	*Cascading Style Sheets*
DTO	*Data Transfer Object*
HTML	*Hyper Text Markup Language*
HTTPS	*Hyper Text Transfer Protocol Secure*
ISAMI	*International Symposium on Ambient Intelligence*
FICC	*Future of Information and Communication Conference*
MVC	*Model-View-Controller*
SR	*Sistema de Recomendação*
TIC	*Tecnologias de Informação e Comunicação*
UC	*Use Case*
USD	*Dólar Americano*

Lista de Símbolos

$	Dólar
€	Euro

1 Introdução

O tema da dissertação desenvolvida está enquadrada na área desportiva e baseia-se na realização de uma aplicação móvel para aconselhamento de treino de jovens atletas. Assim sendo, de modo a introduzir esta temática, neste capítulo, primeiramente, será apresentado o contexto do projeto, de seguida, é realizado o levantamento do problema e identificados os objetivos a atingir com o desenvolvimento do mesmo, por último, é ainda apresentada a estrutura do presente documento.

1.1 Contexto

Atualmente, o papel do desporto na sociedade tem vindo a crescer [1]. O mesmo, desempenha um papel essencial na vida de muitas pessoas, tanto em praticantes como não praticantes (espetadores). O aumento desta tendência está relacionado com inúmeros fatores. Entre eles, é identificado a presença das novas tecnologias na área, esta presença, possibilita o fácil acesso a dados e informações relevantes para atletas e treinadores (casuais ou competitivos), assim como, para o público em geral [1]. Consequentemente, a utilização de aplicações desportivas de informação e notícias é uma prática muito comum por qualquer pessoa, no entanto, os sistemas tecnológicos nesta área não ficam só por aí, uma vez que, o surgimento de softwares para uma vertente mais competitiva também é uma realidade. Estes softwares, tem o principal objetivo de apoiar e simplificar os treinadores e/ou atletas a exercer o seu trabalho.

Por outro lado, com a evolução do fenómeno da AI (Inteligência artificial), surgem novos softwares ligados ao setor. Estes, tem como propósito, apoiar os seus utilizadores na tomada de decisões. Mais detalhadamente, baseiam-se na análise de um conjunto alargado de dados, essencialmente, proveniente de estatísticas de jogos. E, posteriormente, apresentam ao utilizador informações relevantes, a ter em conta, antes do utilizador realizar certas ações.

Assim, de modo a tirar proveito da combinação das novas tecnologias focadas em inteligência artificial ao ambiente desportivo, o presente projeto pretende inovar e causar

impacto, através da produção de um mecanismo que permita a evolução da performance desportiva dos atletas e consequentemente a obtenção de melhores resultados coletivos.

1.2 Problema

As Tecnologias de Informação e Comunicação (*TICs*) estão a ser cada vez mais utilizadas no mundo desportivo, nomeadamente no futebol, quer com o objetivo de potenciar o treino dos atletas, quer no apoio à decisão desportiva [2] [3]. No entanto, os sistemas de gestão de treinos existentes para a formação de jogadores, não relacionam as performances dos treinos e jogos com as características, técnicas, táticas, físicas e mentais dos respetivos atletas no processo de seleção e recomendação do plano de treino [3]. Para além disso, esses sistemas não possuem capacidades de aprendizagem para se adaptar, evoluir e encontrar novas recomendações de treino. Estas limitações tornam os resultados menos positivos, em virtude de não serem focados nas necessidades dos respetivos jogadores.

Para responder a esta matéria, será avaliado se a possibilidade da existência de um mecanismo de recomendação irá permitir alcançar uma melhoria das performances dos jogadores.

Assim sendo, este projeto pretende oferecer suporte a jovens jogadores e aos seus treinadores, para uma melhor análise sobre os seus atletas e suas capacidades, focando na evolução destas através do uso de tecnologia associadas a técnicas de inteligência artificial, como sistemas de recomendação para a sugestão de planos de treino específicos consoante as necessidades dos respetivos jogadores.

O problema identificado aponta então, de uma maneira geral, para a conceção de uma solução que auxilie treinadores a gerirem as suas equipas e que permita jogadores melhorarem as suas performances individuais e coletivas.

1.3 Objetivos

O objetivo do presente projeto passa pelo desenvolvimento de um protótipo, que visa ajudar jovens jogadores de futebol a evoluírem. Para tal, pretende-se desenvolver uma aplicação *web* móvel, com o propósito de facilitar a interação entre treinador e membros do "Staff" técnico de um clube, com os jovens jogadores do mesmo, assim como a interação dos jovens atletas com outros utilizadores, tendo estas interações sempre o mesmo objetivo, o acompanhamento do jovem jogador e a sua potencialização enquanto atleta.

Por outras palavras, o objetivo desta tese consiste primeiramente, na análise e investigação do surgimento das novas tecnologias na área desportiva, nomeadamente o papel dos sistemas de recomendação ligados ao desporto, e de seguida, no desenvolvimento de uma aplicação móvel que deverá permitir o aumento do desempenho do jovem atleta considerando diversos atributos. O sistema a desenvolver deve ser realizado de forma intuitiva e amigável, com o principal propósito de melhorar o acompanhamento de jovens jogadores de futebol assim como a sua potencialização, através do uso de técnicas de Inteligência Artificial.

A aplicação desenvolvida, principalmente direcionada à formação, terá como principais funcionalidades:

- Existência de quatro tipo de perfis únicos na aplicação (Treinador, membro Staff Técnico, Jogador, Utilizador Associado);
- Possibilidade do uso da aplicação coletivamente (em equipa), ou individualmente (só o jogador);
- Possibilitar o registo e a configuração da sua equipa, enquanto treinador;
- Ter numa só plataforma informações de todos os jovens jogadores registados, através de perfis de jogadores;
- Possuir uma interface de recolha de dados estatísticos, que pode ser usada quer por membros do staff técnico assim como por utilizadores associados ao jovem jogador em questão;
- Análise e organização destes mesmos dados recolhidos através de um painel desenhado para tal, onde os jogadores possam ver detalhadamente todas as suas ações recolhidas assim como médias, gráficos, etc…
- Possibilitar a recomendação de treinos, consoante as necessidades dos diferentes jogadores;
- Possibilitar a criação de relatórios baseados nas performances dos jogadores;
- Oferta de uma aplicação responsiva multiplataforma.

Por último, o sistema implementado deve estar operacional no início da época desportiva 2019/2020, para assim, ser possível testar a usabilidade e aceitabilidade do software produzido, em escalões de formação de equipas de futebol amador.

1.4 Estrutura da dissertação

A presente dissertação está estruturada com a seguinte organização:

No primeiro capítulo, referente à Introdução, são definidos o contexto e o levantamento do problema, bem como, os objetivos a atingir no decorrer do projeto.

No segundo capítulo, numa primeira fase, é explorado o contexto do projeto e detalhado o estado da arte relativo a sistemas de recomendação, avaliando possíveis soluções para o sistema de recomendação a ser usado, seguidamente, são identificados softwares existentes que auxiliem treinadores e jogadores de futebol a melhorar a sua atividade desportiva.

No terceiro capítulo, é realizada a análise de valor, através da definição de conceitos importantes, assim como, a identificação da proposta de valor e do modelo *Canvas* aplicados ao projeto em questão.

No quarto capítulo é apresentado o Design da solução, exponde inicialmente uma definição da visão da solução, que dá uma proposta de solução teórica ao problema exposto,

posteriormente, é realizada a análise e conceção do projeto, contendo os requisitos funcionais e não funcionais, e por último, é exibida a arquitetura do sistema baseada no Modelo arquitetural 4+1.

No quinto capítulo é exposta a implementação da solução, detalhando, cada uma das componentes do projeto, nomeadamente, o sistema de recomendação, a aplicação servidora, a aplicação *web* e os casos de uso identificados.

No sexto capítulo é abordada a avaliação do projeto, a partir da identificação das grandezas, hipóteses e da metodologia associada, assim como, da respetiva análise aos resultados obtidos.

Por fim, no sétimo capítulo, são apresentadas as conclusões atingidas com o desenvolvimento deste projeto.

2 Estado de Arte

Nesta secção, será primeiramente, detalhado o contexto em que o projeto se insere, apresentando estudos recentes sobre a área de atividade desportiva. De seguida, será exposto o estado de arte associado a sistemas de recomendação, identificando possíveis abordagens para o software a desenvolver, assim como, aplicações desportivas de auxílio a treinadores de futebol, estas serão analisadas, com vista a identificar a aplicação mais completa em termos de conteúdo e verificar que das aplicações estudadas nenhuma contempla um sistema de treino especializado.

2.1 Contexto

Presentemente, vivemos numa sociedade em que o desporto tem vindo a adquirir uma importância cada vez maior. O mesmo, é considerado um fenómeno social importante em todos os níveis da sociedade moderna, uma vez que proporciona um grande impacto em áreas-chave da vida social afetando as relações, os valores éticos e o estilo de vida das pessoas [3].

Desse modo, sendo o desporto uma atividade com uma elevada importância mundial para a sociedade, o mesmo torna-se num foco de desenvolvimento para as novas tecnologias. Atualmente, já são utilizados sistemas informáticos para descobrir as últimas notícias desportivas, transmissões ao vivo, estatísticas dos jogos, entre outros [4].

Assim, podemos verificar que, existem cada vez mais empresários e investidores a concentram-se nesta indústria, tornando as aplicações móveis ligados ao desporto numa das principais áreas de mercado dos softwares móveis. Segundo um relatório anual elaborado em 2016 pela *Flurry Analytics*, é possível verificar que as aplicações desportivas se tornaram a segunda categoria de aplicativos móveis que mais cresceu, apontando um crescimento acima dos 40%, sendo que esta categoria só foi superada por aplicações relacionadas com redes sociais e de mensagens, como se pode constatar no Gráfico 1 [4].

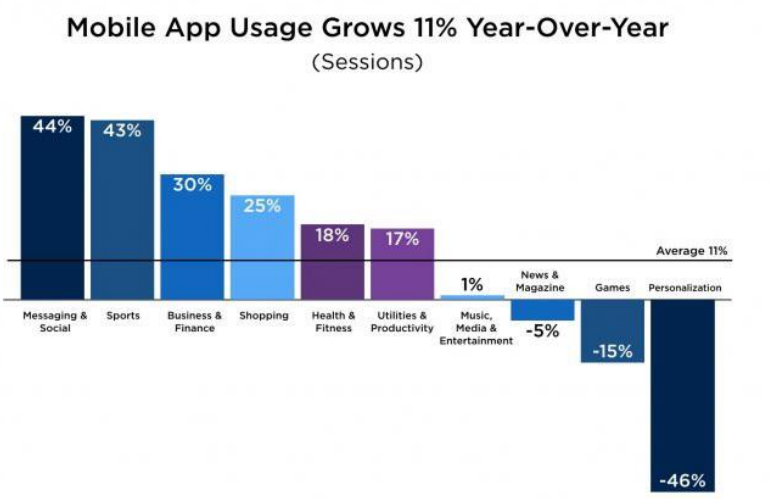


Gráfico 1 - Crescimento das aplicações moveis por áreas [4]

Entretanto, com a evolução das novas tecnologias baseadas em inteligência artificial, manifestam-se diversas aplicabilidades das mesmas ligadas à área desportiva, nomeadamente, o cálculo de previsões relacionadas com os vencedores de jogos ou mesmo de campeonatos [5], até ao desenvolvimento de sistemas de análise de performance desportivas. Assim sendo, com a evolução dos sistemas informáticos, surge a possibilidade do desenvolvimento de novas funcionalidades para softwares desportivos.

O desenvolvimento desses softwares, são destinados a melhorar o rendimento de uma equipa através de análises de dados desportivos, e também, fornecendo apoio técnico à tomada de decisões. Para além disso, o uso de aplicações tecnológicas numa vertente mais competitiva também é uma realidade, pois, obter uma certa “vantagem competitiva” face ao adversário é uma característica cada vez mais importante para os desportos de elite [6].

Em suma, estes softwares são cada vez mais importantes nos dias de hoje, principalmente, devido a possibilitar a melhoria da performance desportiva de uma equipa, através de análises que poderão se aplicar aos seus jogadores ou a jogadores das equipas adversárias identificando os principais pontos fracos a explorar.

Por fim, de modo a tirar partido do aumento da tendência dos softwares baseados em AI, surge a possibilidade de elaborar um sistema de recomendação, com vista a melhorar a performance dos jogadores, através da sugestão de planos de treino específicos, referente às necessidades dos respetivos atletas. De seguida, na secção seguinte, será explorado este assunto, através do esclarecimento e análise dos conceitos relacionados com esta temática.

2.2 Sistemas de recomendação

Nesta secção será estudado o conceito de Sistema de Recomendação, para isso, primeiramente é realizado uma introdução do tema, esclarecendo o conceito e enquadrando o assunto, de seguida, são identificadas as quatro principais técnicas associadas a um sistema de recomendação, assim como, os respetivos problemas e algoritmos mais utilizados. Após isso, são identificados os impactos que os sistemas de recomendação estão a ter globalmente, apresentando como exemplo duas das maiores multinacionais existentes.

Por fim, ainda neste é capítulo, são avaliadas duas soluções para o sistema de recomendação a ser usado, concluindo qual das soluções será posta em prática.

2.2.1 Introdução

Nos dias de hoje, apresentar aos utilizadores as informações mais relevantes é uma tarefa importante a desempenhar por qualquer empresa, especialmente aquelas que estão presentes no setor do comercio eletrónico (*e-commerce*), e que assim disponibilizam recomendações dos seus produtos para os seus consumidores, contudo os sistemas de recomendação não se encontram presentes só no comércio eletrónico, atualmente também é possível identificar estes sistemas em qualquer uma das seguintes áreas: entretenimento (recomendação de filmes, músicas e jogos), conteúdo (recomendação de documentos, artigos científicos, aplicações *e-learning*), serviços (serviços de recomendação de viagens e alojamento), e por fim, até na área social (recomendação de pessoas e conteúdo, como “tweets” nas redes sociais) [7] [8].

Com a evolução do fenómeno da AI (Inteligência artificial), surgiram os sistemas de recomendação. Os sistemas de recomendação são assim, uma subárea do *machine learning,* e tem como objetivo a sugestão de itens a um utilizador, com base no seu histórico de preferências [9].

Mais detalhadamente, os sistemas de recomendação recolhem informações sobre as preferências dos seus utilizadores para um conjunto de itens (como por exemplo, filmes, músicas, livros, destinos de viagem, entre outros), em que as informações possam ser adquiridas explicitamente (normalmente pela recolha de avaliações dos utilizadores) ou implicitamente (monitorizando o comportamento dos utilizadores, como músicas ouvidas, livros lidos, sites visitados, etc). Os SR tentam equilibrar fatores como precisão, novidade, dispersividade e estabilidade nas recomendações, resultando na obtenção de recomendações de qualquer tipo de produtos e/ou serviços [10].

Os sistemas de recomendação podem ainda ser identificados como uma estratégia de marketing, já que tem a finalidade de recomendar produtos que estejam de acordo, com o interesse do utilizador, desse modo, é mais provável que o respetivo utilizador venha a adquirir o produto recomendado [9]. De acordo com um estudo realizado pelo site *ReportsnReports.com*, o mercado de sistemas/motores de recomendação deverá aumentar dos 801.1 milhões USD, obtidos em 2017, para os 4414.8 milhões USD até 2022, resultando num

CAGR de 40,7% entre os anos 2017-2022, esta subida dever-se-á ao aumento no foco para melhorar a experiência do consumidor, assim como, do crescimento na tendência da digitalização [11]. Com isto, podemos concluir que os sistemas de recomendação estão cada vez mais presentes no nosso dia-a-dia, tornando-se até indispensáveis nos ambiente *web* dos *e-commerce*, uma vez que constituem uma forma de aumentar a satisfação do cliente e tomar posições no mercado, cada vez mais competitivo, das atividades dos negócios eletrónicos [12].

2.2.2 Técnicas e modelos

Em relação às técnicas existentes nos Sistemas de Recomendação, estas podem ser classificados em quatro categorias, tendo por base, como a recomendação é calculada, desse modo, são identificadas as seguintes:

- **Baseado em Conteúdo:** Um sistema de recomendação baseado em conteúdo recomenda ao utilizador, produtos que sejam semelhantes ao que ele preferiu no passado, com base na descrição dos itens e no perfil dos interesses do utilizador [13]. Mais concretamente, a recomendação é realizada a partir de *tags* "descritoras" de itens, assim, itens com características próximas destas *tags* serão recomendados [9], ou seja, no processo de recomendação, o mecanismo compara os itens que já foram avaliados positivamente pelo utilizador com os itens que ele ainda não avaliou e procura semelhanças nos atributos do item. Após a pesquisa, o mecanismo retorna os itens semelhantes aos classificados positivamente para os mesmos, poderem ser recomendados ao utilizador [14]. Assim, num cenário de recomendação de filmes, por exemplo, um utilizador que assista e classifique o filme "Matrix" com uma classificação alta, teria recomendações de filmes de ação e ficção científica [9].

Figura 1 - Exemplo de recomendação baseada em conteúdo [14]

- **Filtragem Colaborativa:** A técnica de filtragem colaborativa, pode ser dividia em duas abordagens:
    - **Baseada no Utilizador (*User Based*):** Esta abordagem consiste na recomendação de itens que utilizadores com gosto semelhante preferiram no passado, um exemplo prático seria, "Se um utilizador gostou de A e de B, um outro utilizador que gostou de A também poderá gostar de B" [9].
    - **Baseada em Itens (*Item Based*):** Por outro lado, a abordagem baseada em itens, representa a recomendação de artigos com base no cálculo da similaridade entre os itens tendo em conta as várias avaliações de outros utilizadores para esses itens [15].

  De uma maneira geral este tipo de recomendação apresenta resultados positivos, evitando o problema de recomendações repetitivas.

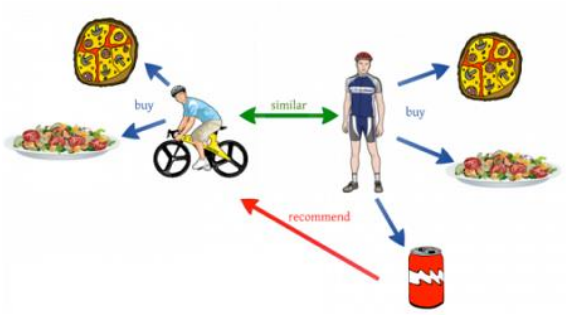


Figura 2 - Exemplo de recomendação realizada por filtragem colaborativa [14]

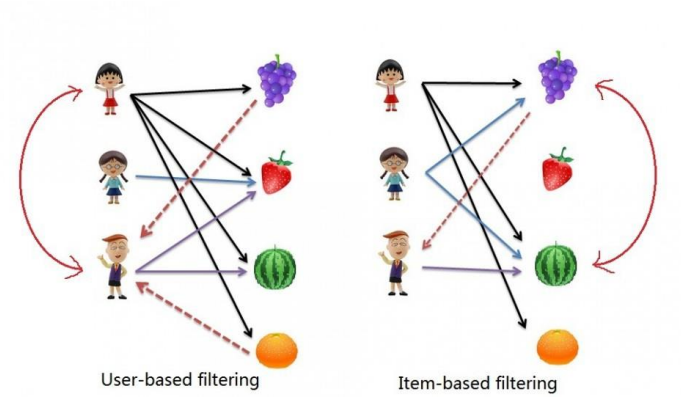


Figura 3 - Diferença entre filtragem baseada no utilizador da baseada no item [15]

- **Baseado em Conhecimento:** Esta técnica recomenda com base no conhecimento acerca da necessidade de um utilizador para com um determinado item. As recomendações são realizadas através do estabelecimento de pesos de conveniência, consequentes do conhecimento que se sabe das relações de um item para com um determinado utilizador [16]. Por exemplo, um sistema que

recomenda viagens baseado em conhecimento pode tirar partido não só do que se conhece acerca da experiência do utilizador em viagens anteriores, mas também do que se sabe sobre as características dos locais que visitou e dos locais disponíveis a recomendar [16].

- **Sistemas Híbridos:** Por fim, um sistema híbrido consiste na combinação das técnicas mencionadas anteriormente (filtragem colaborativa, filtragem baseada em conteúdo e filtragem baseada em conhecimento), de modo a superar as limitações/problemas impostas por cada uma, como por exemplo os problemas de "*Sparsity*" e "*Cold Start*" relacionadas com a filtragem colaborativa ou de "*Serendipity*" da abordagem baseada em conteúdo, os mesmos, serão identificados na secção seguinte [16].

2.2.3 Problemas das Técnicas identificadas

Como podemos depreender qualquer uma das abordagens expostas acima tem os seus pontos fortes, bem como, pontos fracos. Assim sendo, de modo a diferenciar ainda mais as principais técnicas ligadas a sistemas de recomendação, serão de seguida, detalhados os principais problemas referentes a cada uma delas.

Baseado em Conteúdo:

- **Acaso (*Serendipity*):** As técnicas baseadas em conteúdo não têm nenhum método de identificar algo inesperado, desse modo, os sistemas apenas recomendam os objetos que têm grande correspondência com o perfil do utilizador [16].
- **Análise Limitada de Conteúdos**: As técnicas estão limitadas às características que manual ou automaticamente são associadas aos objetos. Se o objeto possuir poucos elementos descritivos associados a recomendação torna-se difícil [17].
- **Conteúdo Inacessível**: Contrariamente aos objetos de texto, muitos tipos de objeto não possuem conteúdos passíveis de serem analisados pelo sistema de informação, dependendo totalmente da informação fornecida pelos utilizadores [17].

Filtragem Colaborativa e Baseado em Conhecimento:

- **Esparsidade (*Sparsity*)**: Fenômeno em que os utilizadores em geral, classificam apenas um número limitado de itens, resultando em falta de informações para determinados itens [18].
- **Escalabilidade**: Os sistemas necessitam quantidade mínima de dados antes de se tornarem eficientes, no entanto, se a quantidade de utilizadores e itens for aumentando exponencialmente, os algoritmos de recomendação irão sofrer sérios problemas de escalabilidade.
- **Novo Utilizador / Novo Item (*Cold Start*)**: O problema ocorre quando não é possível fazer recomendações de confiança devido a uma falta inicial de classificações, um exemplo prático seria o sistema ter que lidar com a chegada de um novo utilizador ou item, ou seja, utilizadores para os quais ainda não

existiu qualquer tipo de registo de atividade e itens para os quais ainda não foram registada qualquer classificação [10].

- **Utilizador Incomum (*Gray Sheep*)**: Utilizadores com perfis e classificações incomuns não recebem recomendações muito precisas, pois não têm muitas características em comum com a maioria dos utilizadores [17].

Sistemas Híbridos:

- **Maior Complexidade**: Como o sistema irá combinar duas ou mais técnicas de recomendação, o mesmo terá uma maior complexidade, bem como, um maior custo de implementação [16].

2.2.4 Algoritmos mais usados

Na presente secção são apresentados os principais algoritmos associados às técnicas de recomendação identificadas.

2.2.4.1 SlopeOne

O *Slope One* é um algoritmo de Recomendação relativo à técnica de Filtragem Colaborativa, do tipo "*Item-Based*", o mesmo tem uma implementação fácil, e apresenta bons resultados práticos, sendo altamente escalável. O objetivo do algoritmo é calcular predições a partir da comparação entre avaliações de utilizadores a certos itens.

O algoritmo funciona, considerando que um utilizador tenha atribuído classificações não binárias a certos itens. Essas classificações são colocadas numa matriz de Utilizadores x Itens, de tal forma que cada linha corresponda às classificações de um utilizador j a N itens. Se um utilizador j não tiver dado classificações a um item i, o respetivo elemento fica igual a 0. De seguida, na Figura 4, é apresentada a matriz com o conjunto de classificações de determinados utilizadores a um conjunto de itens [9].

Items

Users					
	3	0	3	2.5	4
	1.5	0	4	0	5
	0	1	1.5	1	0
	4	3	0	1.5	4.5
	2	2.5	0	2	4
	5	0	4.5	0	0
	1	2	1	0	3.5
	0	5	0	1	4

Figura 4 - Matriz de classificações de determinados utilizadores a certos itens [9]

Observando a mesma matriz, podemos concluir que uma linha j representa as classificações dadas por um certo utilizador a todos os itens no espaço definido. Já uma coluna i, representa as classificações recebidas em relação ao item i, pelos diferentes utilizadores existentes. A partir dessa matriz, podemos obter relações entre os dados, tornando possível gerar uma interpolação matemática e prever qual seria a classificação dada por um utilizador j ao item i que ele ainda não avaliou [9].

Num exemplo mais concreto, supomos que temos, um utilizador A que deu classificação 2 para um item i, e classificação 4 a um item j, e por outro lado, temos um utilizador B que deu classificação 3 para o item i. Através do algoritmo *SlopeOne*, é possível calcular a predição da classificação que o utilizador B daria ao item j, com a seguinte equação:

$$(2 - 4) = (3 - ?)$$
$$? = 2 + 3$$
$$? = 5$$

De acordo com o algoritmo, a previsão da classificação que o utilizador B daria ao item j seria de 5. Apesar do exemplo apresentado acima ser destinado a unicamente dois utilizadores e dois itens, é possível aplicar o algoritmo *SlopeOne* a um conjunto mais alargado de dados. Assim, para análise de um maior conjunto de dados, será necessário obter a média das diferenças entra as classificações dos utilizadores aos itens existentes. Para isso, é apresentada, de seguida, a fórmula geral para cálculo das predições [9].

$$P(A,i) = \frac{(R(A,j) + Diff(i,j)) + (R(A,k) + Diff(i,k)) + \ldots + (R(A,z) + Diff(i,z))}{N}$$

O objetivo desta fórmula será calcular a predição da classificação dada do utilizador A para o item i. Supondo que tenhamos N itens que variem entre i a z, aonde α é o item corrente a analisar, é realizado o somatório da média das diferenças de avaliações entre os itens i e α para todos os outros utilizadores (Diff (i, α)) e o valor que o utilizador A deu de classificação ao item α (R (A, α)).

2.2.4.2 k-Nearest Neighbors

O algoritmo de recomendação *k-Nearest Neighbours* (kNN) é o algoritmo de referência para o processo de recomendação de filtragem colaborativa [10], e tanto pode ser aplicado a abordagens baseadas no utilizador como baseadas nos itens. As suas principais vantagens, são a simplicidade e resultados razoavelmente precisos. Já em relação às desvantagens são, a escalabilidade e vulnerabilidade à escassez da base de dados do sistema de recomendação [10]. Essencialmente o algoritmo calcula a similaridade e recomenda itens que os vizinhos (k) mais próximos "gostam" [19]. Assim sendo, o método responsável pelas recomendações baseia-se então nas três seguintes etapas:

1. Usando a medida de similaridade selecionada, produzimos o conjunto de k vizinhos para o utilizador a. Os k vizinhos são os k utilizadores com os valores de similaridade mais próxima de 1.
2. Após o cálculo do conjunto de k utilizadores (vizinhos) semelhante ao utilizador, é realizada a previsão do item i para utilizador a, essa operação pode ser realizada por uma das seguintes abordagens de agregação: a média, a soma ponderada ou a desvio médio.
3. Por fim, para obter as principais recomendações, escolhemos os n itens, que possuem os valores previstos mais altos para o respetivo utilizador [10].

Na Figura 5 é possível visualizar um caso de estudo, aonde foi utilizado o algoritmo *kNN* do tipo "*user-based*". Para este caso, foram utilizadas as seguintes "regras":

K: 3;

Agregação: Média;

Medida de similaridade: $1 - \frac{\sum_{i=1}^{n}\sqrt{(A_i - B_i)^2}}{n}$

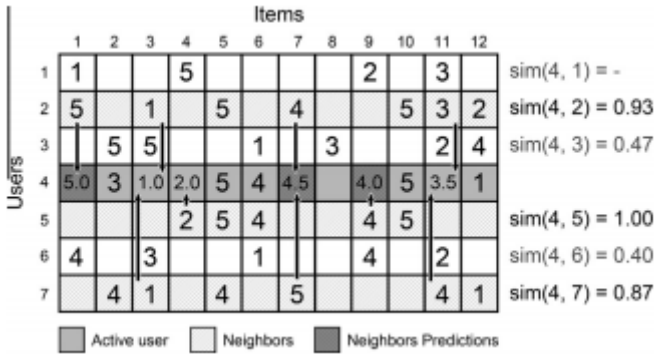


Figura 5 - Exemplo da aplicação do algoritmo kNN [10]

Em relação ao cálculo das medidas de similaridade, para além da simples "diferença das médias quadradas" identificada acima, são normalmente utilizados os métodos: "*Cosine similarity*" e "*Pearson correlation*", apresentados abaixo [20].

Cosine similarity:

$$sim(\vec{a}, \vec{b}) = cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}|| \times ||\vec{b}||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Pearson correlation:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

2.2.4.3 Matrix Factorization

Para reduzir os problemas de altos níveis de esparsidade e escalabilidade nas bases de dados dos sistemas de recomendação, podem ser utilizadas diversas técnicas de redução de dimensionalidade. Os métodos de redução são baseados na factorização de matrizes. [10]

Entre eles temos a decomposição do valor singular (*SVD - Singular Value Decomposition*), que é considerada uma das técnicas existentes de redução de dimensionalidade. Desse modo, a *SVD* de uma matriz A de m × n, pode ser obtida pela fórmula [21]:

$$SVD(A) = U\Sigma V^T$$

Onde,

U e V são m × m e n ×n matrizes ortogonais respetivamente;

Σ é a matriz ortogonal singular m × n com elementos não negativos.

A matriz U m × m é denominada matriz ortogonal se U^T for igual a uma matriz m x m de identidade. Os elementos diagonais em Σ (σ1, σ2, σ3, ... σn) são chamados valores singulares da matriz A, geralmente, os valores singulares são colocados na ordem decrescente em Σ. Os vetores da coluna de U e V são chamados “left singular vectors” e “right singular vectors”, respetivamente [21].

O SVD possui muitas propriedades desejáveis e é usado em diversas aplicações. Uma das suas propriedades é a baixa aproximação de classificação da matriz A. A SVD truncada da classificação k é definida como [21]:

$$SVD(A_k) = U_k\Sigma_k V_k^T$$

Onde U_k e V_k são m × k e n × k matrizes compostas pelas primeiras k colunas da matriz U e as primeiras k colunas de matriz V, respetivamente. A Σ_k Matriz é a submatriz diagonal do princípio k × k de Σ.

A_k representa a aproximação linear mais próxima da matriz original A com classificação reduzida k.

Uma vez concluída a transformação, o utilizador e os itens podem ser considerados como pontos no espaço k-dimensional.

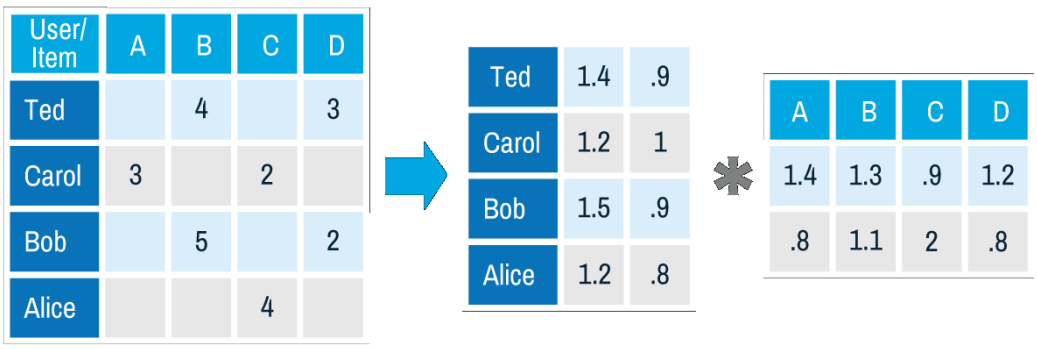


Figura 6 - Exemplo do uso da Factorização de matriz [19]

Por fim, um exemplo prático seria, após a decomposição da matriz acima, poderíamos prever a classificação que o Ted daria ao item A, através do cálculo do produto escalar entre os vetores (1,4; .9) associado ao utilizador Ted e (1.4; .8) associado ao item A. Como resultado, obtemos o valor 2.68 [19].

2.2.5 Impacto dos Sistemas de Recomendação

Com o advento do consumo em dispositivos móveis e a propagação do *e-commerce*, sistemas de recomendação tornaram-se um tema extremamente atrativo. Através de algoritmos simples e facilmente integráveis com aplicações *web*, eles agregam valor ao negócio *online*, promovendo itens de consumo direcionados a um público alvo. Assim, esta tendência resultará numa maior satisfação por parte dos consumidores. "*Quando um cliente sente que é compreendido pela marca, é mais provável que ele permaneça fiel e continue comprando pela mesma.*" [9] Posto isto, os sistemas de recomendação são imprescindíveis, uma vez que são eles que calculam as várias preferências implícitas ou explicitas dos utilizadores e fornecem aos clientes sugestões personalizadas e específicas para as suas necessidades [22].

De seguida, serão identificadas de que forma os sistemas de recomendação de duas das maiores empresas mundiais de setores distintos tem impacto no seu negócio.

2.2.5.1 Netflix

A *Netflix* é um serviço de *streaming* que permite aos clientes verem uma variedade de séries, filmes e documentários em dispositivos ligados à internet [23]. O seu sistema de recomendações contém dezenas de algoritmos em consideração e compara o consumidor com outros (milhares/milhões) de utilizadores com gostos semelhantes em mais de 190 países onde o serviço da *Netflix* está disponível [24]. De acordo com um estudo recente por parte da "*McKinsey*", até 75% do que os consumidores assistem na *Netflix* vem do sistema de recomendação da empresa [25].

Por outras palavras, a maioria dos utilizadores da *Netflix* já não realiza pesquisas sobre quais os conteúdos que poderá estar interessada, esse trabalho é realizado automaticamente pelo sistema de recomendação, os mesmos economizam não só o tempo do utilizador, mas também o capital da empresa, uma vez que, segundo o artigo "*The Netflix Recommender*

System: Algorithms, Business Value, and Innovation", produzido pelos executivos da *Netflix* (Carlos A. Gomez-Uribe e Neil Hunt), o efeito combinado de personalização e recomendações poupa mais de um bilhão de dólares por ano [26].

2.2.5.2 Amazon

A *Amazon*, é sem dúvida uma das maiores, senão a maior empresa de comércio eletrónico (*e-commerce*) do mundo, a mesma, disponibiliza aos seus utilizadores uma vasta variedade de produtos e recomenda aqueles que melhor se aplicam às preferências dos seus consumidores, através de um sistema de recomendação baseado no uso da filtragem colaborativa [27]. É de salientar que segundo o estudo realizado pela "*McKinsey*" 35% das vendas da empresa provêm do seu sistema de recomendação [25], o que também levou a um aumento significativo (de 29%) nas vendas totais, elevando seu volume de vendas anual em 2016 para 135,99 bilhões de dólares. Esses valores devem-se não só às recomendações realizadas no seu site, mas também às recomendações recebidas via e-mail, as quais, incrivelmente, tem uma maior taxa de sucesso [22] [28].

2.2.6 Sistemas de Recomendação ligados ao desporto

Hoje em dia, o desporto tem um papel imprescindível na sociedade, segundo Nelson Mandela, *"O desporto tem o poder de mudar o mundo. O desporto consegue unir as pessoas como mais nenhuma atividade. O desporto fala às pessoas numa linguagem que todos podem compreender".* Desse modo, sendo o desporto uma atividade com uma gigantesca importância mundial, o mesmo torna-se um foco de desenvolvimento para as novas tecnologias. Assim, o uso de aplicações tecnológicas está difundido em diversas áreas desportivas e a adoção dessas ferramentas para obter uma "vantagem competitiva" é uma característica cada vez mais importante para os desportos de elite, como é o caso do futebol, basquetebol, entre outros. Estas ferramentas analisam os dados coletados e tem um grande impacto no modo como os atletas são monitorizados, uma vez que os sistemas de apoio à decisão forneçam conselhos úteis em relação a treinos e outros desempenhos avançados [6] [29].

Num mundo desportivo cada vez mais equilibrado, onde os pequenos detalhes separam a vitória da derrota, as análises desportivas ganham força como um fator diferencial. Da mesma maneira, que num ambiente de negócios onde a análise de dados oferece grandes possibilidades de melhor desempenho, no ambiente desportivo a mesma coisa acontece, os clubes que decidem fazer essa análise estão a ganhar vantagens decisivas face aos seus adversários, possibilitando equipas pequenas fazerem frente a equipas que teoricamente são favoritas [30].

A análise de dados no ambiente desportivo permite que o treinador, os jogadores e os gerentes avaliem objetivamente e, assim, melhorem seu desempenho. O uso de abordagens computacionais e estatísticas na análise das suas performances está a ganhar popularidade, especialmente, em relação à comparação e evolução do desempenho do atleta, visualização e previsão do resultado do jogo [31].

Como foi dito anteriormente, as competições desportivas são decididas por pequenos detalhes, portanto, os clubes querem ter sob controle qualquer aspeto que se possa tornar decisivo. Logo, são contratadas pessoas especializadas em analisar os adversários para saber antecipadamente os seus pontos fortes e fracos, para assim, ter mais recursos para poder enfrentá-los. No entanto, devido ao vasto número de variáveis existentes na análise, esta não é uma tarefa fácil de realizar. Assim sendo, graças ao "*Sports Analytics*", que pode ser definida como a área de investigação de desempenho desportivo profissional usando técnicas científicas, (esta disciplina frequentemente emprega princípios e técnicas de estatística, "*data mining*", teoria dos jogos, biomecânica, cinesiologia, etc) [31] esta tarefa pode ser automatizada, tornando o processo mais eficiente, sendo capaz de analisar inúmeros padrões de o jogo através de mecanismos de *Machine Learning*, assim, torna-se possível ter uma melhor preparação para competições e até mesmo fazer correções durante a competição em tempo real. Para além que a análise pós-competição ganha valor, com informações de qualidade obtida através desta ferramenta [30].

Juntamente com a evolução do "*Machine Learning*" na área desportiva, surgem também os sistemas de recomendação, estes têm o objetivo claro de ajudar os atletas e/ou treinadores a obter melhores resultados na sua respetiva área.

Um exemplo prático seria o desenvolvimento de um sistema de recomendação para prever o sucesso dos arremessos nos jogos de basquetebol da NBA. Como sabemos, no basquetebol, conhecer as probabilidades de sucesso de um arremesso é o ideal, todavia, podemos concluir que existem diversos fatores que afetam o sucesso do mesmo, nomeadamente, a localização, o tipo de arremesso (estilo) e, claro, a habilidade do jogador em causa [32]. Assim sendo, de acordo com o artigo científico "*Shot Recommender System for NBA Coaches*", foi realizado um estudo, com vista a produzir um sistema de recomendação para um maior aproveitamento dos arremessos no basquetebol, no mesmo, o sistema de recomendação desenvolvido baseia-se no algoritmo "*Matrix Factorization*" que analisando dados referentes a épocas anteriores, calcula a probabilidade de encesto da posição do respetivo jogador com o tipo de arremesso utilizado. No mesmo, é então possível concluir que arremessos pelo centro, a menos de 8 metros, realizados com o estilo "*Cutting Dunk Shot*", tem mais hipótese de dar cesto [32].

Outro exemplo estudado passa pela implementação de um sistema de recomendação que permitisse maratonistas alcançar um novo recorde pessoal, através da sugestão de um novo tempo para completar a prova, que seja desafiador, mas ao mesmo tempo realizável, assim como, de um plano de corrida, que indique a velocidade média a que o atleta deve obter a cada 5km da corrida, na Figura 7, é possível visualizar um exemplo dessa recomendação, mostrando os componentes de tempo de finalização e de velocidade média entre o nPB (*non Personal Best*) e o PB (*Personal Best*).

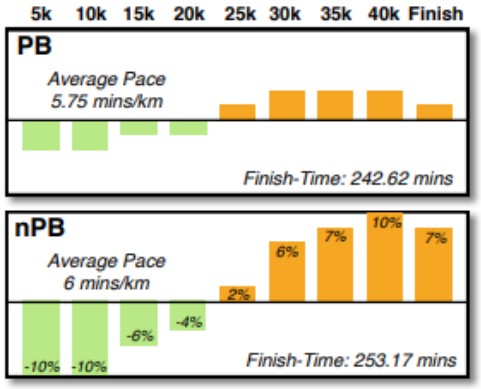


Figura 7 - Exemplo comparativo de um plano de corrida entre nPB e PB

A Maratona é um desporto popular de participação em massa que atrai milhões de atletas para as ruas, esta modalidade tem vindo a crescer um pouco por todo o mundo e atrai corredores de todas as idades. Correr uma maratona não é fácil, completar o percurso de 42.2 km no dia da corrida envolve dedicação e empenho após meses de treino [33].

Assim sendo, de modo a facilitar os atletas atingirem novos recordes pessoais, é previsto um novo tempo, bem como, recomendado um plano para a respetiva corrida. Estes valores são calculados através da análise de um conjunto de dados, mais precisamente, 387.077 registos de corrida individual, provenientes dos últimos 12 anos da Maratona de Chicago, que posteriormente relacionados com o do histórico dos melhores tempos pessoais obtidos do respetivo atleta, é então possível prever um novo "melhor tempo pessoal" que seja realizável para ser batido numa corrida futura. Mais detalhadamente, este valor é calculado pela previsão de similaridade associada a um previsão de erro, obtida a partir de três algoritmos padrão de "machine learning", nomeadamente, linear regression (Reg), k-Nearest Neighbours (kNN) e elastic nets (EN), cada um deles normalmente gerará diferentes previsões de "Personal Best Time", o que levará a diferentes planos de corrida, no fim, será apresentando aquele que tiver menor taxa de erro. [33].

2.2.7 Possíveis abordagens para o Sistema de Recomendação

Após o estudo das várias técnicas existentes de sistemas de recomendação, é necessário avaliar qual a melhor abordagem a seguir para o desenvolvimento do projeto. O objetivo da ferramenta a implementar seria obter os melhores treinos (Itens) adequados a cada jogador (Utilizador), com base na análise de dados existentes no sistema. Assim sendo, foram identificadas duas possíveis abordagens, relativas à implementação do sistema de recomendação a ser usado, estas serão detalhas abaixo.

2.2.7.1 Baseada em Conteúdo

A primeira abordagem passa pela elaboração de um sistema de recomendação baseado em conteúdo, permitindo assim, obter os treinos associados às necessidades especificas dos jogadores, baseando-se na análise de dados estatísticos pós jogo/treino. Mais detalhadamente, seriam obtidas e analisadas as estatísticas dos últimos jogos e treinos do jogador, de modo a

verificar quais os atributos que o jogador precisaria de melhorar. Após esse levantamento de dados, o sistema recomendaria um plano de treino destinado a melhorar a performance do atleta, este plano irá conter um conjunto de exercícios de treinos específicos, associados às capacidades a aperfeiçoar por parte do jogador.

Esta abordagem caracteriza-se pela vantagem de atribuir treinos unicamente com base nas estatísticas recebidas, não dependo assim de informação dos utilizadores e dos seus históricos de planos de treinos, o que permite de uma maneira mais simplificada a atribuição de treinos específicos aos respetivos jogadores, todavia, esta solução pode levar a resultados menos positivos, uma vez que a mesma, não avalia o sucesso dos jogadores relacionado com os seus planos de treino previamente sugeridos, resultando assim, na recomendação de planos de treino menos eficientes.

2.2.7.2 Híbrida

Por outro lado, foi explorado a possibilidade de o sistema de recomendação, seguir um processo de recomendação Híbrido, de modo, a extrair o melhor de cada tipo de técnica dos sistemas de recomendação analisados. O mesmo, analisaria as estatísticas recebidas e os planos de treinos previamente sugeridos, de modo, a propor o melhor plano de treino possível ao respetivo jogador.

Mais detalhadamente, o sistema de recomendação híbrido seria dividido em duas etapas, para assim, combater os problemas "*serendipity*", "*cold start*", "*gray sheep*", entre outros. A primeira etapa seria baseada em conteúdo, em efeitos práticos, as estatísticas obtidas seriam analisadas e aquelas que estejam abaixo de um certo valor identificado pelo treinador e/ou pela média realizada da equipa, seriam associadas a um conjunto de exercícios treinos recomendados pelo sistema, relacionados com a respetiva posição do jogador.

Após existir uma base de conhecimento média/alta, seria posta em prática a segunda etapa, baseada em filtragem colaborativa, na qual seriam recomendados planos de treino de jogadores semelhantes, ou seja, jogadores da mesma posição que alcançaram sucesso no passado ao seguir os respetivos planos de treino. De seguida, são então exemplificadas ambas as etapas:

- **1º etapa** (Baseada em Conteúdo)**:** jogador A (médio) teve estatísticas dos passos curtos abaixo de um valor previamente definido pelo treinador OU abaixo da média realizada pela equipa/pelos colegas da mesma posição, e por isso, são recomendados um ou mais treino(s) que estejam associados à TAG passe curto.
- **2º etapa** (Filtragem Colaborativa)**:** jogador A (médio) teve estatísticas fracas nos passes curtos, o jogador B (médio) no passado, com o plano de treinos X conseguiu melhorar os seus passes curtos. Desse modo, seria recomendado o plano de treinos X do jogador B ao jogador A.

De modo, a identificar qual etapa/técnica de recomendação se deve usar, deverá existir um algoritmo que verifique consoante os dados existentes se existe conhecimento suficiente para ser realizada um recomendação via filtragem colaborativa, de seguida, na Figura 8 é simplificado o processo de recomendação híbrido.

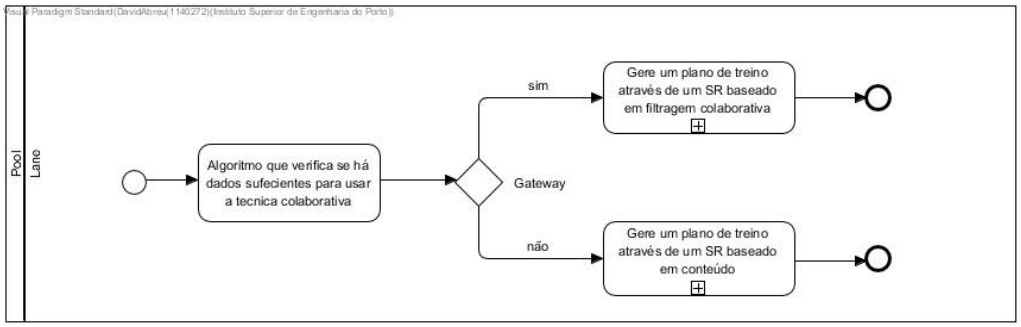


Figura 8 - Processo de recomendação híbrido

Por fim, podemos concluir que esta abordagem tem uma vasta lista de vantagens, como dar resposta a alguns dos problemas expostos na secção **2.2.3 - Problemas**, que ao utilizar unicamente uma das técnica de recomendação poderá levar, no entanto, a mesma, peca pela sua complexidade, uma vez que, teriam de ser implementados algoritmos de recomendação baseados em conteúdos, assim como, baseados em filtragem colaborativa. O que poderia ser, de certo modo, desnecessário caso não existam dados suficientes para realizar a segunda etapa, baseada em filtragem colaborativa.

2.2.7.3 Conclusão

Em suma, tendo em conta a descrição das duas abordagens é possível verificar que ambas as soluções têm lados positivos, assim como, lados negativos. Contudo, após uma análise cuidada por parte de todos os intervenientes deste projeto, foi concluído que se optará pela implementação da primeira abordagem, ou seja, o sistema de recomendação será baseado em conteúdo, isto devido à escassez de dados que irá existir na base de conhecimento para o período de avaliação do software, assim como, a elevada complexidade que o desenvolvimento da abordagem híbrida requer. No entanto, a segunda abordagem não será posta de parte, uma vez que, o objetivo seria a sua implementação em trabalho futuro, de modo, a tornar o software mais completo e eficiente no processo de recomendação de treinos.

2.3 Aplicações para treinadores de futebol

Nesta secção serão identificadas cinco aplicações existentes no mercado que de certo modo auxiliam treinadores de futebol a gerirem a sua equipa, é de salientar que não foram encontrados softwares destinados unicamente a atletas que permitissem de forma direta melhorar as suas performances, no entanto, algumas das aplicações estudadas, são também dirigidas aos jogadores, possibilitando assim, a visualização e análise dos exercícios de treino propostos.

Os softwares analisados foram obtidos através da procura nos motores de pesquisa, (*Google* e *Apple Store*), mediante os conceitos chaves, "aplicações de auxílio à gestão de equipa no futebol" e "softwares de recomendação de treinos no futebol", após isso, foram considerados os softwares mais usados nesta área em Portugal, em que os mesmos tivessem uma classificação de avaliação superior a 4, na escala de 1-5. Os softwares obtidos, dividem-se entre aplicações móveis e aplicações browser/desktop, e estão disponíveis tanto para sistemas operativos IOS, como, Android, no caso das aplicações móveis. Foram apenas apresentadas cinco aplicações, uma vez que, por norma os softwares encontrados não apresentam funções exclusivas, ou seja, os softwares são bastantes semelhantes entre si, oferecendo aos seus utilizadores funcionalidades base bastante idênticas.

De seguida, para cada um dos softwares analisados serão numa primeira fase, detalhadas as suas funcionalidades, apresentando o respetivo preço e as suas principais qualidades. Após isso, será realizada uma comparação dos cinco softwares expostos, através de uma tabela e gráfico comparativo, concluindo qual dos mesmos será o mais eficiente para o treinador/jogador efetuar o seu trabalho, tanto em termos gerais, como em termos de qualidade-preço.

Para além disso, é de salientar que esta análise foi realizada com o objetivo de retirar ideias sobre possíveis funcionalidades a acrescentar ao sistema, assim como, constatar que a adição do sistema de recomendação será uma funcionalidade singular e única na aplicação a desenvolver.

2.3.1 Tactical Soccer

Tactical Soccer é uma aplicação desktop e móvel, que permite treinadores de futebol gerirem a sua equipa, assim como, atletas melhorem a sua performance ao longo de uma temporada. Este software foi desenvolvido com o principal objetivo de melhorar a qualidade e produtividade dos treinos diários individuais e/ou coletivos, obtendo assim melhores resultados em ambientes competitivos [34].

Através da simples e intuitiva interface, é possível definir todos os dados sobre a equipa, como os seus jogadores e respetivos detalhes, a calendarização dos jogos da temporada atual, indicando as estatísticas dos jogos realizados, e muito mais. Posteriormente, os jogadores identificados poderão ser convidados a utilizar a aplicação, permitindo uma melhor comunicação e um aumento do espírito de equipa.

Após existirem dados na aplicação é então possível realizar as principais funcionalidades do software, que passam pela criação e partilha de exercícios, sessões de treino e opções táticas por parte do treinador, visualização do calendário da temporada com detalhes do histórico dos jogos realizados, análise de estatísticas individuais e fácil comunicação e partilha de informação entre todos os membros da equipa. Por último, a aplicação permite ainda a exportação de todo o tipo de relatórios e dados para *PDF* e *CSV*, facilitando, assim a sua posterior análise.

Por fim, este software, encontra-se totalmente disponível em inglês, português, espanhol, mandarim e italiano, propiciando a aplicação, o suporte e uma lista de tutoriais em qualquer um dos idiomas descritos anteriormente.

No que toca ao preço do software, existem quatro possíveis ofertas, que se distinguem pelo número de utilizadores a registar. No caso de ser uma licença individual, esta tem um custo de $99.99 com a validade de um ano, ou $9.99 com a validade de um mês. Já no caso de múltiplos utilizadores o preço da licença por utilizador desce, mais especificamente, uma licença de 10 ou mais utilizadores, tem o preço de $59.99 por utilizador por ano, ou $5.99 por utilizador por mês, o que resulta num valor total de $599.99 anuais ou $59.99 mensais para 10 utilizadores.

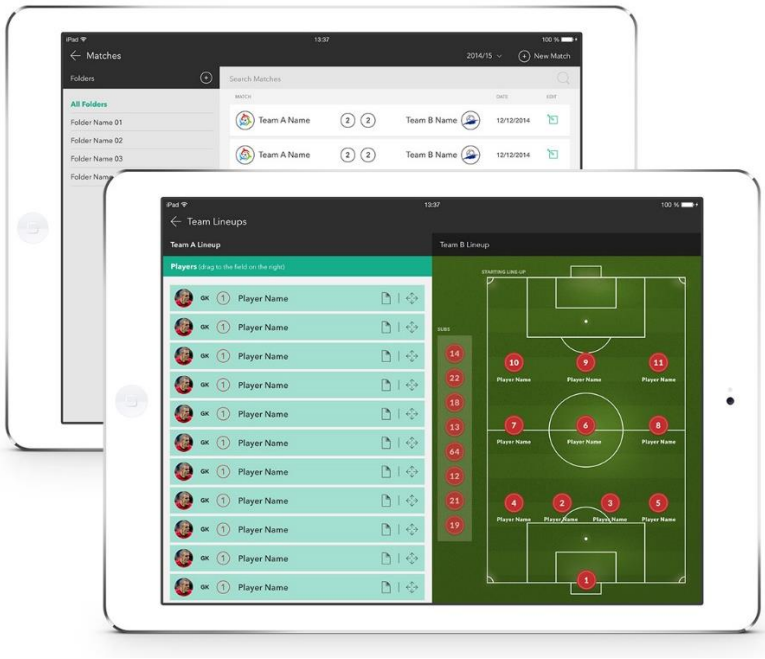


Figura 9 - Gestão da sua equipa na aplicação Tactical Soccer [34]

2.3.2 My Coach Football

My Coach Football é uma aplicação grátis destinada a treinadores de futebol amadores ou profissionais. A mesma, permite orientar os jogos e os treinos de uma equipa, criando e partilhando exercícios específicos com uma comunidade de treinadores, assim como, introduzir as informações sobre os jogos em tempo real graças à aplicação para smartphones, disponível tanto para *Android* como *IOS* [35].

De uma maneira mais prática, *My Coach Football*, tem como principal funcionalidade gerir as várias ações de uma equipa, que vão desde a gestão dos membros da equipa técnica, dos dados pessoais dos jogadores e do calendário da temporada, até à administração dos jogos e preparação dos treinos. Esta aplicação permite ainda o treinador definir as convocatórias dos próximos jogos, notificando os respetivos jogadores e também, a criação dinâmica de novos exercícios de treino, através de um editor com uma interface complexa, no final é possível a

partilha do exercício com a sua equipa e com uma comunidade de utilizadores da aplicação. Por outro lado, como não poderia faltar, este software dispõe de um dashboard, onde proporciona a visualização de estatísticas ligadas aos jogos realizados, referentes à equipa em geral ou mais detalhadamente a cada um dos respetivos jogadores. Por fim, considera-se que este é um sistema eficiente que permite que tanto os treinadores como outros membros do clube guardem as informações temporada após temporada e assistam assim à evolução das equipas e dos jogadores que as compõem.

O software em causa está disponível nos principais idiomas europeus, nomeadamente, inglês, italiano, espanhol e português, e tal como foi dito anteriormente, é um software grátis para qualquer dispositivo.

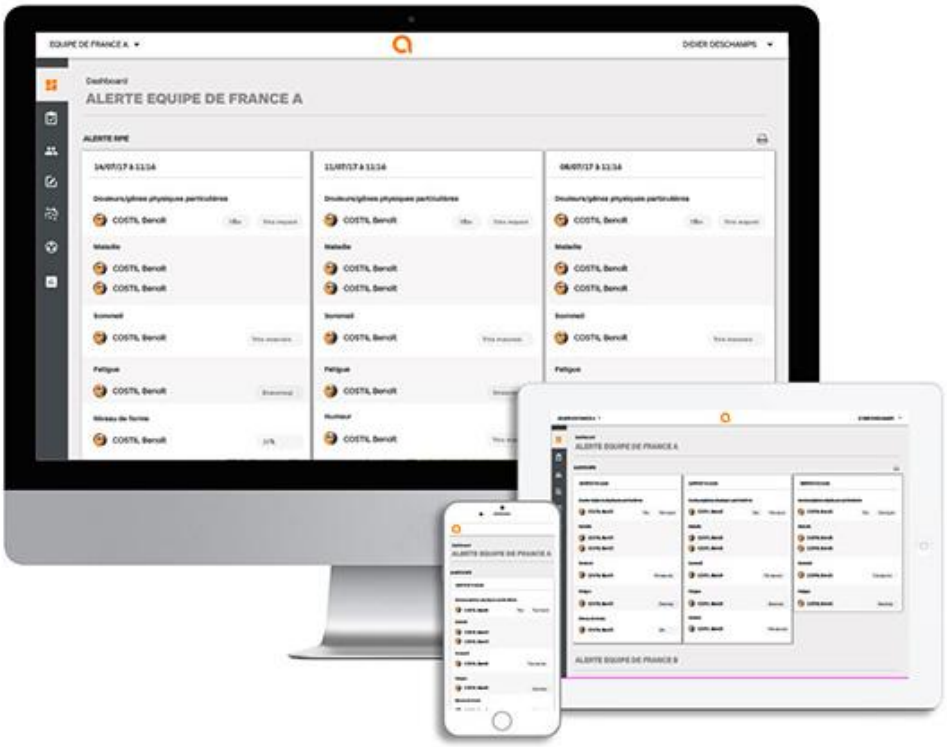


Figura 10 - Aplicação MyCoachFootball nos diferentes dispositivos [35]

2.3.3 Dossier do Treinador

O Dossier do Treinador é uma aplicação *web* portuguesa que tem tudo o que um treinador precisa para realizar o seu trabalho com qualidade durante toda a época desportiva. Esta aplicação está preparada para lidar com equipas técnicas de Futebol 11, 9, 8, 7, 5 e de Futsal, tanto masculinas como femininas [36].

O utilizador só precisa de escolher a variante que quer utilizar e o software fica automaticamente preparado e configurado de forma adequada. O software conta com um design de simples utilização que inclui um editor de exercícios completo e intuitivo, estatísticas automáticas, e mais de vinte fichas referentes a diversas funcionalidades, como, perfis

detalhados de jogadores, convocatórias para os jogos, relatório de jogos, calendário desportivo, mapa de assiduidades e ausências, entre outros. É de salientar que estas fichas poderão ser facilmente exportadas para *PDF* para uma leitura e análise offline.

Por fim, o Dossier do Treinador está presente nas mais variadas redes sociais, proporcionando uma vasta lista de tutoriais, bem como, a possibilidade de uma fácil comunicação com o cliente para esclarecimento de dúvidas.

Já em relação à sua aquisição e disponibilidade, este software está apenas disponível em Português, existindo somente a opção de comprar a sua licença anual de 365 dias, que tem o valor de 24.9 €, a mesma permite o livre acesso a todas as fichas e funcionalidades do Dossier do Treinador, é de realçar que possíveis futuras atualizações na aplicação não têm qualquer custo adicional para o utilizador.

Figura 11 - Exemplo de um plano de treino pela aplicação Dossier do Treinador [36]

2.3.4 SportEasy

SportEasy é uma solução *web* e móvel para gerir equipas amadoras, a mesma pode ser utilizada para qualquer desporto e é manuseada por todos os membros do clube. Este software, considera-se o melhor e mais completo assistente virtual para clubes desportivos [37].

No mesmo, é possível efetuar a fácil gestão dos jogos e treinos de uma equipa, através da criação de eventos, que podem variar entre jogos amigáveis, de competição e treinos, os mesmos ficam disponível no calendário partilhado para todos os membros do clube. Após a criação de um evento, o treinador é capaz de realizar a convocatória para jogo/treino, inserir os dados estatísticos referentes ao mesmo (como, golos, assistências, minutos, entre outros) e escrever observações do evento. É de salientar que é ainda possível para o treinador analisar o desempenho da equipa a qualquer momento da temporada, a partir da observação dos atrasos

e ausências dos jogadores e de estatísticas individuais e/ou coletivas. Este software, permite ainda uma fácil comunicação entre todos os membros do clube e pais dos jogadores via emails e notificações móveis, chat partilhado e fóruns dos jogos. Por fim, SportEasy tem o objetivo de reforçar o espírito de equipa, permitindo os atletas votar no melhor jogador em campo, comentar no chat partilhado e fóruns, bem como, visualizar todas as estatísticas da equipa.

SportEasy é um software gratuito na sua versão "Basic", a mesma não tem acesso a todas as funcionalidades e apresenta publicidade indesejada, no entanto, é possível verificar que ela contempla as principais funcionalidades necessárias para a gestão de uma equipa de futebol, podendo gerir de forma grátis até 30 membros associados à equipa. Para além desta versão, existe também a versão "Premium", com um custo da licença anual por 60€/ano ou da licença mensal por 8€/mês, esta, já abrange todas as funcionalidades, incluindo a apresentação de estatísticas individuais/coletivas, personalização dos perfis dos jogadores e remoção da publicidade, por ultimo, existe também a versão "Club" com um custo de 2€ por membro por ano, esta é destinada a utilizadores que possuem mais do uma equipa, tendo acesso às funcionalidades abrangidas pela outras versões, assim como, a gestão de múltiplas equipas e gestão de membros do clube.

Figura 12 - Gerir a sua equipa pela aplicação SportsEasy [37]

2.3.5 Soccer Coach – Team Sports Manager

Soccer Coach – Team Sports Manager é uma aplicação móvel destinada exclusivamente a treinadores de futebol, só estando disponível apenas para os sistemas IOS. A mesma, oferece aos treinadores a capacidade de gerir todas as ações da sua equipa de uma forma simples, eficiente e poderosa [38].

Neste software, os treinadores podem gerir a sua equipa após a configuração inicial da aplicação, indicando os jogadores existentes e os respetivos detalhes. Após isso, a aplicação permite registar jogos e treinos. No caso de registar um novo jogo, o treinador poderá escolher a sua formação inicial, notificando os jogadores via e-mail e indicar os incidentes da partida em tempo real (golos, assistências, substituições, cartões), após o encontro é possível apontar observações e verificar as estatísticas retiradas do mesmo. Por outro lado, se o treinador optar por registar um novo treino, será facultada uma lista de exemplos de exercícios possíveis de realizar, na versão grátis serão apresentados cerca de 40 modelos de exercícios de treino, no

entanto, este valor pode subir até uma totalidade de 160, consoante a compra de conteúdo extra, estes modelos estão destinados para jogadores de todas as idades e repartem-se em 10 categorias distintas (Ataque e Defesa, Remate, Passe e Receção, etc), cada um deste exercícios contêm, o número de jogadores e o material necessário, bem como, uma descrição e uma demonstração em vídeo animado do respetivo exercício. Por fim, o software também permite a análise da performance individual de jogadores ou da equipa, através da visualização de estatísticas retiradas dos jogos realizados.

Podemos considerar que esta é uma aplicação grátis, no entanto, a mesma tem determinado conteúdo pago, tal como, a compra de uma nova slot para a existência de múltiplas equipas no valor de 5.49€, assim como, 115 modelos de exercícios de treino, estes podem ser adquiridos pela quantia de 54.99€ ou repartidos por pacotes de 5 exercícios no valor de 3.49€ cada.

Figura 13 - Visualização de exercício de treino pela aplicação Soccer Coach [38]

2.3.6 Conclusão

Tabela 1 - Tabela de funcionalidades das aplicações estudadas

Aplicações / Funcionalidades	Tactical Soccer	My Coach Football	Dossier do Treinador	SportEasy	Soccer Coach – Team Sports Manager
Gestão da equipa	X	X	X	X	X
Gestão dos jogos	X	X	X	X	X
Calendário da temporada	X	X	X	X	
Planeamento semanal			X		
Definir convocatória	X	X	X	X	X
Histórico de jogos	X	X	X	X	X
Definir classificações pós jogo				X	
Gestão de treinos	X	X	X	X	X
Exemplos de exercícios de treinos	X	X	X		X
Editor de exercícios	X	X	X		
Perfil de jogadores	X	X	X	X	X
Qualidades do jogador			X		
Mapa de assiduidade			X	X	
Múltiplos utilizadores	X			X	
Atribuição de tarefas				X	
Dashboard	X	X		X	
Estatísticas individuais	X	X	X	X	X
Estatísticas coletivas	X	X	X	X	X
Notificações e alertas	X	X		X	X
Exportação para *PDF*	X	X	X		
Chat para a equipa				X	
Ficheiros partilhados	X			X	
FAQs	X	X	X	X	X
Aplicação Web-Browser		X	X	X	
Aplicação Mobile	X	X		X	X
Aplicação Desktop	X				
Suporte técnico	X	X	X	X	X
Múltiplos idiomas	X	X		X	X

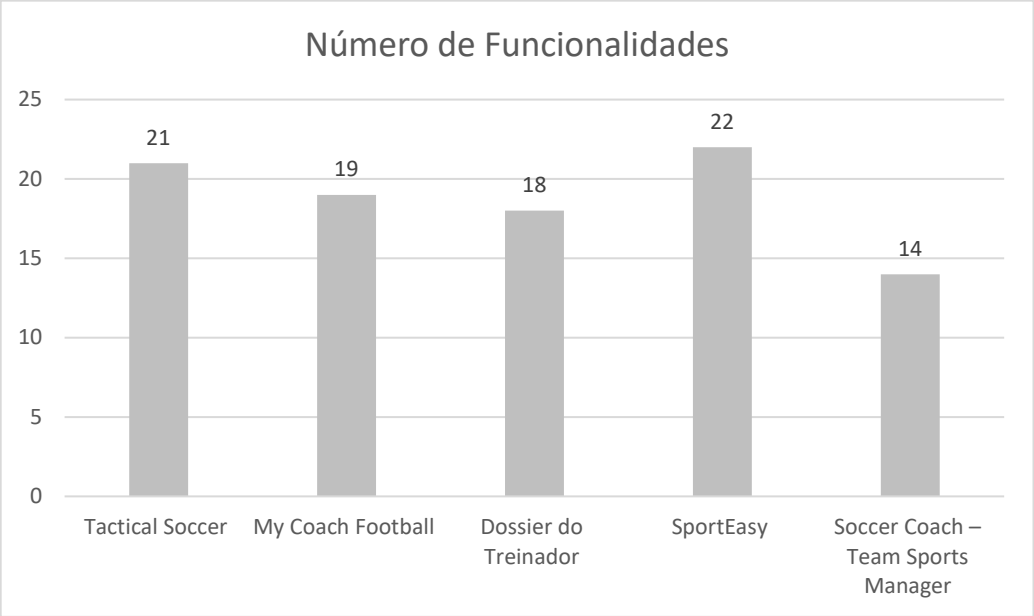


Gráfico 2 - Número de Funcionalidades por Software

Pode-se concluir através da visualização da Tabela 1 e do Gráfico 2, expostos acima, que todas os softwares implementam um vasto número de funcionalidades e possuem as ferramentas básicas para gestão de uma equipa, como gestão dos jogos e treinos, visualização do histórico e análise de estatísticas dos jogos realizados. Com base ainda nesta análise, é possível verificar que o software mais completo seria o *SportEasy*, visto que implementa cerca de 80% funcionalidades identificadas (ou seja, 22 das 28 funcionalidades), no entanto, apesar deste elevado valor, não podemos assumir que este será o "melhor software", em juízo do mesmo conter o maior número de funcionalidades implementadas. Pois, com uma análise mais cuidada, pode-se averiguar que o *SportEasy* não implementa uma ferramenta essencial para treinadores mais focados em melhorar os treinos de uma equipa, que passa pela existência de exemplos/modelos de exercícios de treinos existentes, e/ou um editor de exercícios, ao contrário de outros softwares analisados, que para além disso, permitem a partilha desses conteúdos com jogadores e/ou outros treinadores.

Outro fator diferencial para esta avaliação é a existência de softwares multiutilizadores, como é o caso do *SportEasy e Tactical Soccer*, ou seja, os mesmos são destinados a jogadores, bem como, para a equipa técnica, o que permite uma maior comunicação entre todos os membros do clube, fornecendo acesso a informação útil sobre os mais diversificados dados da equipa a qualquer pessoa associada à equipa.

Relativamente à aquisição dos softwares em causa, podemos afirmar que a aplicação *My Coach Football* é totalmente grátis, já as aplicações *SportEasy e Soccer Coach – Team Sports Manager*, também apresentam versões grátis, no entanto, ambas possuem conteúdo extra que é pago. Em relação, aos softwares *Tactical Soccer* e *Dossier do Treinador*, estes, só podem ser

utilizados consoante a compra de uma licença, contudo, ambos disponibilizam uma versão de demonstração.

Por fim, após testar cada uma das aplicações acima, é possível afirmar que o software mais interessante no mercado será o *Tactial Soccer*, isto devido, ao mesmo, conter as principais funcionalidades que o treinador necessita para gerir a sua equipa, tanto no domínio dos jogos como nos treinos, oferecendo um suporte bastante completo a qualquer utilizador do software. No entanto, apesar deste ser considerado o software mais interessante, tem um preço bastante elevado de $59.99, ou seja, cerca de 52€, por utilizador por ano, desse modo, a escolha do software que apresenta melhor qualidade-preço seria a aplicação *SportEasy* na sua versão premium, que tem um custo total de 60€ por ano para todos os utilizadores.

É ainda de realçar que nenhum dos softwares apresentados implementa mecanismos de Inteligência artificial, tais como, "machine learning" para análise de dados estatísticos e sistemas de recomendação para sugerir certas ações, com vista a melhorar as performances dos atletas. Desse modo, é possível afirmar que o desenvolvimento do presente projeto será único na vertente da existência de um sistema de recomendação de treinos que fornece suporte para os jogadores melhorarem as suas qualidades consoante os dados estatísticos obtidos dos jogos e treinos anteriores.

3 Análise de Valor

Esta secção tem o objetivo de responder a questões relacionadas com a análise de valor do negócio em que este projeto se insere. O principal objetivo de uma análise de valor passa por maximizar o valor de um produto ou serviço, ao menor custo possível [39].

Para isso, começa-se por apresentar o processo de inovação de *Peter Koen*, isto é, o modelo NCD (*New Concept Development*), e aplicar os elementos do mesmo ao presente projeto. De seguida, é identificada a criação de valor, através do esclarecimento dos conceitos de valor, valor para o cliente e valor percecionado, bem como, do reconhecimento dos benefícios e sacrifícios do projeto para o cliente. Posteriormente, é apresentada a proposta de valor do projeto realizado. Por fim, é disponibilizado um modelo de negócio, utilizando o modelo de *CANVAS*, para o projeto em questão.

3.1 Processo de Inovação

Segundo *Peter Koen*, professor especializado na área de negócios, o processo de inovação pode ser dividido em três áreas: o *front end* difuso (FFE), seguido do processo de desenvolvimento de novos produtos (NPD), seguido da comercialização, conforme indicado na Figura 14 [40].

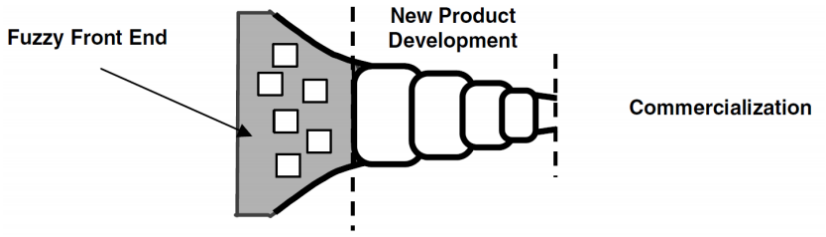


Figura 14 - Processo de inovação definido por Peter Koen [40]

A primeira fase, FFE, é o ponto de partida onde as oportunidades são identificadas e os conceitos são desenvolvidos, antes de entrar no processo de desenvolvimento de novos

produtos (NPD), a mesma, permite ainda a criatividade e criação de valor de uma maneira sistemática. A segunda fase é mais orientada ao objetivo de trazer um novo produto para o mercado e, portanto, mais rigorosa em termos de temporais. Por último, a terceira fase é considerada a comercialização do produto.

Conforme é possível entender, existe uma notável diferença de atividades entre a primeira e a segunda fase. Para corrigir esta diferença, foi desenvolvido o modelo NCD (*New Concept Development*). De seguida, a Figura 15, apresenta esse mesmo modelo.

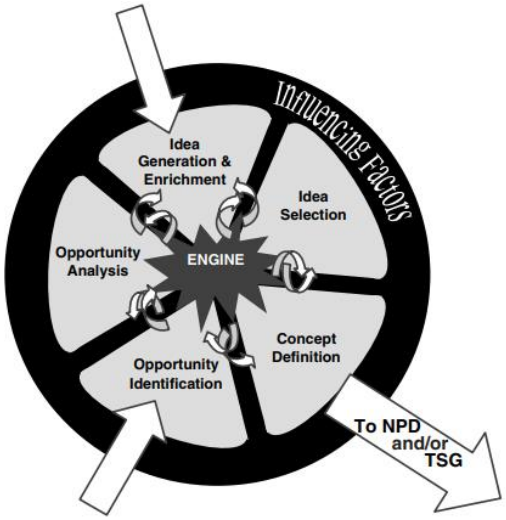


Figura 15 - O modelo New Concept Development (NCD) [40]

O modelo é constituído por três partes chave:

1. O motor, representa a liderança, cultura e estratégia de negócio da organização, o mesmo, serve ainda de apoio aos elementos existentes na área interna;
2. As áreas internas, representam os cinco elementos chave. Nesta área, as ideias e conceitos devem iterar entre os cinco elementos;
3. Os fatores externos, que são geralmente incontroláveis pela organização, e que de certo modo, afetam todo o processo de inovação desde a fase de FFE até à comercialização do produto.

Na figura exposta acima, é ainda possível deduzir que as setas de entrada representam os possíveis pontos de partida no desenvolvimento de um projeto, nomeadamente, identificação de oportunidades e/ou geração de ideias. Já em relação, à seta de saída, esta representa um conceito de desenvolvimento de produto a sair do modelo, quer pelo processo de *New Product Development* (NPD), ou pelo processo *Technology State Gate* (TSG). [40]

Desse modo, podemos concluir que o modelo NCD contêm cinco elementos chave presentes na área interna, entre eles: identificação de oportunidades, análise de oportunidades, geração de ideias, seleção de ideias e definição de conceito. De seguida, são esclarecidos cada

um destes elementos, através de uma breve descrição, e posteriormente os mesmos são aplicados ao projeto a desenvolver.

- **Identificação de oportunidade**: Neste elemento, a organização identifica as oportunidades que deseja seguir [40].
- **Análise de oportunidade**: Neste elemento, uma oportunidade é avaliada para assim se averiguar se vale a pena prosseguir [40].
- **Geração e melhoramento de ideias**: Este elemento, diz respeito ao nascimento, desenvolvimento e maturação de uma ideia concreta, concluindo que a geração de ideias é um processo evolutivo [40].
- **Seleção de ideia**: Neste elemento, as melhores ideias são selecionadas, estas, devem ser obtidas, através de métodos como seleção de ideias a partir de feedback, a partir de um conjunto de metodologias, ou até usando a teoria de opções [40].
- **Definição de conceito**: Neste elemento são definidos os principais aspetos para o projeto ser bem-sucedido [40].

No elemento de **Identificação de oportunidade**, identificou-se a oportunidade de criar um software que fosse capaz de aconselhar treinos específicos às necessidades de jovens jogadores de futebol. Esta identificação foi baseada no aumento da tendência da relação da área desportiva com as novas tecnologias, ou seja, atualmente, existe uma clara crescente de popularidade do uso das tecnologias, nomeadamente, associadas a mecanismos de inteligência artificial para promover uma certa vantagem desportiva face aos seus adversários. Outro aspeto relevante passa pela identificação do número de aplicações deste tipo têm aumentado em consequência do aumento de interesse por parte dos utilizadores neste ramo de aplicações, sejam elas apenas de registo e visualização de dados.

No elemento de **Análise de oportunidade**, é reconhecido através de uma análise ao mercado, que não existem aplicações com um sistema de recomendação de treinos que fornece suporte para os jogadores melhorarem as suas qualidades, o que torna este projeto único nesse sentido.

No elemento de **Geração e melhoramento de ideias**, através de uma análise *brainstorming,* foram discutidas e identificadas um conjunto de ideias, entre elas: desenvolver uma aplicação *web*, assim como, uma aplicação mobile, elaborar um sistema de gestão para equipas de futebol, produzir um sistema de recomendação de treinos e, por último, implementar um mecanismo de controle de dados individuais e coletivos.

No elemento de **Seleção de ideia**, devem ser selecionadas ideias que trazem maior valor para as entidades presentes no projeto, neste caso, os treinadores e jogadores, desse modo, para o projeto a desenvolver as ideias escolhidas são contemplas de seguida:

- Aplicação *web* responsiva;
- Sistema de gestão para equipas de futebol;

- Sistema de recomendação de treinos;
- Mecanismo de controle de dados individuais e coletivos;

No elemento de **Definição de conceito**, ou seja, neste elemento, deve-se definir concretamente os objetivos de um plano de trabalho. Como não existem muitos riscos associados a este projeto, o principal objetivo foi criar um plano de trabalho exequível no tempo disponível com os recursos existentes.

3.2 Criação de Valor

O desenvolvimento de uma nova aplicação deverá atingir a criação de certos valores, para assim, obter o sucesso desejado. De modo, analisar esses valores, serão apresentados abaixo três conceitos importantes neste ramo, nomeadamente: valor, valor para o cliente e valor percecionado, após serem esclarecidos estes conceitos, serão identificados os principais benefícios e sacrifícios para o cliente, de acordo com o presente projeto.

Valor

"O valor é definido em diferentes contextos teóricos como a necessidade de desejo, interesse, crenças, atitudes e preferências." [41] A criação de valor é fundamental para o sucesso de qualquer negócio, seja qual for a atividade de negócio. Qualquer atividade de negócio prende-se com a troca de valor com os seus clientes, seja ele através de um bem ou através de um serviço, quer seja tangível e/ou intangível. [41]

Valor para o Cliente

O valor para o cliente (VC) é uma avaliação realizada pelos clientes relativamente à relação existencial entre os benefícios e os sacrifícios de um determinado valor. Posto isto, é de realçar que os clientes usufruem de produtos ou serviços não só para a sua satisfação, mas também para a sua própria necessidade. [42]

Valor percecionado

"Valor percecionado é a avaliação geral do consumidor sobre a utilidade de um produto com base nas perceções do que é recebido e do que é dado." [43] Enfatizamos ainda mais essa afirmação, dizendo que o valor percecionado pelo produtor significa algo diferente do valor percecionado pelo cliente, ou seja: o produtor é menos sensível ao preço, enquanto o consumidor é mais sensível à qualidade do produto [44].

Benefícios e sacrifícios para o cliente

Relativamente aos benefícios deste projeto para o cliente, foi definido o desenvolvimento de uma aplicação intuitiva para auxiliar treinadores na sua atividade de gestão da sua equipa e jogadores a melhorar a suas performances através de um mecanismo de recomendação de treinos, para além disso, deverá estar presente uma ferramenta de planificação da época desportiva.

Já no caso de sacrifícios, poder-se-á considerar o custo das licenças do software, assim como, da dependência da inserção de múltiplos dados sobre jogos/treinos/jogadores, de modo, a que certas funcionalidades fiquem disponíveis e com resultados coerentes.

Por fim, de modo a sumarizar a avaliação da relação entre os benefícios e os sacrifícios, aplicados ao presente projeto, serão apresentados na Tabela 2 o resumo dos mesmos.

Tabela 2 - Benefícios e sacríficos para o cliente

Benefícios	Sacrifícios
Aplicação inovadora e útil	Custo
Interfaces completas e intuitivas	Dependente de dados
Funcionalidades de gestão	-
Recomendações de treino	-
Calendarização das atividades	-

3.3 Proposta de Valor

Segundo o teórico de negócios, *Alex Osterwalder*, a proposta de valor é entendida como o conjunto de produtos e serviços que criam valor para um segmento específico de clientes [45].

Contudo, analisando melhor a definição, podemos concluir que a proposta de valor é um conceito de marketing que determina, se o que o seu negócio oferece realmente tem valor para os seus clientes. Expondo o motivo pelo qual um cliente deve escolher um produto ou marca concreta, entre todas as outras alternativas do mercado [46]. Assim, é possível afirmar que a proposta de valor deve descrever o produto de forma simples e clara, diferenciando o mesmo dos seus concorrentes diretos com a razão pela qual os clientes devem optar pelo produto da sua empresa, e não o da concorrência.

De seguida, é então definida a proposta de valor para o presente projeto:

"*Oferta de uma aplicação destinada a treinadores de futebol e seus jogadores que disponibiliza suporte técnico para gerirem toda sua equipa ao longo de uma época desportiva, a mesma é ainda capaz de melhorar a performance dos jovens jogadores através de recomendação de planos de treinos específicos para as necessidades dos respetivos atletas.*"

3.4 Modelo Canvas

O Modelo *Canvas*, é uma ferramenta utilizada por empreendedores de todo o mundo, que tem como principal objetivo definir a estrutura do negócio. Através desta ferramenta é possível avaliar se a empresa tem asseguradas as principais condições para garantir a viabilidade do seu negócio [47]. De seguida, a Figura 16 apresenta o modelo *Canvas* com a

identificação das principais diretrizes do negócio presentes nos vários blocos do modelo realizados no âmbito deste projeto.

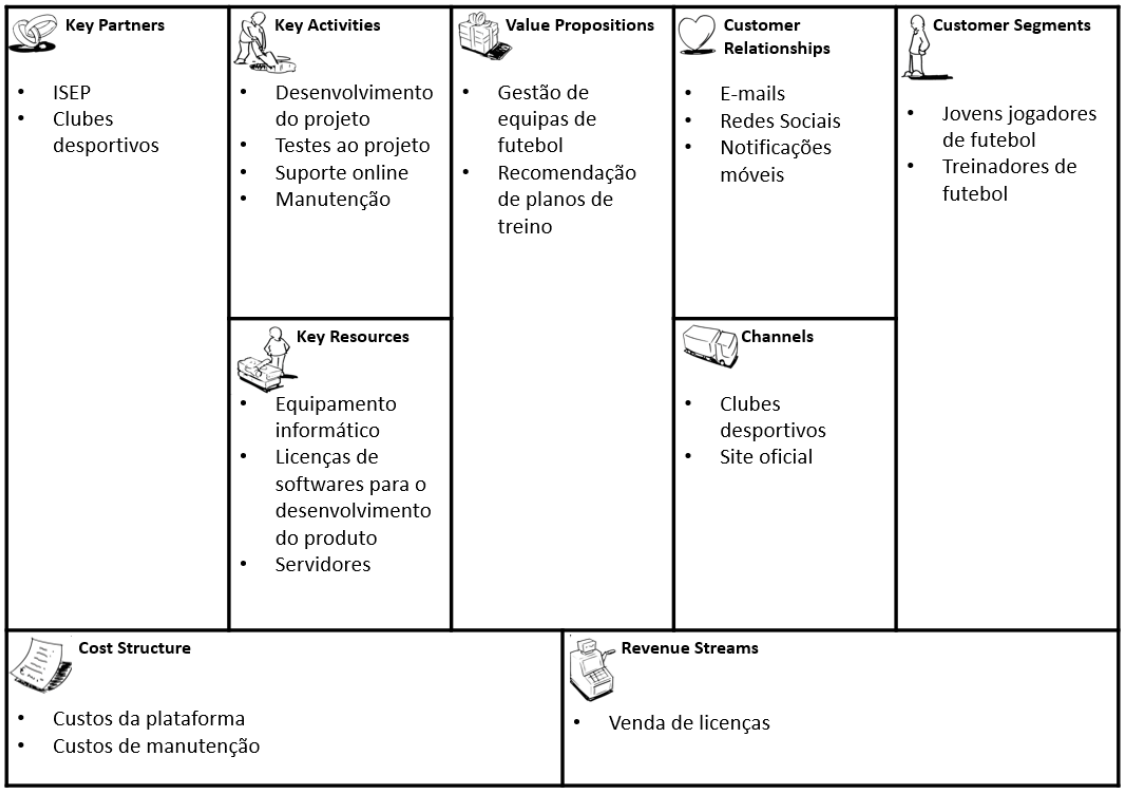


Figura 16 - Modelo *Canvas* para o presente projeto

4 Design da Solução

O presente capítulo tem como objetivo descrever o design realizado para a solução proposta. Desse modo, primeiramente, será detalhado a visão da solução que contempla uma proposta de resolução para o problema descrito.

De seguida, é exposta toda a análise e conceção da solução. Esta, contém a identificação dos *stakeholders*, atores do sistema e a descrição dos requisitos funcionais e não funcionais.

Posteriormente, é apresentada a arquitetura da solução, através do modelo de vistas 4+1, identificando os seguintes componentes: vista lógica, vista de implementação, vista de implantação, vista de dados e vista dos casos de uso. Após isso, são expostas possíveis alternativas que foram estudadas, mas não escolhidas para a arquitetura da solução.

4.1 Visão da Solução

Nesta secção, realiza-se uma análise à visão da solução elaborada. Esta, será baseada na conceção de uma aplicação *web* responsiva independente, capaz de responder de forma positiva ao problema identificado na secção **1.2 - Problema**.

O problema exposto, visa avaliar se a possibilidade da existência de um mecanismo de recomendação irá permitir alcançar uma melhoria das performances dos jogadores, desse modo, é proposto a criação de um sistema de recomendação de treinos para jovens jogadores de futebol. O mesmo, deve ser desenvolvido mediante a técnica baseada em conteúdo, como analisado e justificado na secção, **2.2.7 - Possíveis abordagens para o Sistema de Recomendação**. De seguida, a Figura 17, apresenta a estrutura da proposta sugerida para a resolução do problema.

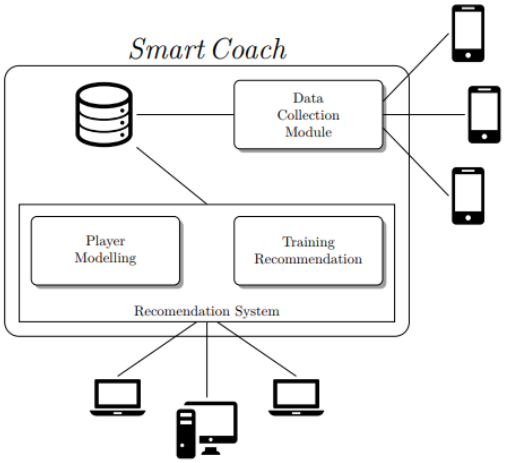


Figura 17 - Diagrama da proposta sugerida [2]

O sistema de recomendação a desenvolver deve ser apoiado através de dados do desempenho dos atletas durante as partidas de futebol anteriormente realizados. Essas informações de desempenho devem ser recolhidas usando uma aplicação *web*, por membros do staff técnico ou por amigos/família associados ao respetivo jogador, como é apresentado no "*Data Collection Module*", presente na Figura 17, posteriormente, esses dados são armazenados numa base de dados para uso futuro por parte do sistema de recomendação [2].

Após isso, através do acesso à informação existente da performance dos jogadores, o sistema de recomendação é executado. Como se pode visualizar na figura anterior, este componente é divido nos módulos, "*Player Module*" e "*Training Recomendation*".

O primeiro, refere-se à modelação dos perfis dos jogadores a analisar. Estes perfis baseiam-se na posição do jogador, assim como, em estereótipos definidos durante entrevistas com vários treinadores de futebol [2]. Os estereótipos, variam entre características físicas (por exemplo, altura, peso, velocidade, altura de salto, etc.) e desempenhos realizados. A Tabela 3 mostra os atributos mais importantes que serão recolhidos para cada jovem jogador durante as partidas. Alguns desses atributos são mais ou menos importantes, dependendo da posição em que o atleta jogou [2].

Tabela 3 - Atributos dos jogadores a recolher por posição [2]

Attribute	Goalkeeper	Centre Back	Full-back/Wing-back	Defensive Midfielder	Centre midfield	Attacking midfield	Winger	Centre forward
(In)Complete saves	●	–	–	–	–	–	–	–
Passing accuracy	●	●	●	●	●	●	●	○
Clearances	○	●	●	●	●	○	○	○
(In)Complete interception	○	●	●	●	○	○	○	○
Ball recovery	–	○	○	●	●	●	○	○
(In)Successful tackles	–	●	●	●	●	●	○	○
Fouls committed	●	●	●	●	●	●	●	●
Fouls suffered	○	○	○	○	●	●	●	●
(In)Successful dribbles	–	○	○	○	○	●	●	●
Duels won/lost	–	●	●	●	●	●	●	●
(In)Successful crosses	–	○	●	○	○	●	●	○
Shots/Shots on target	–	○	○	○	●	●	●	●
Offsides	–	○	●	○	●	●	●	●
Assists	–	●	●	●	●	●	●	●
Goals	–	●	●	●	●	●	●	●

● Major attribute ○ Minor attribute – Not applicable

Já em relação ao "*Training Recomendation*", este módulo descreve o processo de treino de recomendação. O mesmo, tem a responsabilidade de determinar os exercícios de treino para um certo atleta. Para isso, no contexto do sistema *SmartCoach*, as entidades referentes a exercícios de treino terão como parâmetros, elementos associados aos estereótipos definidos do "*User Model*" do respetivo jogador. Isso irá permitir a relação dos atributos de desempenho dos jovens atletas com a possibilidade de sugestão de exercícios de treino específicos às necessidades dos jogadores.

De forma a possibilitar uma solução completa, por parte do módulo "*Training Recomendation*", este projeto será integrado com um projeto paralelo de apoio à decisão de treino de equipas técnicas, realizado pelo estudante Hugo Soares, o projeto em questão, baseia-se num sistema denominado, *SmartCoachManager,* de atribuição, classificação e gestão de exercícios de treino, o que possibilitará a disponibilização de uma API com dados relevantes sobre exercícios de treinos específicos. Podemos concluir que esta integração irá beneficiar o projeto a desenvolver, uma vez que permitirá a aplicação ter acesso a informações de exercícios de treinos existentes e seus detalhes, tais como, nome, descrição, e imagens e vídeos exemplificativos. Mais concretamente, os exercícios partilhados irão estar associados a certos atributos de treino, tais como, finalização, contenção, técnica, etc. Desse modo, após a execução do algoritmo do sistema de recomendação são retornados os atributos necessários a melhorar, e com base nessas características, são obtidos os exercícios de treinos provenientes da API integrada, gerando assim, um plano de treino específico e eficaz para um certo jogador.

Por fim, para uma melhor compreensão da solução elaborada, é exposto todo o processo de recomendação do sistema a desenvolver para um certo jogador, através de um diagrama de processo *BPMN*.

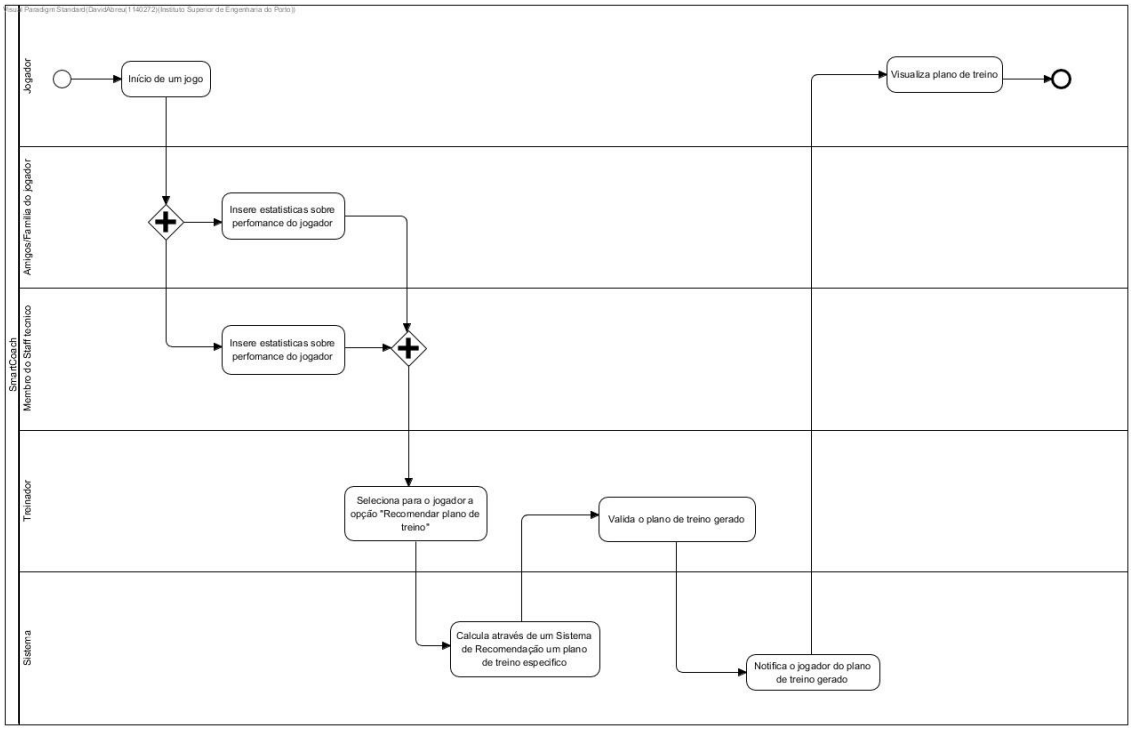


Figura 18 - Diagrama do Processo de Recomendação

Podemos verificar que o processo dá início com o pontapé de saída de um jogo de futebol, durante realização do jogo ou no final do mesmo, são inseridas as estatísticas do jogador para aquele jogo na aplicação, quer por parte de membros do Staff técnico, quer por utilizadores associados ao jogador em questão. Após isso, torna-se possível para o Treinador através da aplicação selecionar a opção “Recomendar plano de Treino”, o que irá levar o Sistema, mais concretamente, o sistema de recomendação, através de um algoritmo baseado em conteúdo recomendar um plano de treino. De seguida, o Treinador deve validar o plano de treino gerado e o sistema deve disponibilizar e notificar o mesmo para o respetivo Jogador.

4.2 Análise e Conceção

Neste capítulo, será apresentado a engenharia de requisitos do projeto a desenvolver, para isso, serão inicialmente identificados os *Stakeholders* e Atores do sistema. De seguida, serão expostos os requisitos funcionais, bem como, os não funcionais. Terminando com a apresentação do modelo de domínio e esclarecimento dos seus conceitos principais.

4.2.1 Stakeholders

Stakeholders ou partes interessadas, são consideradas uma entidade que poderá ser afetada, afetar ou simplesmente ter interesse, de forma direta ou indireta, positiva ou negativamente pelo sistema. Assim sendo, é importante identificar quem são essas entidades, uma vez que estas poderão influenciar o processo de desenvolvimento do sistema.

Desse modo, foram identificados os seguintes *stakeholders*:

- **Equipas de Futebol**: Entidade que pretende utilizar a aplicação desenvolvida internamente.
- **Orientador**: Pessoa encarregue de orientar o estudante na construção e no desenvolvimento do projeto.
- **Estudante**: Entidade responsável pelo desenvolvimento da aplicação.
- **Treinador**: Pretende obter bons resultados desportivamente e gerir o bom funcionamento da aplicação.
- **Staff Técnico**: Pretende ajudar o treinador a gerir a sua equipa.
- **Jogador**: Pretende melhorar as suas performances desportivas.
- **Utilizador Associado**: Pretende aceder a aplicação de forma simples e intuitiva, com vista a ajudar o jogador a melhorar as suas performances.

4.2.2 Atores

Um Ator do sistema é considerado um utilizador que desempenha um papel específico na aplicação. Assim sendo, foram identificados os seguintes Atores do sistema:

- **Utilizador Não Autenticado**: Utilizador do sistema sem conta associada, só terá acesso a visualizar a página principal e às funcionalidades de Registo e Login.
- **Treinador**: É o utilizador com permissões para gerir as várias configurações do sistema, nomeadamente, gestão da sua equipa, incluindo os seus jogadores/membros do Staff/jogos, entre outros. O mesmo também tem acesso a todas a funcionalidades do software.
- **Staff Técnico**: É o utilizador responsável pelo levantamento de dados estatísticos sobre a performance do jogador. O mesmo também poderá auxiliar o treinador na gestão da sua equipa.
- **Jogador**: Utilizador do sistema, que possui acesso de visualização às funcionalidades da aplicação, assim como, a possibilidade de gerar uma recomendação de treinos para si mesmo.
- **Utilizador Associado**: Utilizador associado (normalmente, familiar ou amigo) de um certo jogador, que tem a responsabilidade de inserir estatísticas ao vivo desse jogador, num determinado jogo.

4.2.3 Requisitos funcionais

Os requisitos funcionais são exatamente aquilo que o sistema deve fazer, representando as funcionalidades que o sistema deve possuir [48].

Desse modo, nesta secção, serão enumeradas as funcionalidades identificados ao longo do projeto que, por sua vez, deram origem à criação de casos de uso. Assim, para cada um dos Atores identificados será, primeiramente, apresentado o diagrama de casos de uso, e de seguida, será exposta uma breve descrição dos mesmos.

4.2.3.1 Casos de Uso do Utilizador Não Autenticado

Com base nos requisitos levantados foram identificados os seguintes casos de uso para o utilizador não autenticado:

Figura 19 - Diagrama de Casos de uso do Utilizador não Autenticado

UN01 – Login: O utilizador não autenticado, poderá realizar Login na aplicação, inserindo o seu email e palavra-passe.

UN02 – Registo: Caso o utilizador não autenticado não tenha uma conta no sistema, o mesmo poderá se registar na aplicação inserindo os seus dados pessoais e escolhendo uma Função, (Treinador, Staff Técnico, Jogador ou Utilizador Associado). Por fim, caso já exista a sua equipa no sistema, o utilizador poderá inserir o código da equipa, enviando assim, um pedido para se juntar à mesma, no entanto, caso o utilizador não queira fazer a associação de imediato, poderá deixar este procedimento para mais tarde.

4.2.3.2 Casos de Uso do Treinador

Com base nos requisitos levantados foram identificados os seguintes casos de uso para o treinador:

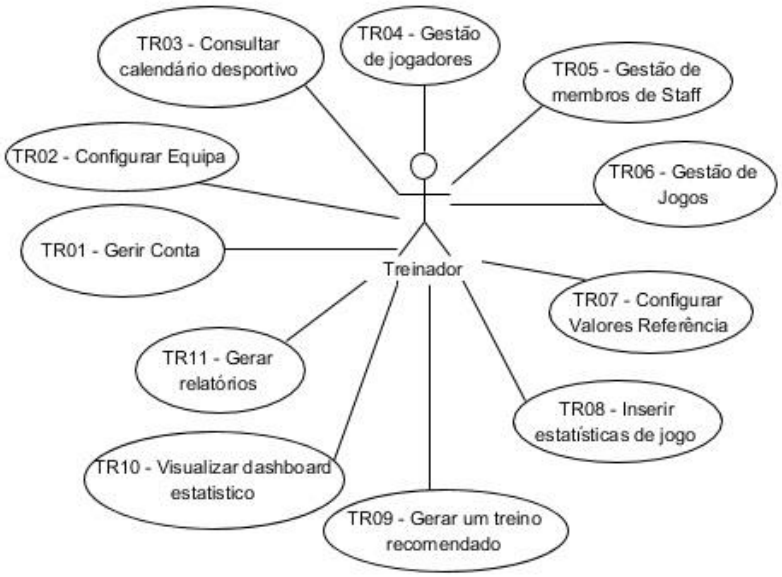


Figura 20 - Diagrama de casos de uso do Treinador

TR01 – Gerir Conta: O Treinador a qualquer momento poderá atualizar os seus dados pessoais, assim como, ter acesso às funcionalidades de Abandonar Equipa e/ou Apagar Conta.

TR02 – Configurar Equipa: O Treinador ao selecionar a opção "Configurar Equipa", nas definições da aplicação, deverá indicar os dados para a criação/atualização da sua equipa, os dados possíveis a inserir são: o nome, a abreviatura, o escalão, a descrição e localização da equipa.

TR03 – Consultar calendário desportivo: O Treinador poderá consultar calendário da época desportiva, contendo os próximos jogos/treinos, assim como, os jogos já realizados. Para cada um dos eventos existentes, o treinador poderá visualizar os detalhes do mesmo.

TR04 – Gestão de jogadores: O Treinador ao selecionar a opção "Gestão de Jogadores", deverá puder convidar, remover, ou visualizar os detalhes dos jogadores do plantel da sua equipa.

TR05 – Gestão de membros de Staff: O Treinador ao selecionar a opção "Gestão de membros de Staff", deverá puder convidar, remover, ou visualizar os detalhes dos membros do staff técnico associados à sua equipa.

TR06 – Gestão de jogos: O Treinador ao selecionar a opção "Gestão de Jogos", terá um novo painel aonde deverá indicar a ação que pretende realizar, as opções disponíveis são: visualizar detalhes de jogo, inserir resultado, criar novo jogo, atualizar dados do jogo ou remover jogo.

TR07 – Configurar Valores Referência: O Treinador deverá definir valores referência dos limites para a análise da performance dos jogadores, de modo, a verificar quais atributos o jogador deve focar para melhorar a sua performance. Estes valores são automaticamente gerados aquando do registo do Treinador, no entanto, deve ser possível ao Treinador, modificar/atualizar os mesmos sempre que pretender.

TR08 – Inserir estatísticas de jogo: O Treinador pretende visualizar estatísticas coletivas e individuais referentes a um dado jogador, para isso, deverá selecionar a opção "Visualizar estatísticas" e será apresentado num novo ecrã um conjunto de gráficos que mostram as mais variadas estatísticas dos jogos realizados.

TR09 – Gerar treino recomendado: O Treinador deve puder gerar um treino recomendado, para isso, precisa inicialmente de selecionar um ou mais jogadores, inserir um período de análise, assim como, a data do treino a ser criado. Esta funcionalidade, através de um sistema de recomendação analisa os dados recolhidos do respetivo jogador no período selecionado, e comparando com os valores referência e o histórico de estatísticas recolhidas de outros utilizadores da mesma posição da aplicação recomenda um plano de treino específico consoante as necessidades do atleta.

TR10 – Visualizar *dashboard* estatístico: O Treinador pretende visualizar estatísticas coletivas da sua equipa e/ou individuais de um certo jogador, para isso, deverá selecionar a opção "*Dashboard*", na qual será apresentado num novo ecrã um conjunto de gráficos que mostram as mais variadas estatísticas e indicadores da sua equipa e seus jogadores.

TR11 – Gerar relatórios: O Treinador deverá puder gerar relatórios da performance de um jogador ao selecionar a opção "Gerar Relatório", na página dos detalhes do respetivo jogador, este relatório deve conter toda a informação do jogador, assim como, das estatísticas obtidas nos jogos que participou.

4.2.3.3 Casos de Uso do Staff Técnico

Com base nos requisitos levantados foram identificados os seguintes casos de uso para o membro de Staff Técnico:

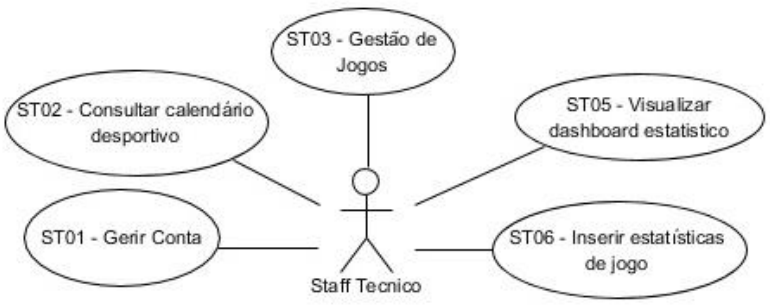


Figura 21 - Diagrama de Casos de uso do membro do Staff Técnico

ST01 – Gerir Conta: O membro do Staff Técnico a qualquer momento poderá atualizar os seus dados pessoais, assim como, ter acesso às funcionalidades de Abandonar Equipa e/ou Apagar Conta.

ST02 – Consultar calendário desportivo: O membro do Staff Técnico poderá consultar calendário da época desportiva, contendo os próximos jogos/treinos, assim como, os jogos já realizados. Para cada um dos eventos existentes, o membro do Staff Técnico poderá visualizar os detalhes do mesmo.

ST03 – Gestão de jogos: O membro do Staff Técnico ao selecionar a opção "Gestão de Jogos", terá um novo painel aonde deverá indicar a ação que pretende realizar, as opções disponíveis são: visualizar detalhes de jogo, inserir resultado, criar novo jogo, atualizar dados do jogo ou remover jogo.

ST04 – Visualizar *dashboard* estatístico: O membro do Staff Técnico pretende visualizar estatísticas coletivas da sua equipa e/ou individuais sobre um certo jogador, para isso, deverá selecionar a opção "*Dashboard*", na qual será apresentado num novo ecrã um conjunto de gráficos que mostram as mais variadas estatísticas e indicadores da sua equipa e seus jogadores.

ST05 – Inserir estatísticas de jogo/treino: O membro do Staff Técnico pretende retirar estatísticas importantes do desempenho dos jogadores nos jogos e treinos realizados, para isso, o Staff técnico, deve selecionar a opção "Inserir dados de jogo" e para um certo jogador, introduzir os dados estatísticos referentes à sua performance, como por exemplo, número de passes curtos realizados com sucesso ou número de remates enquadrados com a baliza, entre outros.

4.2.3.4 Casos de Uso do Jogador

Com base nos requisitos levantados foram identificados os seguintes casos de uso para o jogador:

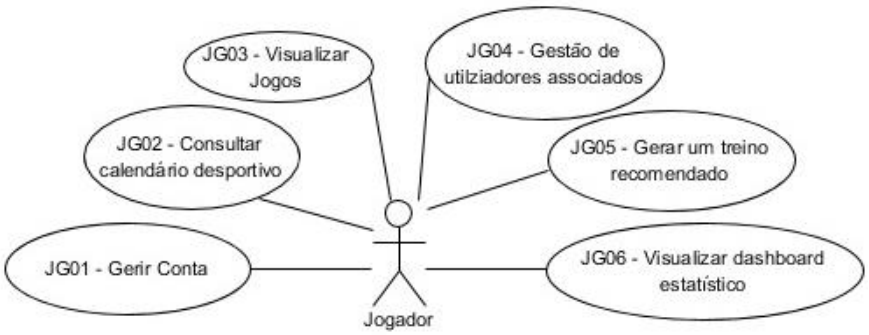


Figura 22 - Diagrama de Casos de uso do Jogador

JG01 – Gerir Conta: O Jogador a qualquer momento poderá atualizar os seus dados pessoais, assim como, ter acesso às funcionalidades de Abandonar Equipa e/ou Apagar Conta.

JG02 – Consultar calendário desportivo: O Jogador poderá consultar calendário da época desportiva, contendo os próximos jogos/treinos, assim como, os jogos já realizados. Para cada um dos eventos existentes, o Jogador poderá visualizar os detalhes do mesmo.

JG03 – Visualizar jogos: O Jogador ao selecionar um jogo será exibido os detalhes do mesmo, no caso de um jogo já realizado, o jogador, visualiza os dados do jogo (equipa visitante, equipa visitada, data, hora, local e convocados), o resultado final e outras ocorrências, bem como, a possibilidade de ver as várias estatísticas recolhidas do jogo, no caso do jogo estar agendado, o jogador tem apenas acesso aos dados do jogo (equipa visitante, equipa visitada, data, hora, local e convocados).

JG04 – Gestão de Utilizadores Associados: O Jogador deverá puder atualizar a lista de utilizadores associados à sua conta, desse modo, poderá remover utilizadores atualmente associados, convidar novos utilizadores e aceitar/rejeitar convites de utilizadores para serem seus associados.

JG06 – Gerar um plano de treino recomendado: O Jogador deve puder gerar um treino recomendado, para isso, precisa de inserir um período de análise, assim como, a data do treino a ser criado. Esta funcionalidade, através de um sistema de recomendação analisa os dados recolhidos do respetivo jogador no período selecionado, e comparando com os valores referência e o histórico de estatísticas recolhidas de outros utilizadores da mesma posição da aplicação recomenda um plano de treino específico consoante as necessidades do atleta.

JG05 – Visualizar *dashboard* estatístico: O Jogador pretende visualizar estatísticas coletivas da sua equipa e/ou individuais sobre o seu desempenho, para isso, deverá selecionar a opção "*Dashboard*". Esta funcionalidade, irá apresentar num novo ecrã um conjunto de gráficos que mostram as mais variadas estatísticas e indicadores da sua equipa e sobre o seu desempenho.

4.2.3.5 Casos de Uso do Utilizador Associado

Com base nos requisitos levantados foram identificados os seguintes casos de uso para o utilizador associado:

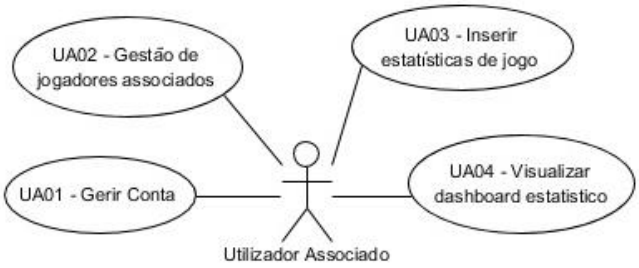


Figura 23 - Diagrama de Casos de uso do Utilizador Associado

UA01 – Gerir Conta: O Utilizador Associado a qualquer momento poderá atualizar os seus dados pessoais, assim como, ter acesso à funcionalidade de Apagar Conta.

UA02 – Gestão de Jogadores Associados: O Utilizador Associado deverá puder atualizar a lista de jogadores associados à sua conta, desse modo, poderá remover jogadores atualmente associados, convidar novos jogadores e aceitar/rejeitar convites de jogadores para serem seus associados.

UA03 – Inserir estatísticas de jogo: O Utilizador Associado pretende recolher estatísticas importantes do desempenho do jogador nos jogos realizados, para isso, o utilizador, deve selecionar a opção "Inserir dados de jogo" e consoante a escolha de um dos seus jogadores associados, poderá de seguida introduzir os dados estatísticos referentes à sua performance, como por exemplo, número de passes curtos realizados com sucesso ou número de remates enquadrados com a baliza, entre outros.

UA04 – Visualizar *dashboard* estatístico: O Utilizador Associado pretende visualizar estatísticas individuais sobre o desempenho dos jogadores que segue, para isso, deverá selecionar a opção "*Dashboard*". Esta funcionalidade, irá apresentar num novo ecrã um conjunto de gráficos que mostram as mais variadas estatísticas e indicadores sobre o desempenho dos seus jogadores.

4.2.4 Requisitos não funcionais

Os requisitos não funcionais, são requisitos que especificam como o sistema se deve comportar. Os mesmos, podem ser considerados como atributos do sistema. [48]

Para definir os requisitos não funcionais, é utilizado o modelo FURPS+ (acrónimo de *Functionality*, *Usability*, *Reliability*, *Performance* e *Supportability* e + para categorias adicionais), este modelo, permite a fácil classificação de requisitos não abrangidos pelos requisitos funcionais. De seguida, é apresentada a descrição de cada uma das entidades presentes no acrónimo:

- ***Functionality*** (Funcionalidade): Especifica as funcionalidades que não se relacionam com os casos de uso, nomeadamente: auditoria, interoperabilidade e segurança [49].
- ***Usability*** (Usabilidade): Avalia características relacionadas com a estética e consistência da interface com o utilizador [49].
- ***Reliability*** (Confiabilidade): Refere-se à integridade, conformidade e disponibilidade do software [49].
- ***Performance*** (Desempenho): Avalia os requisitos de desempenho do sistema, nomeadamente: tempo de resposta, tempo de recuperação, tempo de inicialização e tempo de encerramento [49].
- ***Supportability*** (Suportabilidade): Os requisitos relacionados com características, como: testabilidade, adaptabilidade, manutenibilidade, compatibilidade, configurabilidade, instabilidade, escalabilidade e localização [49].
- **+:** Utilizado para categorias adicionais como requisitos de design, requisitos de implementação, requisitos de interface e requisitos físicos [49].

Assim sendo, tendo em conta as entidades presentes no Modelo FURPS+, serão apresentados os principais requisitos não funcionais identificados para o sistema a desenvolver:

1. **Funcionalidades**
   - Autenticação dos utilizadores para aceder às funcionalidades da aplicação.
   - Autorização dos acessos a certas funcionalidades, tendo em conta o seu Role na aplicação.
   - Segurança dos dados de cada utilizador da aplicação, como as suas credencias de acesso, os seus dados pessoais, entre outros.

2. **Usabilidade**
   - Pretende-se que o sistema seja ágil, fácil de usar e completo em termos de informação.

3. **Confiabilidade**
   - A aplicação deve ter uma taxa de erro muito baixa.

4. **Desempenho**
   - A especificação funcional deve ter em consideração o estado-da-arte bem como os requisitos que irão ser levantados.

5. **Suportabilidade**
   - A aplicação deve suportar os seguintes navegadores *web*: Internet Explorer, Google Chrome, Firefox, Opera.

6. **Restrições de Design**
   - Adoção do processo de desenvolvimento de software iterativo e incremental;
   - Adoção de boas práticas de design, nomeadamente padrões GRASP.

7. **Restrições de Implementação**
   - Adoção de normas de codificação;
   - A aplicação *web* deve ser responsiva e acessível através das mais variadas plataformas (e.g. PC, Laptop, Tablet, Smartphones).

8. **Restrições de Interface**
   - Interfaces simples e intuitivas.

4.2.5 Modelo de Domínio

O modelo de domínio é considerado uma representação visual de classes conceituais, ou objetos do mundo real para um determinado problema [50]. De seguida, na Figura 24, irá ser apresentado o modelo de domínio desenvolvido para a análise do problema proposto, o mesmo inclui os principais conceitos e seus atributos, assim como as relações entre si. Para uma melhor interpretação deste artefacto, serão posteriormente, esclarecidos os principais conceitos presentes no diagrama.

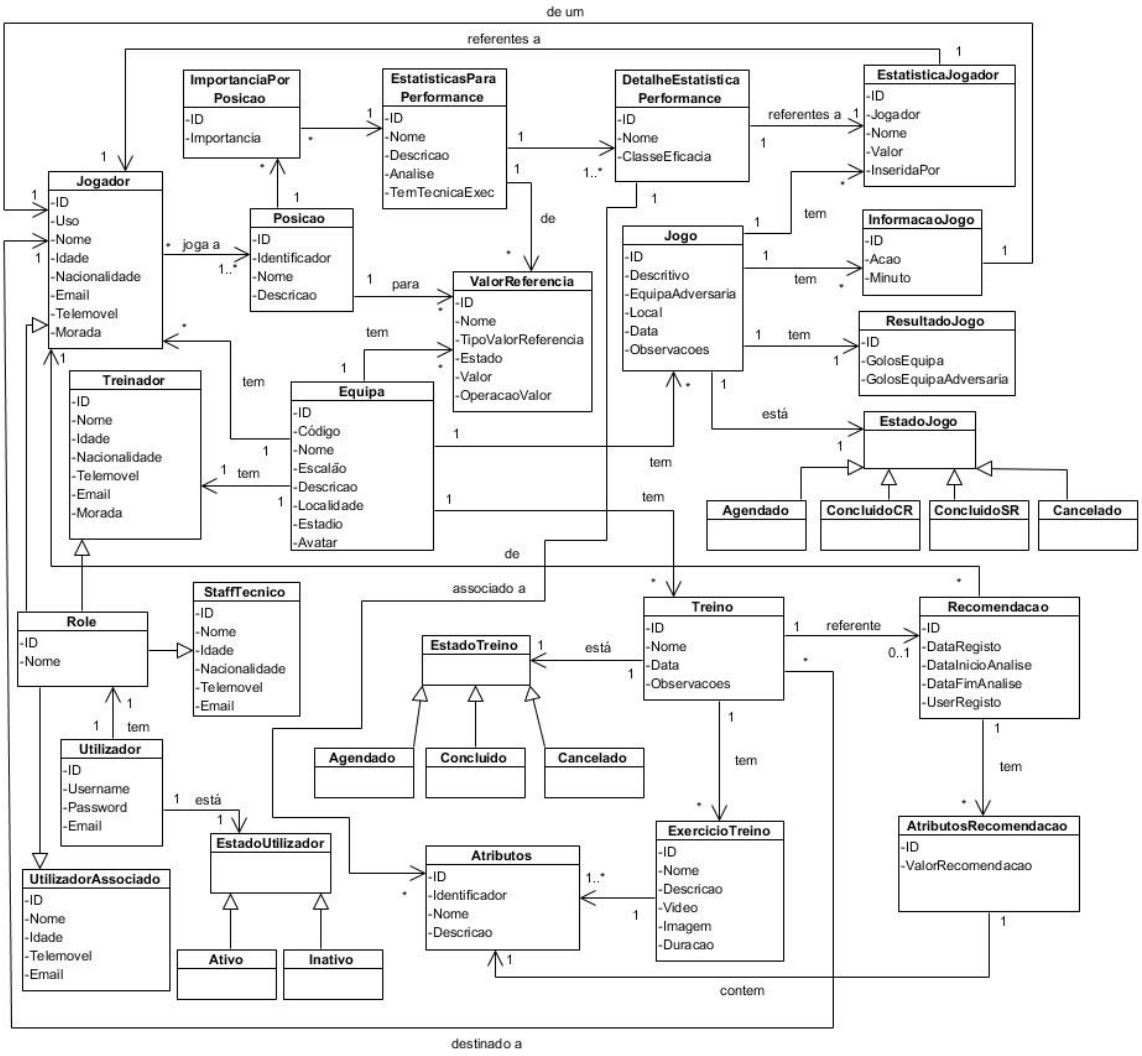


Figura 24 - Modelo de Domínio

4.2.6 Conceitos

Esta secção tem o objetivo de descrever os principais conceitos inerentes no modelo de domínio identificado, assim, primeiramente, será descrito o conceito de Utilizador identificando os três tipos de utilizador existente, de seguida, é apresentado o conceito de Equipa, de Jogo, de Treino, e por último, de Valor Referência.

4.2.6.1 Utilizador

O conceito Utilizador, como o nome indica é referente aos usuários da aplicação, na qual se pode distinguir três tipos diferentes de utilizadores, com diferentes permissões:

- Treinador (*TR*): É o utilizador principal da aplicação, e pode ser considerado o administrador da aplicação, pois tem permissões para gerir as várias configurações do sistema, nomeadamente, gestão da equipa, gestão dos jogadores, gestão dos jogos, entre outros.

- Staff Técnico (*ST*): É o utilizador com permissões para auxiliar o Treinador, este tem acesso a algumas configurações da equipa, assim como, a possibilidade de inserção de dados relativos a estatísticas dos jogos.
- Jogador (*JG*): É o utilizador mais comum do sistema, o mesmo, tem por norma, permissões de acesso a visualização de dados e também a possibilidade de gerar um plano de treino.
- Utilizador Associado (*UA*): É o utilizador associado a um ou mais jogadores e deverá ter permissões de acesso à inserção de estatísticas de jogos e a visualização de um *dashboard* estatístico.

Qualquer utilizador da aplicação terá um *username*, uma password, um nome, uma idade e um email associado (dados obrigatórios), no qual é identificado pelo seu *username*. Para verificação da existência de um utilizador é usado o seu estado, que pode variar entre Ativo e Inativo.

4.2.6.2 Equipa

Entidade principal da aplicação, na mesma, está definida o nome, o escalão, a descrição e a localização da equipa. Esta entidade, tem ainda associado a lista de jogadores, membros de staff e o treinador, assim como, a lista dos jogos, treinos e uma tabela de valores referência. É ainda importante salientar, que a Equipa tem um código único, que permitirá a realização de convites de adesão por parte dos jogadores e membros do staff técnico.

4.2.6.3 Jogo

Esta entidade, como referida no conceito acima, está associada a uma equipa, a mesma, diz respeito a uma partida de futebol realizada ou a realizar, e tem os atributos descritivos referentes às equipas que se confrontam, assim como, uma data e possíveis observações. Caso o Jogo já tenha sido realizado, este conceito poderá estar relacionado com dados estatísticos provenientes do mesmo.

Para concluir, nesta entidade, está presente o atributo estado do Jogo, este poderá variar entre quatro valores possíveis (Agendado, Concluído Sem Resultado, Concluído Com Resultado ou Cancelado), a Figura 25, apresenta o diagrama de atividade dos estados que o Jogo pode passar ao longo do tempo.

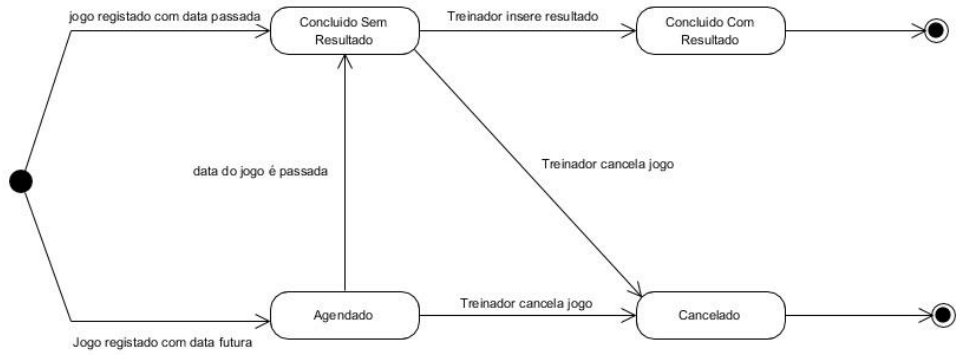


Figura 25 - Diagrama dos possíveis Estados da entidade Jogo

4.2.6.4 Treino

Esta entidade, semelhante ao conceito Jogo, está associada a uma equipa, e diz respeito a um treino agendado ou já realizado. O Treino tem os atributos nome, data e possíveis observações. É destinado a todos utilizadores existentes na equipa, e pode ter programado um ou mais planos de treino específicos para determinados jogadores.

Por fim, a entidade Treino, diz respeito a um conjunto de exercícios de treino, recomendados pelo sistema de recomendação, destinados a um jogador.

4.2.6.5 Valor Referência

O conceito Valor Referência, está relacionado com os valores limites para a análise da performance dos jogadores, por via do sistema de recomendação, estes valores são automaticamente gerados, assim que uma equipa é criada, e podem ser posteriormente atualizados por parte do treinador.

4.2.6.6 Estatística para Performance

Esta entidade, diz respeito ao conjunto de estatísticas a levantar durante um jogo, a mesma é definida pelo seu nome, descrição e uma enumeração com o tipo de análise que pode variar entre:

- Simples – Destinada a estatísticas que só importa o número de ocorrências, por exemplo: número de cortes realizados;
- Eficácia – Destinada a estatísticas que seja necessário obter a taxa do sucesso, por exemplo: (%) passes curtos com sucesso;
- Tipo – Destinada a estatísticas que tenham detalhes do tipo de execução, por exemplo: golos marcados (remate, cabeça, bola parada);

Por fim, pode-se dizer que é uma entidade geral (para todas as equipas) e fixa, ou seja, não poderão ser adicionadas novas instâncias ou atualizadas as existentes.

4.3 Arquitetura da Solução

Nesta secção será exposta a arquitetura da solução desenvolvida, como referido anteriormente, para descrever o software com a maior precisão possível, a estrutura utilizada é baseada no modelo de arquitetura "4 + 1", como é ilustrado na Figura 26

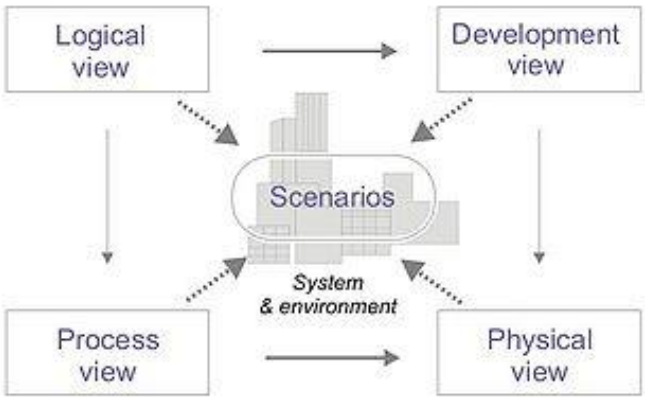


Figura 26 - Modelo de arquitetura 4 + 1 [51]

Contudo, de maneira a otimizar a arquitetura proposta pelo modelo acima, é apresentado uma nova vista referente à persistência de dados (vista de Dados), em substituição da vista de Processos, uma vez que neste projeto, a mesma, é incluída na vista de casos de uso (cenários). Assim, as vistas utilizadas para documentar o presente sistema são:

- **Vista Lógica**: Descreve todos os componentes do sistema.
- **Vista de Implantação**: Descreve onde os componentes do sistema estão instalados.
- **Vista de Implementação**: Descreve as camadas do sistema.
- **Vista de Dados**: Descreve os elementos persistentes arquitecturalmente significativos no modelo de dados.
- **Vista de Cenários**: Descreve a relação entre os atores e os casos de uso do sistema.

4.3.1 Vista Lógica

A vista lógica identifica os principais subsistemas e packages aplicando uma separação e distribuição de responsabilidades. Nas próximas secções é mostrado, inicialmente, uma vista de alto nível e, de seguida, uma vista para cada um dos principais componentes presentes nessa vista de alto-nível, descrevendo as entidades presentes.

4.3.1.1 Arquitetura Geral

Para a arquitetura da solução, foi tomada em conta duas possíveis abordagens:

1. Arquitetura monolítica, baseada no desenvolvimento de uma API;
2. Arquitetura baseada em micro serviços;

Após uma análise cuidada, foi concluído que a solução a adotar seria a primeira, ou seja, baseada no desenvolvimento de uma API. A secção **4.4.1 - Vista Lógica Alternativa**, apresenta a alternativa considerada, bem como, a justificação da escolha da primeira abordagem, tendo em conta, um estudo dos prós e contras de cada uma delas.

Por outro lado, como referido na secção **4.1 - Visão da Solução**, o desenvolvimento do presente sistema será integrado com uma API de um projeto paralelo de apoio à decisão de treino de equipas técnicas, logo, de modo a apresentar a solução arquitetural escolhida e a integração com a API externa, será exposta, de seguida, através de um diagrama de componentes a arquitetura de alto nível do Sistema *SmartCoach*, presente na Figura 27.

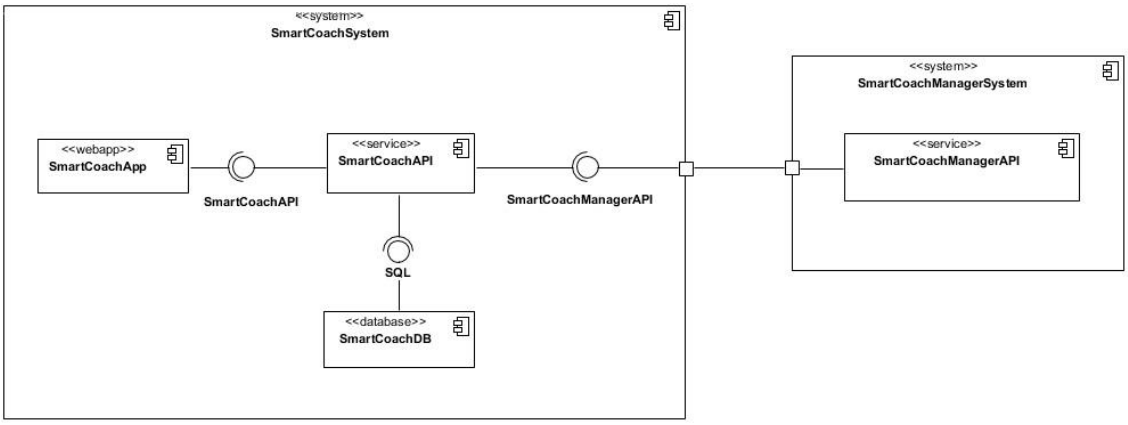


Figura 27 - Diagrama de componentes da Arquitetura geral

- ***SmartCoachSystem*:** Componente principal com o estereótipo "*system*", representa o sistema como um todo;
- ***SmartCoachAPP*:** Aplicação cliente que corre num *web* browser e está disponível para ser acedida a qualquer momento por qualquer utilizador, através de um sistema desktop ou móvel (daí o estereótipo "*webapp*"). Para ter acesso aos dados, este componente comunica com a *SmartCoachAPI* através de ligações *HTTPS*, que será descrita abaixo;
- ***SmartCoachAPI*:** 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 (daí o estereótipo "*service*");
- ***SmartCoachDatabase*:** Componente responsável por guardar os dados referente ao domínio implementado.
- ***SmartCoachManagerSystem*:** Sistema independente que é composto pela *SmartCoachManagerAPI* e sua base de dados.
- ***SmartCoachManagerAPI*:** Componente responsável por disponibilizar serviços relacionados com os exercícios de treino.

4.3.1.2 Arquitetura *Backend*

De seguida, na Figura 28, é retratada a estruturação interna do componente *SmartCoachAPI*, este componente, representa a arquitetura *backend* do projeto.

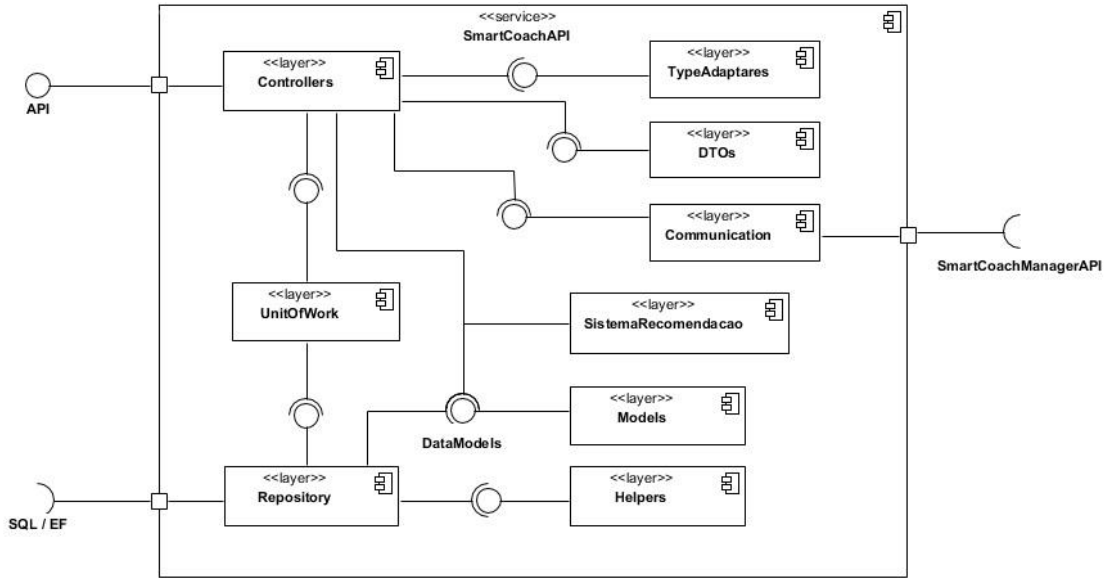


Figura 28 - Diagrama de componentes da Arquitetura Backend

Antes da explicação de cada entidade presente no diagrama anterior, é importante definir que o estereótipo "*layer*" presente nos componentes da figura anterior, representa uma camada na API. De seguida, são então explicadas as entidades presentes na Figura 28:

- ***DTOs***: Usado para transferir dados entre a API e a aplicação agrupando um conjunto de valores (atributos) numa classe simples de forma a otimizar a comunicação. Não apresenta lógica de negócio;
- ***Controller***: Responsável por tratar dos pedidos à API. Usa DTOs para a receção e envio de dados e utiliza o UnitOfWork para a obtenção de dados provenientes da base dados;
- ***UnitOfWork***: Responsável por manter uma lista de objetos afetados por uma transação e coordenar a escrita de mudanças, tratando, caso seja necessário, de possíveis problemas de concorrência;
- ***Repository***: 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;
- ***Model***: Responsável por representar as classes de domínio e lógica de negócio sempre que necessário;
- ***Helpers***: Responsável por possuir classes que sustentam e complementam o bom funcionamento da aplicação, como o *SmartCoachInitializer*, o *EmailService*, entre outros;

- ***SistemaRecomendação***: Responsável por o mecanismo de recomendação e suas classes modelos, que tratam da componente de recomendação do software;
- ***TypeAdapters***: Responsável por possuir a lógica necessária para a conversão entre objetos *DTOs* e *Models*, assim como, a operação inversa;
- ***Communication***: Responsável por tratar da comunicação com serviços externos, como o *SmartCoachManager*, retratado na Arquitetura Geral.

4.3.1.3 Arquitetura *Frontend*

Por fim, para a arquitetura *Frontend* foi utilizado o padrão MVC para o componente *SmartCoachApp*, com a seguinte estrutura:

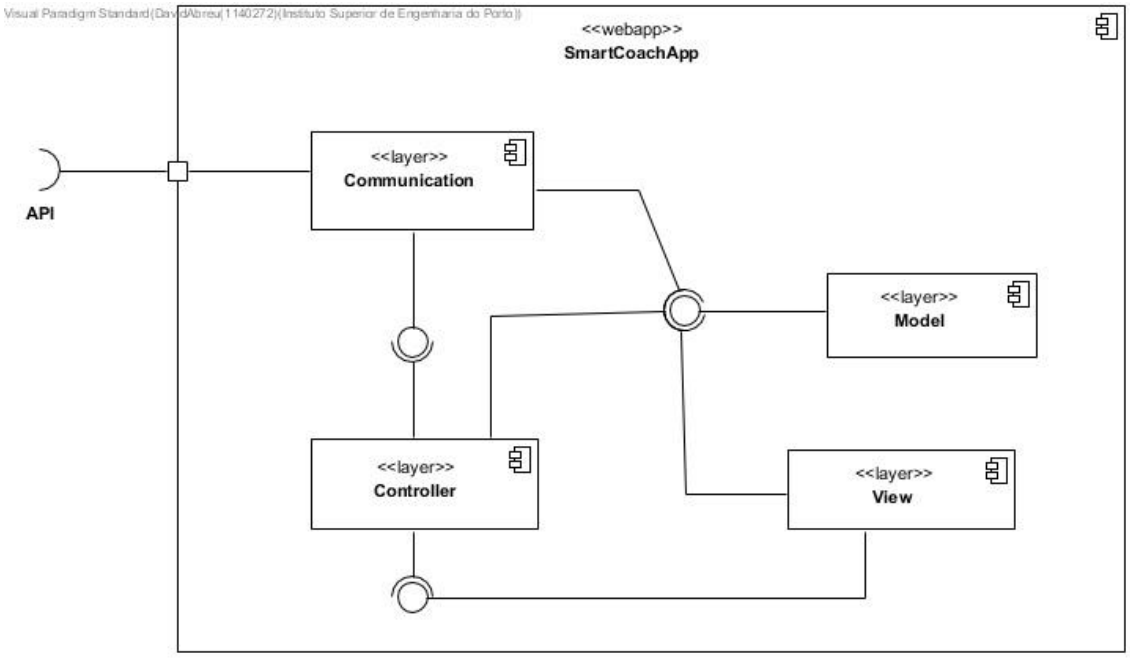


Figura 29 - Diagrama de componentes da Arquitetura Frontend

Os componentes presentes na figura anterior são:

- ***Controller***: Responsável por possuir métodos de ação que lidam com solicitações do browser, recuperar dados necessários do *model* e retornar as respostas apropriadas.
- ***Model***: Responsável por responder a pedidos para devolver informação sobre o estado da informação (usualmente vindos da *view*) e responder a instruções para 5 mudar o estado da informação (usualmente vindos do *controller*).
- ***View***: Responsável por gerir a apresentação das páginas 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.

4.3.2 Vista de Implantação

Deployment ou implantação é o processo de disponibilizar num ambiente específico o software que está a ser desenvolvido para os seus *stakeholders* [52]. De forma a facilitar este processo, foi elaborado um diagrama de implantação, aonde são representados os nós que compõe o sistema, os seus componentes, e as relações entre eles. De seguida, na Figura 30 pode ser consultado o respetivo diagrama.

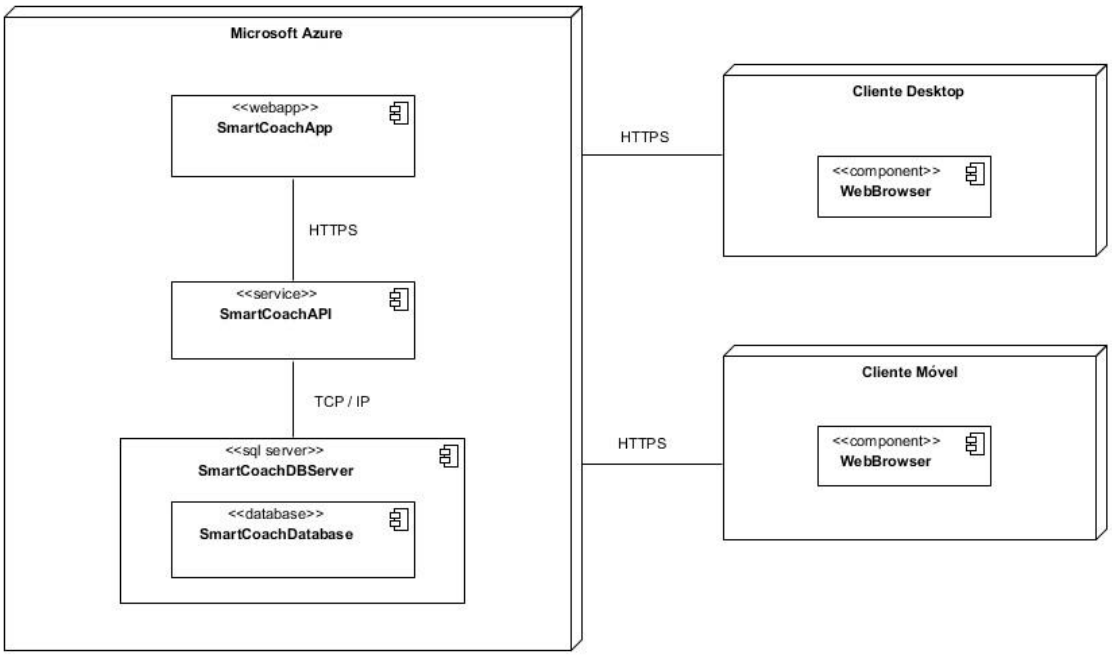


Figura 30 - Diagrama de Implantação do Sistema

Como se pode verificar na imagem acima, foi adotado o método de implantar o software no *Microsoft Azure*. Foi seguida esta abordagem, muito devido à sua simplicidade no processo de implantação, tanto para a aplicação *web*, como para a aplicação servidora e sua base de dados.

Para uma explicação mais detalhada, irá ser elaborada a descrição para cada um dos três principais nós presentes no diagrama.

- ***Microsoft Azure***: Servidor do *Azure,* que irá conter implantados todos os componentes do sistema. Desse modo, a *SmartCoachApp*, irá comunicar por via HTTPS com a *SmartCoachAPI,* que por sua vez, irá transferir os dados através de uma ligação TCP/IP para um Servidor SQL onde se encontra a base de dados *SmartCoachDatabase*, que é usada para persistir os dados necessários para o bom funcionamento do projeto.
- **Cliente Desktop**: Cliente que acede à aplicação desenvolvida por via de um navegador de internet através de um sistema desktop.
- **Cliente Móvel**: Cliente que acede à aplicação desenvolvida por via de um navegador de internet através de um sistema móvel.

4.3.3 Vista de Implementação

A vista de implementação disponibiliza a estrutura de implementação de um dado projeto através da apresentação dos packages existentes para cada um dos componentes identificados na vista lógica de alto nível (secção **4.3.1.1 - Arquitetura Geral**). Cada package está associado a uma camada. De seguida, na Figura 31, está disponível a estrutura adotada para o projeto a desenvolver.

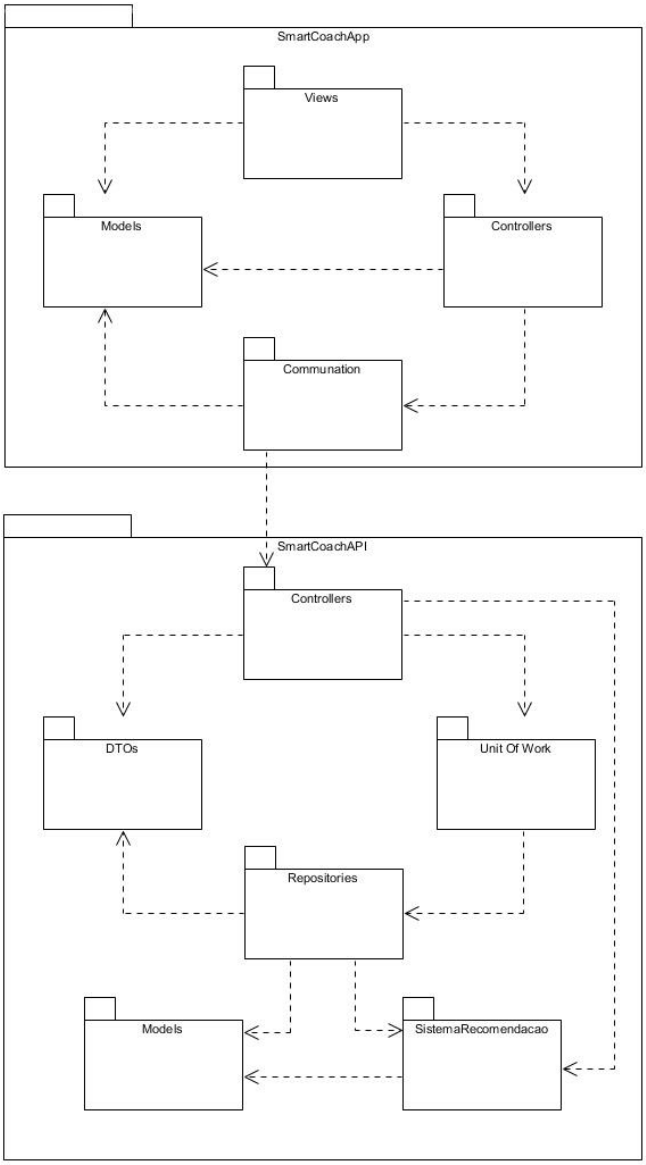


Figura 31 - Vista de Implementação do Sistema

Como podemos visualizar na Figura 31, para o componente *FrontEnd* - ***SmartCoachApp***, estão presentes os seguintes packages:

- ***Views***: Este package insere-se na camada de apresentação e contém as páginas HTML (*Views*) que são apresentadas no browser;

- ***Models***: Este package insere-se na camada de negócio e irá ser responsável pelas classes modelo que incluem as regras de negócio do projeto a desenvolver;
- ***Controllers***: Este package insere-se na camada de controladores e possui as classes *controller* destinados a servirem de intermediários entre as ações do utilizador e os serviços a disponibilizar;
- ***Communication***: Este package insere-se na camada de comunicação e disponibiliza as classes responsáveis por comunicar com a *SmartCoachAPI*.

Já em relação ao componente *BackEnd* - ***SmartCoachAPI***, são identificados os packages subsequentes:

- ***Models***: Este package insere-se na camada de negócio e irá conter as classes modelo que incluem as regras de negócio do projeto a desenvolver;
- ***DTOs***: Este package irá conter as classes *DTOs*, que são responsáveis disponibilizar serviços de transferência de dados de forma eficiente;
- ***Controllers***: Este package insere-se na camada de controladores e possui as classes *controllers* destinados a servirem de intermediários entre as ações do *SmartCoachApp* e os serviços a disponibilizar;
- ***Repositories***: Este package insere-se na camada de acesso aos dados e disponibiliza as classes *repositories*, responsáveis por comunicar com a *SmartCoachDatabase*;
- ***UnitOfWork***: Este package insere-se na camada de acesso aos dados e disponibiliza a classe *UnitOfWork* que contém as instâncias das classes *repositories*, responsáveis por comunicar com a *SmartCoachDatabase*;
- ***SistemaRecomendacao***: Este package insere-se na camada de recomendação e possui as classes modelo da componente de recomendação, assim como, uma classe "*core*", com o processo de recomendação.

4.3.4 Vista de Dados

A vista de Dados, apresenta a persistência de dados do sistema, a mesma, baseia-se na produção de um modelo de dados. Este modelo é considerado uma ferramenta que permite demonstrar como é construída a estrutura lógica de dados na camada de dados. Esta estrutura terá o objetivo de dar suporte ao processo de negócio, através da identificação das relações e restrições entre as várias entidades existentes nesta camada [53]. De seguida, na Figura 32, é possível visualizar as tabelas, assim como, as relações existentes na Base de Dados, a estrutura definida encontra-se normalizada, permitindo assim, assegurar o correto armazenamento dos dados da aplicação.

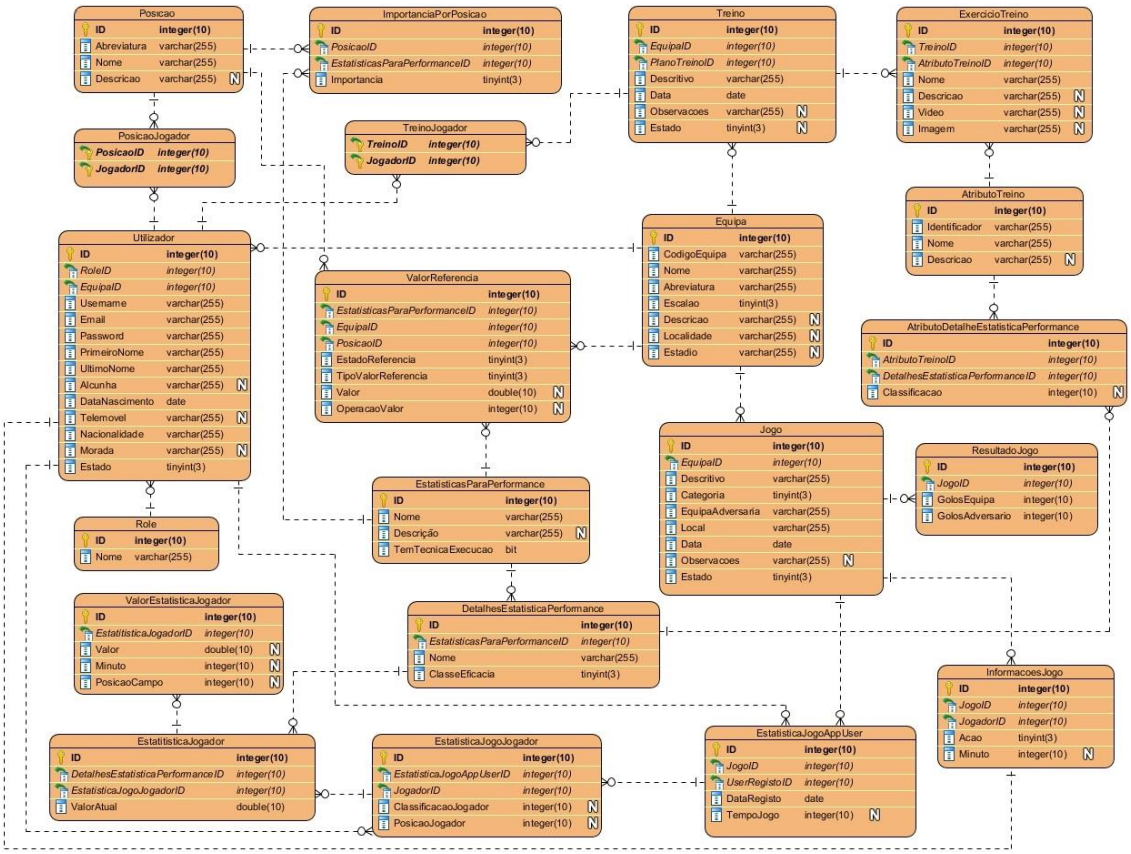


Figura 32 - Modelo de Dados

Em relação ao tipo de dados utilizados, pode-se concluir que os Identificadores/*IDs* das Tabelas apresentadas (*PrimaryKeys* e *ForeignKeys*) são todos do tipo *Integer*, por outro lado, para distinguir as enumerações existentes no projeto foram atribuídos o tipo de dados *TinyInt*, por último, para permitir a persistência de dados textuais, foi atribuído o tipo *Varchar.*

4.3.5 Vista de Cenários Arquiteturalmente Relevantes

A Vista de Cenários, é o que liga os elementos das outras quatro vistas, uma vez que o desenvolvimento da mesma, tem dependência das escolhas arquiteturais realizadas. Esta vista baseia-se em sequências de interações entre os objetos e processos, sendo uma abstração dos requisitos mais importantes [54].

De modo a identificar os casos de uso arquiteturalmente relevantes, foi definido que são aqueles que "exercitam" as partes mais críticas da arquitetura do sistema e demonstram a funcionalidade central do sistema a desenvolver. Assim, na Figura 33 estão representados os casos de uso que apresentam as características mais importantes/relevantes a nível arquitetural.

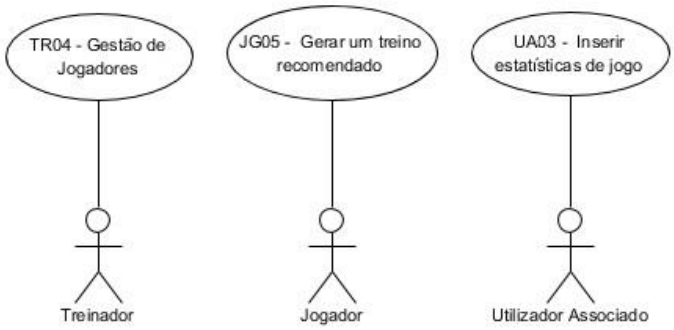


Figura 33 - Diagrama de casos de usos arquiteturalmente relevantes

De seguida, para cada um dos casos de uso representados na figura anterior é descrito com recurso a diagramas de sequência esta funcionalidade, explicando ainda, o porquê da sua escolha.

4.3.5.1 TR04 – Gestão de Jogadores

Este caso de uso foi escolhido por possuir duas operações fundamentais na gestão da equipa, nomeadamente, convidar e remover jogadores, para além disso, este caso de uso, apresenta ligações entre todos os componentes do sistema (aplicação, API e base de dados), assim como, uma conexão com o *EmailService*. Através deste caso de uso, o treinador terá acesso à lista dos jogadores associados à sua equipa, aonde poderá remover jogadores, ou convidar novos via inserção dos emails dos jogadores. Na Figura 34, mediante um diagrama de sequência está retratado todo o processo do respetivo caso de uso.

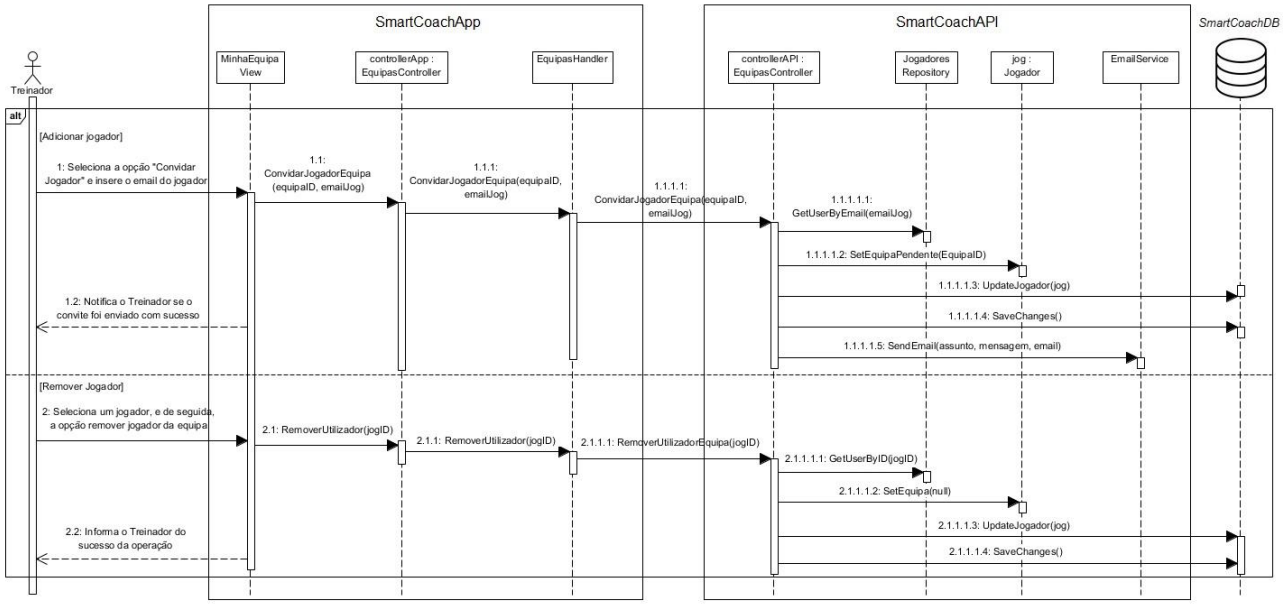


Figura 34 - Diagrama de Sequência do UC Gestão de jogadores

No diagrama da Figura 34, é possível verificar que o treinador ao selecionar a sua equipa, poderá gerir os seus jogadores através de duas operações, a primeira, destinada a adicionar um novo jogador, através da inserção do email do jogador desejado, que através de uma chamada à *SmartCoachAPI*, verifica se o jogador existe no sistema, e em caso afirmativo, adiciona a equipa do treinador, como equipa pendente do jogador, na qual o mesmo, poderá aceitar ou rejeitar a adesão, após isso, a API, comunica com o *EmailService*, que tem como objetivo o envio de um e-mail, com as informações da operação, para o jogador desejado. Por outro lado, para a remoção de um jogador da equipa, o treinador, só necessita de selecionar a opção "Remover Jogador", no jogador que pretende remover, esta ação, através de uma chamada à API, modifica a equipa do jogador selecionado para "*Null"* e atualiza os dados do respetivo jogador na Base de Dados.

4.3.5.2 JG06 – Gerar um plano de treino recomendado

Este caso de uso foi escolhido por se tratar da principal funcionalidade do sistema, e também por possuir ligações entre os vários componentes do sistema, assim como, uma ligação com um serviço externo (*SmartCoachManagerAPI*) para obter os dados de exercícios de treino. Através deste UC o respetivo jogador poderá gerar um plano de treino, tendo em conta as suas performances desportivas. De seguida, na Figura 35, é possível visualizar o diagrama de sequência do respetivo caso de uso.

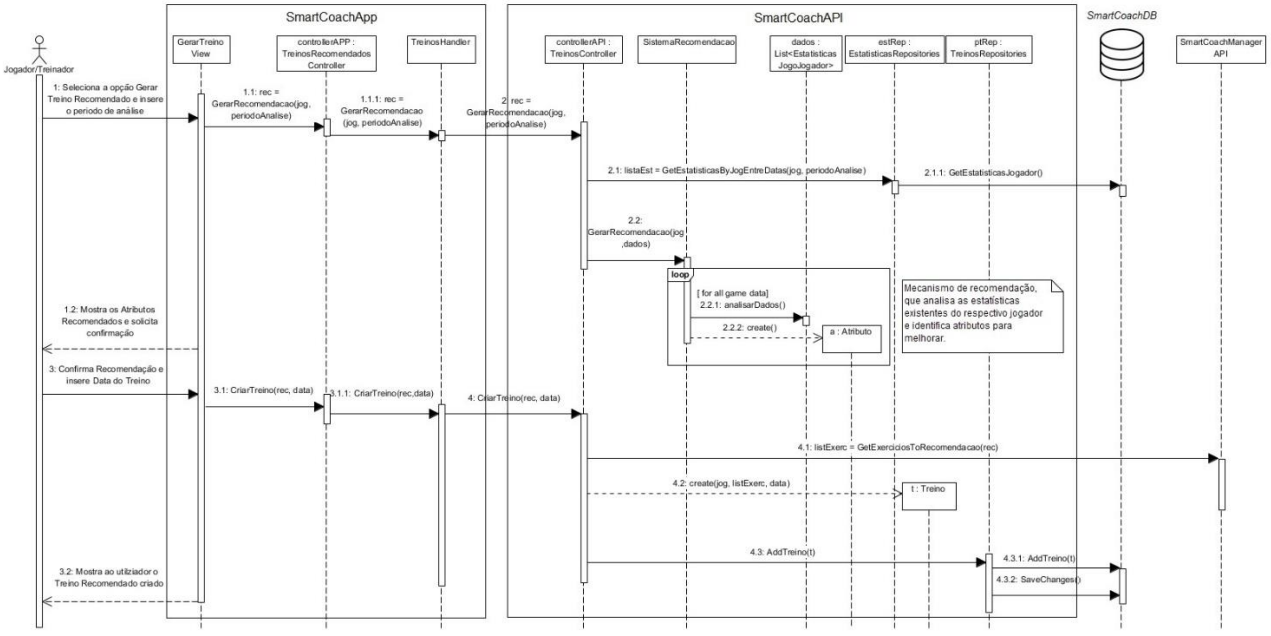


Figura 35 - Diagrama de Sequência do UC Gerar um treino recomendado

No diagrama acima, podemos visualizar que o jogador após selecionar a opção gerar plano de treino, é invocado o mecanismo de recomendação "*SistemaRecomendacao*", pelo *Controller* "*TreinosController*" da *SmartCoachAPI*. Este mecanismo tem o objetivo de analisar as estatísticas obtidas do respetivo jogador no período de análise inserido e identificar possíveis

atributos de treino a melhorar. Após esta análise, é criada uma recomendação que é prontamente, apresentada ao jogador, o mesmo confirma a recomendação, permitindo assim, a invocação do serviço externo "*SmartCoachManagerAPI*", para obtenção dos exercícios de treino mais aptos para a recomendação gerada. De seguida, é então instanciado o objeto "t" (Treino), responsável por conter o jogador em questão, a data do treino e a lista de exercícios retornados pelo respetivo serviço. Após validado pelo sistema, é realizada a persistência do treino na base de dados, através da chamada da *SmartCoachDatabase,* por parte da camada *Repositories* e por fim, apresentado ao jogador o respetivo treino criado.

4.3.5.3 UA01 – Inserir estatísticas de jogo

Este caso de uso foi escolhido por se tratar de uma funcionalidade completa a vários níveis da aplicação, uma vez que a mesma, trata de operações relacionadas com permissões de acesso a conteúdo, atualização de View, tendo em conta informação recebida, e por último de persistência de dados. Através deste UC um utilizador associado ou membro do staff técnico poderá inserir estatísticas de um jogo, para um certo jogador. De seguida, na Figura 36, é possível visualizar o diagrama de sequência do respetivo caso de uso.

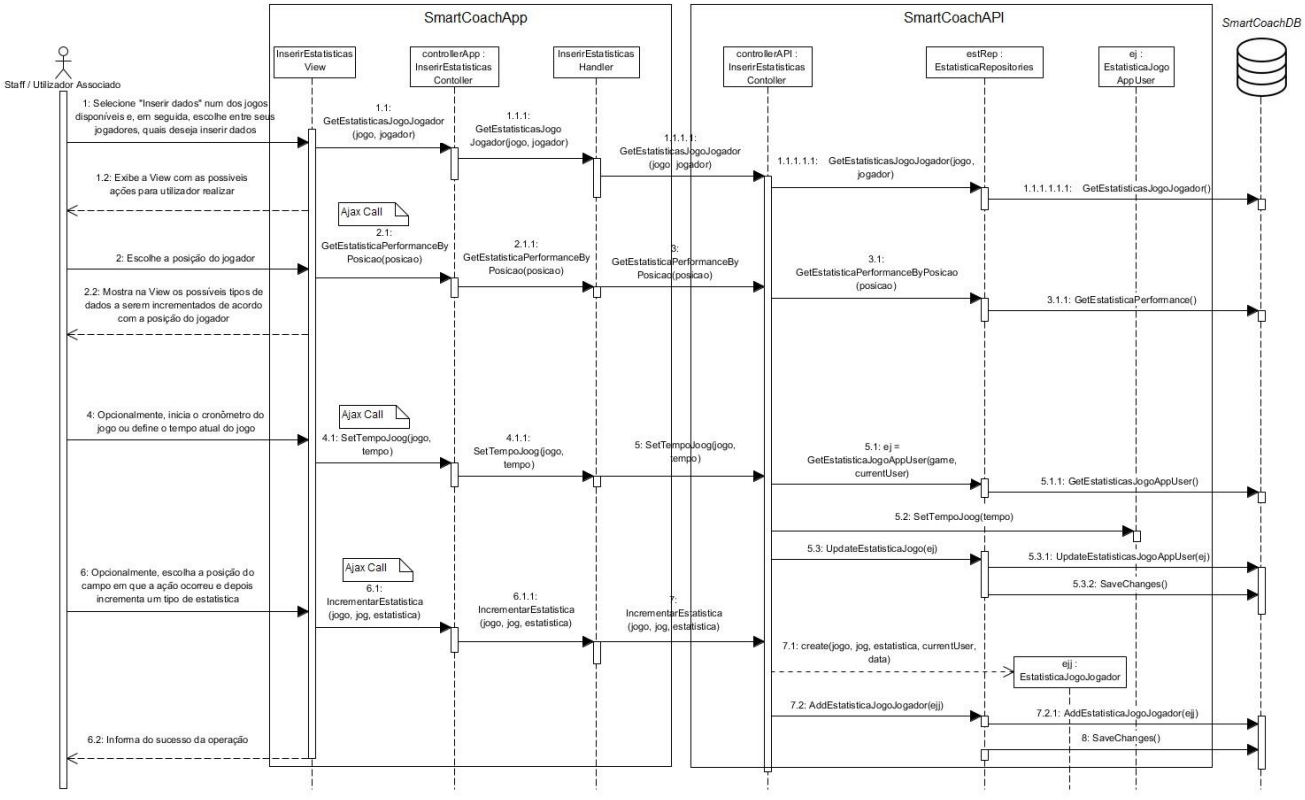


Figura 36 - Diagrama de sequência do UC Inserir estatísticas de jogo

Ao visualizar a Figura 36, é possível verificar que após o Utilizador Associado, selecionar a opção "Inserir estatísticas" para um determinado jogo e escolher o jogador a ser recolhidos dados, o sistema procede à inicialização da *view* invocando a *SmartCoachAPI*, para verificar se o utilizador já inseriu dados do respetivo jogador no jogo selecionado. Após a inicialização da

view, caso seja um jogo ao vivo, o utilizador opcionalmente, poderá inicializar o cronometro do jogo. Para além disso, o mesmo deverá inserir a posição a que jogador está a jogar, assim, através de um pedido *Ajax,* serão apresentadas na *view* as estatísticas para performance que poderão ser recolhidas, dando a opção de incrementar cada uma das mesmas. Por fim, aquando incrementada uma certa estatística, a *view* realiza um pedido *Ajax ao Controller da App, com vista a* comunicar com a API, de modo, a persistir os dados incrementados na Base de dados *SmartCoachDatabase*, através da classe *EstatisticasRepositories*.

4.4 Análise de Alternativas

A análise de alternativas é uma importante etapa no processo de desenvolvimento de software, uma vez que é através desta, que são identificadas e estudadas outras estratégias e abordagens a adotar. Após a identificação de todas as abordagens, deve-se fazer um estudo das mesmas, com vista a verificar as vantagens e desvantagens de cada uma delas, concluindo no final, qual será a melhor estratégia para se implementar na solução final.

Aquando do desenvolvimento do Design do projeto, surgiu na vista Lógica mais do que uma abordagem possível. Assim, de modo, a identificar e justificar a abordagem possível, será na secção abaixo detalhada a alternativa tomada em conta, assim como, uma síntese comparativa, entre a solução escolhida e a alternativa.

4.4.1 Vista Lógica Alternativa

Considerando a abordagem adotada, é identificada uma nova solução a nível de componentes e serviços, a mesma, trata-se de uma abordagem baseada em diferentes micro serviços, capaz de dividir as responsabilidades de domínio em cinco componentes. Para tal deveria proceder-se à divisão do serviço, atualmente existente na solução.

Esta abordagem vem responder à solução identificada na secção **4.3.1 - Vista Lógica**, que apresenta uma arquitetura monolítica, em que a API tem todas as responsabilidades de acesso aos dados, como pode ser verificado analisando a Figura 27. Por outro lado, a alternativa proposta, baseia-se na implementação de uma API *Gateway*, que comunique diretamente com cinco micro serviços, correspondentes a cada uma das principais entidades do sistema, tal como é visível na Figura 37.

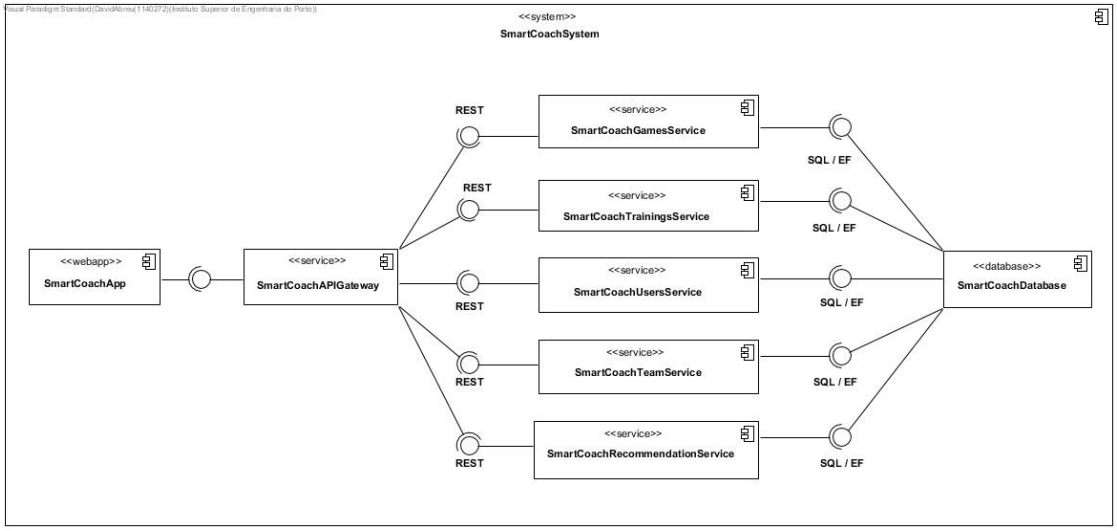


Figura 37 - Diagrama de componentes da vista lógica baseada em microservices

Assim, na Figura 37, é possível verificar que a aplicação *SmartCoachApp*, utilizada pelo cliente comunicaria com uma API *Gateway* com a responsabilidade de invocar via *REST* cada um dos cinco micro serviços existentes, nomeadamente, o *SmartCoachGamesService*, o *SmartCoachTrainingsService*, o *SmartCoachUsersService*, o *SmartCoachTeamService* e o *SmartCoachRecommendationService*. Estes serviços são associados a certas parcelas de domínio, contendo diferentes responsabilidades e regras de negócio. Posteriormente, cada um dos micro serviços identificados acederia à *SmartCoachDatabase*, com vista a realizar as operações necessárias para a persistência e retorno de certos dados.

A adoção desta abordagem é destinada a equipas desenvolvimento grandes e tem como principais pontos positivos, ser formada por serviços independentes que possibilitam uma melhor organização de responsabilidades, permitindo assim, um maior controlo de acesso aos dados. Para além disso, ao utilizar esta abordagem, caso ocorra um erro é mais fácil de identificar e corrigir o mesmo, devido a existir um isolamento de informação. Por fim, é ainda considerado que o uso de micro serviços é um mecanismo simples de entender e simples de implantar. Já em relação aos aspetos negativos, temos que o desenvolvimento desta abordagem é um processo complexo de implementar e integrar com os restantes componentes, tornando difícil a produção de testes de integração com os outros componentes existentes. Para terminar, o uso desta arquitetura, causa normalmente um aumento do consumo da memória e do tempo de resposta [55] [56].

De seguida, Tabela 4 resume as vantagens e desvantagens identificadas acima, ao utilizar esta arquitetura.

Tabela 4 - Vantagens e Desvantagens da solução baseada em micro serviços

Vantagens	Desvantagens
Serviços independentes	Complexo de implementar
Simples de implantar	Complexo de integrar com restantes componentes
Isolamento de falhas	Realização de testes de software
Organização das responsabilidades	Aumento do consumo de memória e do tempo de resposta

Por outro lado, comparando com a arquitetura monolítica esta é destinada a equipas pequenas e apresenta vantagens na facilidade da implementação, da integração, da realização de testes, e por último, o tempo de resposta associados a esta arquitetura será mais curto, possibilitando assim uma maior eficiência do sistema. Já em relação aos aspetos negativos, temos a existência de dependência dos dados, que poderá dificultar a alteração de regras de domínio, e também, caso existe um erro, o mesmo poderá bloquear todo o sistema, e a identificação do erro pode ser uma tarefa complicada [56].

De seguida, Tabela 5 resume as vantagens e desvantagens da adoção desta solução.

Tabela 5 - Vantagens e Desvantagens da solução monolítica

Vantagens	Desvantagens
Fácil de implementar	Dependências de dados
Fácil de integrar	Erro bloqueia sistema
Facilidade da elaboração de testes	Difícil identificar origem de um erro
Tempo de resposta curto	-

Por fim, após a análise dos prós e contras de cada abordagem, podemos concluir que a alternativa exposta relativa a uma arquitetura baseada micro serviços, é uma abordagem viável, contudo, não a melhor opção a ter em conta para o rumo que se pretende dar ao projeto, isto devido, à elevada complexidade em torno da sua implementação. Desse modo, a abordagem relativa a uma arquitetura monolítica será a escolhida para o desenvolvimento deste projeto.

5 Implementação

O presente capítulo, irá descrever processo adotado para desenvolvimento do software, assim sendo, primeiramente, será descrito o sistema de recomendação elaborado, seguindo-se uma explicação do desenvolvimento da aplicação servidora, bem como, da aplicação *web*. Por último, serão ainda expostos a implementação dos casos de uso do sistema.

5.1 Sistema de Recomendação

Nesta secção, será detalhado o processo de desenvolvimento do Sistema de Recomendação presente na aplicação, assim, primeiramente, será apresentada uma introdução teórica, aonde são descritas mais detalhadamente, as possíveis abordagens do sistema de recomendação, seguindo-se a sua implementação prática.

5.1.1 Introdução

Como já foi mencionado anteriormente, o sistema de recomendação faz com que o presente software se diferencia dos restantes por este se tratar de uma componente única desta aplicação. Desse modo, o mesmo deve ser cuidadosamente implementado, com vista a obter um resultado de recomendação preciso, de acordo com as necessidades de um certo jogador.

No contexto deste projeto, e como foi citado na secção **2.2.7 - Possíveis abordagens para o Sistema de Recomendação**, após um estudo completo do conceito de Sistema de Recomendação foram distinguidas duas possibilidades para a implementação do SR, após uma análise de todos os intervenientes do presente projeto, foi então decidido que o sistema de recomendação a ser produzido iria seguir a técnica de recomendação baseada em conteúdo, no entanto, a abordagem híbrida não foi posta de parte, sendo produzido o artigo teórico "*Smart Coach—A Recommendation System for Young Football Athletes*", publicado no "*International Symposium on Ambient Intelligence"* que aborda esta temática [57].

De seguida, é documentada a proposta de solução presente no respetivo artigo, esta através de uma arquitetura composta em três níveis é ilustrada na Figura 38.

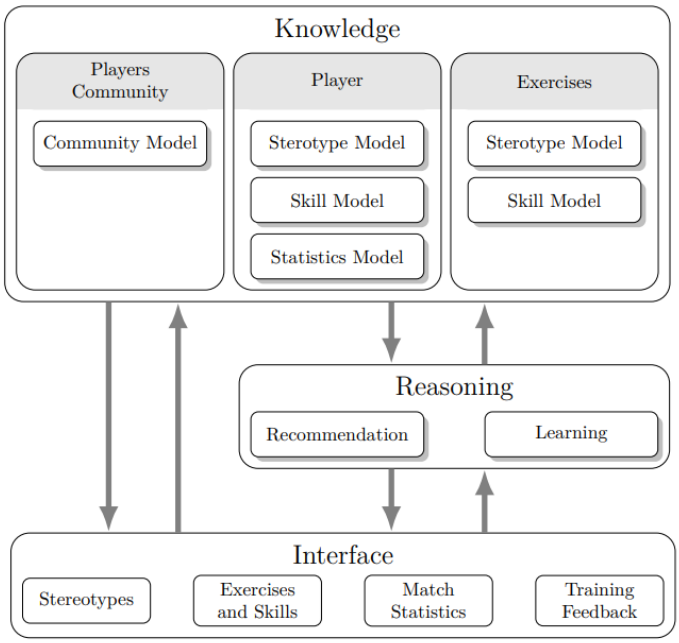


Figura 38 - Proposta de arquitetura do sistema híbrido [57]

O primeiro nível definido como *Knowledge*, agrupa as informações em três grupos distintos e as associa a modelos usados pelo sistema de recomendação. O segundo nível (*Reasoning*), apresenta o mecanismo de recomendação e módulo de aprendizagem, usando dados dos níveis de conhecimento e de interface, recomendando assim os atributos que o jogador deve melhorar e os exercícios adequados para melhorar esses atributos. Por fim, o nível da interface permite a entrada e saída de dados para o sistema, ou seja, o conjunto de estatísticas recolhidas do jogador [57].

Para aprofundar mais este tema, é detalhado como a componente *Reasoning* do software iria operar. Neste componente, o mecanismo de recomendação e o módulo de treino do algoritmo irão analisar o *feedback* sobre os exercícios de treino, assim como, a percentagem de desempenho dos treinos já recomendados do respetivo jogador (inserida no modelo do jogador), esta percentagem tem como objetivo, a identificação se o treino recomendado foi bem-sucedido ou não. Como resultado, com base nos dados pessoais do jogador, juntamente com a análise do histórico de desempenho, de treino e de *feedback* (mencionada acima), a recomendação do jogador é aprimorada com sugestões de exercícios de treino cada vez mais precisos face às suas necessidades, como apresentado na Figura 39 [57].

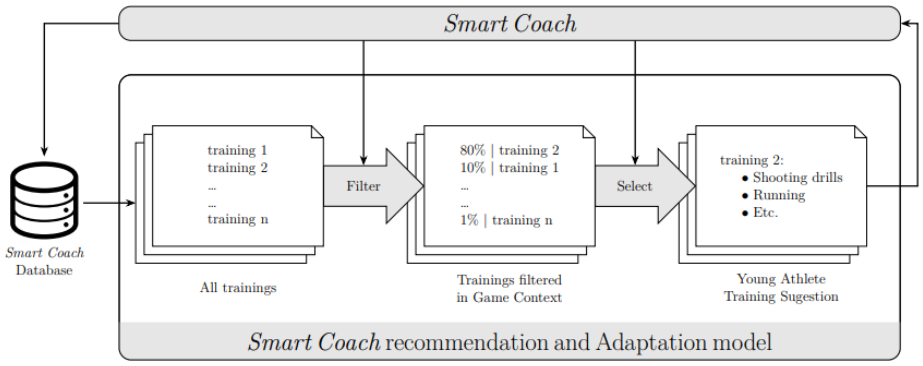


Figura 39 - Modelo de adaptação e mecanismo de recomendação [57]

Por outro lado, para o sistema de recomendação adotado (baseado em conteúdo), em termos de arquitetura o mesmo, irá se inserir na vista *back-end* do sistema, mais concretamente, será um subcomponente da *SmartCoachAPI*. Este módulo, tem como objetivo operar sobre os modelos já definidos do sistema, presentes na secção **4.2.5 - Modelo de Domínio**, e através da análise dos dados existentes na base de dados, nomeadamente, o cruzamento dos dados estatísticos recolhidos do jogador com uma análise ao histórico de jogadores de mesma posição, bem como, aos valores referência, irá ser possível realizar uma recomendação de atributos de treino, de acordo com as necessidades do jogador. Para a detalhar este processo de recomendação, foi inicialmente realizado um esboço que divide o mesmo em sete etapas:

1. Ir buscar todos os dados do respetivo jogador, para o período de analise selecionado;
2. Agrupar os dados dos mesmos jogos de utilizadores diferentes;
3. Determinar o peso de avaliação [0.0 a 1.0] para cada um dos jogos existentes no período de analise (tendo em conta as variáveis de contexto);
4. Consoante o peso do jogo obtido, determinar para cada estatística recolhida a sua [*Analise*], que contém o número ocorrências e a (%) eficácia / (%) tipo de estatística (se aplicável), diferenciando de estatística principal para secundária, com base na posição que o jogador jogou;
5. Para cada [*Analise*] calcular o valor de recomendação, através da comparação com valores referência, bem como, com a média dos valores de jogadores da mesma posição (num período total);
6. Obter o valor de recomendação para cada um dos Atributos de treino existentes na estatística do [*Analise*], tendo em conta, a classificação do Atributo na respetiva estatística, de seguida, adicionar o atributo à lista de recomendação;
7. Remover da lista os Atributos de recomendação que estejam abaixo de um valor recomendação mínimo (0.4) e ordenar a lista resultante.

Tendo em conta o fluxo acima, pode-se concluir que em primeira instância, o sistema usará unicamente informações sobre o estereótipo do jogador e valores referência considerados relevantes pelo treinador para executar o algoritmo, após o período inicial de recolha de dados, o mesmo, já irá puder analisar o histórico de dados estatísticos do jogador e da sua equipa, assim como, de outros jogadores do sistema. De modo a completar o fluxo descrito acima e auxiliar a implementação do sistema de recomendação, foi produzido um

esboço em *pseudo-código* do algoritmo base de recomendação a ser implementado, este é ilustrado em Algoritmo 1.

Algoritmo: Algoritmo base do sistema de recomendação

```
//Inicializar mapa com atributo/valor recomendação
AtributosValorRec ← new Map<Atributo, Valor> ();
ValorRecomendacaoMinimo ← 0.4;
PesoAtributoPrimario ← 1.0;
PesoAtributoSecundario ← 0.6;
//Agrupar dados recolhidos
ListaMediaEstatisticasRecolhidasJogo ← agruparEstatisticasJogoECalcularPesoJogo(dados);
EstatisticasTotais ← agruparEstatisticasByPesoJogo (ListaMediaEstatisticasRecolhidasJogo);
//Preparar dados para recomendação
foreach Estatística ∈ EstatisticasTotais do
    ImportanciaEstatistica ← getImportanciaEstatisticaByPosicaoJogadorJogo(Jogador, Estatística);
    if (ImportanciaEstatistica == AtributoPrincipal) do
        AnaliseRecomendacao ← criarAnaliseRecomendacao(Estatística, PesoAtributoPrimario);
    end if
    if (ImportanciaEstatistica == AtributoSecundario) do
        AnaliseRecomendacao ← criarAnaliseRecomendacao(Estatística, PesoAtributoSecundario);
    end if
    ListaAnaliseRecomendacao.Add(analiseRecomendacao);
end foreach
//Aplicar Recomendação para cada estatística
foreach AnaliseRecomendacao ∈ ListaAnaliseRecomendacao do
        AtributosValorRec ← calcularValorRecomendacaoEstatistica(AnaliseRecomendacao, AtributosValorRec)
end foreach
//Ordenar mapa por valor de recomendação e remover os abaixo do valor mínimo
ordenarObterAtributosRec (AtributosValorRec, ValorRecomendacaoMinimo);
```

Algoritmo 1 - Algoritmo base do sistema de recomendação

5.1.2 Implementação

Para a desenvolvimento do Sistema de Recomendação, foi utilizada a linguagem de programação C#, presente no *ASP.Net*. Isto devido ao facto deste módulo ser considerado um subcomponente da *SmartCoachAPI,* documentada de seguida no capítulo **5.2 - Aplicação Servidora**. Assim, com base no fluxo inicial detalhado na secção anterior, juntamente, com uma análise mais aprofundada ao tema foram identificadas as classes necessárias para realizar a recomendação de atributos de treino, estas, podem ser visualizadas na Figura 40.

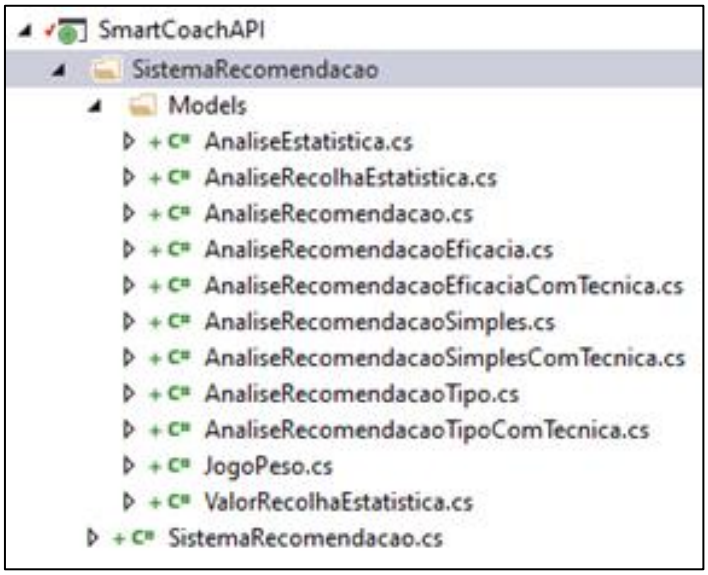


Figura 40 - Componente do Sistema de Recomendação

Resumidamente, na figura acima pode-se diferenciar a classe "*SistemaRecomendacao.cs*", como a classe principal, que contém todo o processo de recomendação, já em relação às restantes, é possível verificar que contêm a lógica necessária, bem como, operações auxiliares para mecanismo de recomendação. Mais detalhadamente, as classes "*AnaliseRecolhaEstatistica.cs*" e "*ValorRecolhaEstatistica.cs*", são utilizadas, para o agrupar e guardar as médias dos dados estatísticos do jogador no período de análise. Já em relação à classe "*AnaliseRecomendacao.cs"* e suas classes filho, estas, consoante a estatística a analisar, guardam o número de ocorrências, a percentagem de eficácia (se aplicável) e o número/percentagem de ocorrências por tipo (se aplicável), é ainda de salientar, que as mesmas, estão preparadas para analisar estatísticas consoante a técnica de execução (Pé direito, Pé Esquerdo, Cabeça). Por último, as classes "*JogoPeso.cs*" e "*AnaliseEstatistica.cs*", tem funções auxiliares para o cálculo do peso de cada Jogo existente no período de análise, assim, como, cálculo do valor de recomendação para a estatística a analisar.

De seguida, para complementar o fluxo e o algoritmo do processo de recomendação identificado na secção anterior, foi produzido um diagrama de sequência, este tem o objetivo de validar a possível realização das etapas do respetivo processo, bem como, auxiliar a implementação do sistema de recomendação. Assim, na Figura 41 é ilustrado o respetivo diagrama.

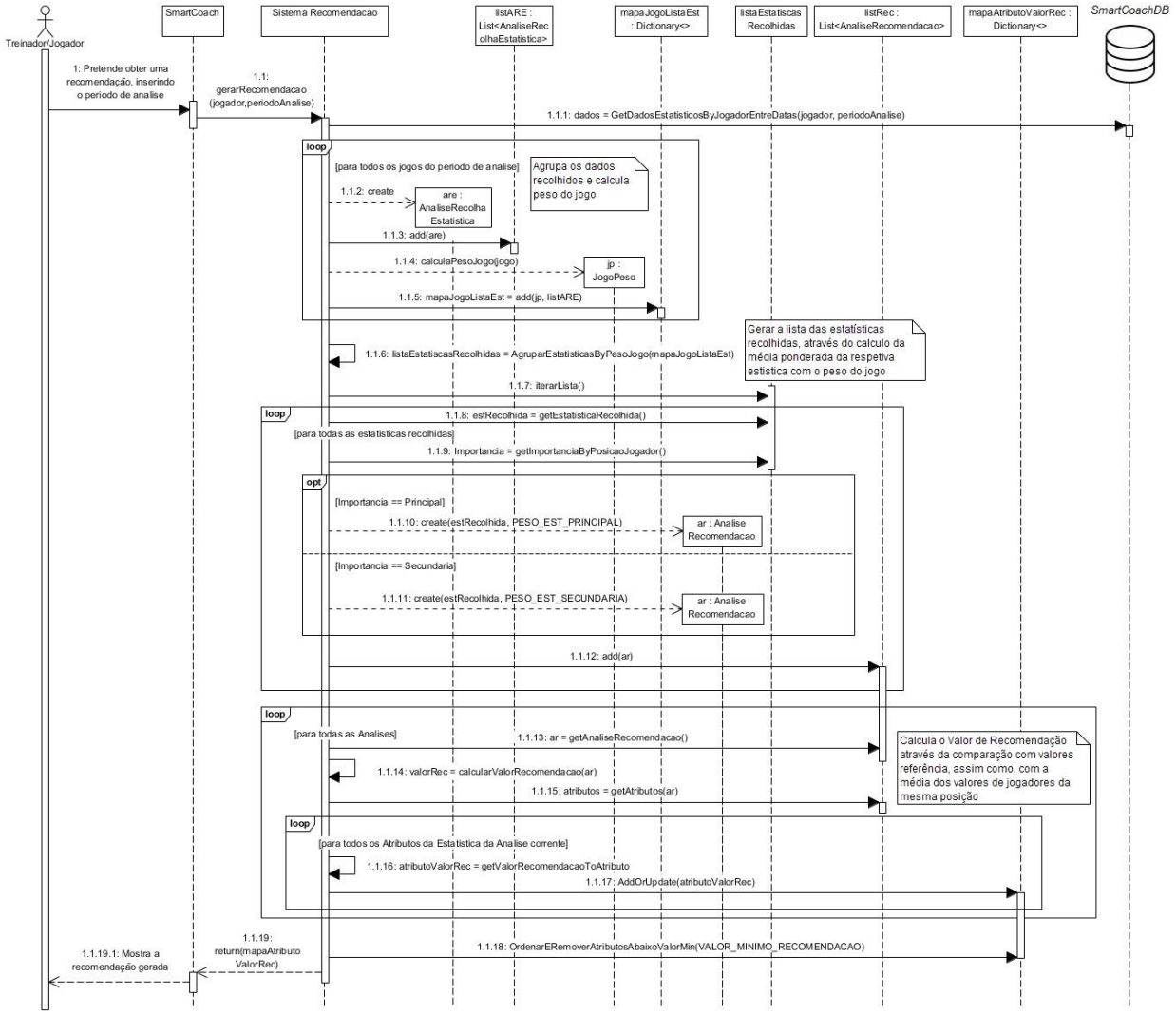


Figura 41 - Diagrama de Sequência do Sistema de Recomendação

Ao analisar o diagrama, torna-se possível verificar que a *SmartCoachAPI* ao invocar o sistema de recomendação é inicialmente realizada uma chamada à base de dados para ir buscar as informações estatísticas recolhidas do respetivo jogador no período de analise selecionado. Após isso, os dados recolhidos são agrupados por jogo (no caso de haver mais do que um utilizador a inserir dados para o conjunto jogador/jogo), calculando a média para cada estatística presente, é de salientar que estes valores são guardados em instâncias da classe "*AnaliseRecolhaEstatistica*" e adicionados ao "*mapaJogoListEst*" que se trata de um elemento do tipo "*Map*/*Dicitonary",* que contém como "*Key*" o jogo, e como "*Values*" a lista de classes *AnaliseRecolhaEstatistica* instanciadas. De seguida, é então avaliado o peso que cada jogo presente no "*mapaJogoListEst*" possui, valor entre 0.0 e 1.0. A respetiva operação é realizada analisando três variáveis de contexto, nomeadamente, as condições atmosféricas, o tipo de piso e por último, a qualidade do piso, estas relacionam a dificuldade da execução de ações desportivas consoante o estado do campo no momento que o jogo foi realizado. Resumidamente, a determinação do peso do jogo, tem como objetivo dar mais valor a jogos em condições normais e desvalorizar jogos em que as condições tenham sido longe das normais.

Desse modo, para complementar a descrição dada, é apresentado um excerto de código, na Figura 42, que representa o cálculo geral do peso do jogo, assim como, cada uma das três funções de obtenção do peso por variável de contexto nas Equação 1, Equação 2 e Equação 3.

```
public static double PesoTipoPiso = 0.3;
public static double PesoQualidadePiso = 0.3;
public static double PesoCondicoesAtmosfericas = 0.4;

1 reference | 0 changes | 0 authors, 0 changes
private static double caclularPesoJogo(Jogo jogo)
{
    var condAtmosfericas = CondicoesAtmosfericasExtensions.GetPesoByCondicoesAtmosfericas(jogo.CondicoesAtmosfericas);
    var tipoPiso = TipoPisoExtensions.GetPesoByTipoPiso(jogo.TipoPiso);
    var qualidadePiso = QualidadePisoExtensions.GetPesoByQualidadePiso(jogo.QualidadePiso);
    return (PesoTipoPiso * tipoPiso) + (PesoQualidadePiso * qualidadePiso) + (PesoCondicoesAtmosfericas * condAtmosfericas);
}
```

Figura 42 - Excerto de código referente ao cálculo do peso do jogo

$$\text{GetPesoByCondicoesAtmosfericas}(x) = \begin{cases} 0.75, & x = N/A \\ 1.00, & x = \text{TempoLimpo} \\ 0.90, & x = \text{Nublado} \\ 0.75, & x = \text{Aguaceiros} \\ 0.50, & x = \text{Tempestade} \\ 0.50, & x = \text{Neve} \end{cases}$$

Equação 1 - Função de obter o peso da variável de contexto condições atmosféricas

$$\text{GetPesoByTipoPiso}(x) = \begin{cases} 0.75, & x = N/A \\ 1.00, & x = \text{RelvadoNatural} \\ 0.75, & x = \text{RelvadoSintetico} \\ 0.50, & x = \text{TerraBatida} \end{cases}$$

Equação 2 - Função de obter o peso da variável de contexto tipo de piso

$$\text{GetPesoByQualidadePiso}(x) = \begin{cases} 0.75, & x = N/A \\ 1.00, & x = \text{Bom} \\ 0.75, & x = \text{Medio} \\ 0.50, & x = \text{Fraco} \end{cases}$$

Equação 3 - Função de obter o peso da variável de contexto qualidade de piso

Após o cálculo do peso do jogo é então invocada a função "*agruparEstatisticasByPesoJogo()*" esta tem a finalidade de agrupar todas as estatísticas obtidas numa única lista, através da realização de uma média ponderada do conjunto de estatísticas obtidas com o peso do jogo em que as mesmas foram recolhidas.

Posteriormente à obtenção da lista geral das estatísticas agrupadas, é necessário analisar cada uma delas individualmente, para isso, são então geradas instâncias do objeto "*AnaliseRecomendacao*", consoante o seu tipo de estatística para performance (Simples, Eficácia ou Tipo), documentadas na secção **4.2.6.6 - Estatística para Performance**. Estes novos

objetos, contém toda informação necessária para análise por parte do mecanismo de recomendação, nomeadamente, a estatística e o seu número ocorrências / (%) eficácia (se aplicável) / (%) tipo de estatística (se aplicável), assim como, o peso da estatística (Primária [1.0] ou Secundária [0.6]), que varia de acordo com a posição que o jogador jogou.

Seguidamente, é então calculado o valor de recomendação para cada objeto "*AnaliseRecomendacao".* Como enunciado anteriormente, esta operação vai comparar os valores presentes na análise com os valores referência da sua equipa (que vale 50% na nota de recomendação), assim como, com os valores médios de jogadores semelhantes (que vale outros 50% na nota de recomendação), em seguida é então apresentada a fórmula geral:

Fórmula: Fórmula do cálculo do valor de recomendação

$$\boldsymbol{Rec_x} = (\boldsymbol{ValRef_x} \times \mathbf{0.5}) + (\boldsymbol{ValM_x} \times \mathbf{0.5})$$

$\boldsymbol{Rec_x} - Valor\ de\ Recomendação\ da\ Estatística\ x;$

$\boldsymbol{ValRef_x} - Valor\ de\ Referência\ da\ Estatística\ x;$

$\boldsymbol{ValM_x} - Valor\ Médio\ de\ jogadores\ semelhantes\ da\ Estatística\ x;$

Equação 4 – Cálculo do valor de recomendação

Para melhor compreensão da fórmula enunciada acima, torna-se necessário decompor a mesma, documentando cada uma das entidades presentes:

A primeira, $\boldsymbol{ValRef_x}$ é responsável pela comparação do número médio de ocorrências da estatística x com o valor referência associado. Mais concretamente, através da verificação do operador do valor referência (≥ ou ≤) da estatística x é realizado o inverso da percentagem de ocorrência, ou seja, no caso de se tratar do operador maior ou igual, será realizada a operação*: **1 – (número ocorrências / valor referência)***, já no caso oposto, será a operação: ***1 – (valor referência / número ocorrências)***. Obtendo assim, o valor de recomendação, como apresentado no excerto de código da Figura 43.

```
double valorRecomendacao = 0;
if (v.OperacaoValor == OperacaoValor.MenorIgual)  {
    if (numOcorrencias > 0 && v.Valor.HasValue){
        valorRecomendacao = 1 - (v.Valor.Value / numOcorrencias);
    }
} else if(v.OperacaoValor == OperacaoValor.MaiorIgual) {
    if (v.Valor.HasValue && v.Valor.Value > 0) {
        valorRecomendacao = 1 - (numOcorrencias / v.Valor.Value);
    }
}
if (valorRecomendacao < 0) {
    return 0.0;
}
return valorRecomendacao;
```

Figura 43 - Excerto de código referente ao cálculo do valor de recomendação por valor referência

Por outro lado, em relação ao cálculo do $\boldsymbol{ValM_x}$, este é alcançado, após a obtenção dos jogadores semelhantes (jogadores y), via comparação com a posições do jogador a avaliar. E de seguida, através da realização de uma média ponderada da estatística x, com a classificação que o jogador y obteve em cada jogo. De modo a alcançar resultados pertinentes, é de salientar que só são analisados os jogos que o jogador y, obteve uma classificação maior ou igual a 6, assim sendo, a Equação 5 apresenta de maneira mais percetível a fórmula implementada. Por fim, após ser calculada a média ponderada da estatística x é realizada a comparação do valor da média obtida com o número de ocorrências do jogador, mediante o inverso da percentagem (idêntico ao realizado com o valor referência, documentado na Figura 43).

Fórmula: Fórmula do cálculo do $ValM_x$

$$\boldsymbol{M_x = \frac{\sum_i^n x_i C_i \ \text{se } C_i \geq 6}{\sum_i^n C_i \ \text{se } C_i \geq 6}}$$

$\boldsymbol{M_x} - Média\ Ponderada\ da\ Estatística\ x, nos\ jogos\ em\ que\ jogadores\ tiveram\ classificação \geq 6;$

$\boldsymbol{n} - Numero\ de\ Jogadores/Jogo\ existentes;$

$\boldsymbol{i} - Índice\ do\ Jogador/Jogo;$

$\boldsymbol{x_i} - Valor\ da\ Estatística\ Recolhida\ x\ do\ respetivo\ Jogador\ no\ Jogo;$

$\boldsymbol{C_i} - Classificação\ do\ respetivo\ Jogador\ no\ jogo;$

Equação 5 - Cálculo do valor médio da estatística x para jogadores semelhantes

Após ser obtido o valor de recomendação para a estatística x, é necessário obter a nota de recomendação para todos os atributos presentes na estatística. Para isso, será analisado individualmente cada um dos atributos, convertendo a sua classificação (valor entre 1 e 5), em valor percentual [0.0 – 1.0]. Assim, para cada um dos atributos é realizado o produto do valor percentual obtido com o valor de recomendação da estatística x, adicionando ou atualizando cada um dos atributos num Mapa <Atributo, Valor>.

Por fim, obtidos os valores de recomendação para cada um dos atributos presentes nas estatísticas, torna-se necessário remover aqueles abaixo de um valor mínimo estipulado de 0.4, (é de realçar, que este valor foi adotado em conformidade com todos os intervenientes deste projeto) e ordenar a lista resultante.

5.2 Aplicação Servidora

Nesta secção, será detalhada o desenvolvimento da aplicação servidora *SmartCoachAPI* e seus subcomponentes. Primeiramente, é importante enunciar que o desenvolvimento de uma API, deveu-se à necessidade de comunicar com outros serviços e componentes do sistema, assim como, desenvolver de forma completa o presente projeto, implementando certos padrões de software que possibilitam auferir de uma melhor organização, resultando numa fácil manutenção do sistema. Este componente foi desenvolvido em C# utilizando a *framework Asp.NET Web API*. O componente principal SmartCoachAPI, divide-se em nove subcomponentes, descritos e justificados na secção **4.3.1.2 - Arquitetura *Backend***. É ainda de salientar que para a implementação de cada subcomponente identificado, foi tido em conta o design elaborado ao longo da secção **4 - Design da Solução**.

5.2.1 Autenticação

A autenticação é um conceito existente em sistemas informáticos que pretende restringir o acesso a dados consoante a existência de uma conta de utilizador no sistema. Assim, o mecanismo de autenticação solicita a identidade do utilizador, através dos dados de início de sessão, antes de revelar informações confidenciais da aplicação.

Este tópico é de máxima importância, uma vez que se trata da segurança do acesso a funcionalidades e dados do sistema que não devem puder ser acedidos por utilizadores não registados no sistema. Assim sendo, o mesmo foi implementado cuidadosamente, utilizando a *framework*, *ASP.NET Identity.* A adoção desta ferramenta irá permitir um sistema de identidade único com as seguintes características [58]:

- Facilidade de conectar dados de perfil sobre o utilizador;
- Controle de persistência;
- Testes de unidade;
- *Role provider*;
- Integração OWIN;
- Entre outros.

Ao utilizar esta *framework*, são automaticamente geradas diversas classes relacionadas com esta matéria, nomeadamente, as classes *AccountController*, *IdentityModels*, *IdentityConfig,* entre outras, estas apresentam a lógica de negócio e o conjunto de operações associadas a este conceito. De seguida, na Figura 44 será apresentado um excerto referente à classe *IdentityConfig*, esta tem como finalidade permitir atualizar as configurações do registo do utilizador, tais como validações para a escolha do seu *username*/email e password de início de sessão, assim como, ativar o uso de *Tokens*, para a possível autenticação com serviços externos à aplicação servidora.

```
1 reference | David Abreu, 156 days ago | 1 author, 1 change
public static ApplicationUserManager Create(IdentityFactoryOptions<ApplicationUserManager> options, IOwinContext context)
{
    var manager = new ApplicationUserManager(new UserStore<ApplicationUser>(context.Get<ApplicationDbContext>()));
    // Configure validation logic for usernames
    manager.UserValidator = new UserValidator<ApplicationUser>(manager)
    {
        AllowOnlyAlphanumericUserNames = false,
        RequireUniqueEmail = true
    };
    // Configure validation logic for passwords
    manager.PasswordValidator = new PasswordValidator
    {
        RequiredLength = 6,
        RequireNonLetterOrDigit = false,
        RequireDigit = false,
        RequireLowercase = false,
        RequireUppercase = false,
    };
    var dataProtectionProvider = options.DataProtectionProvider;
    if (dataProtectionProvider != null)
    {
        manager.UserTokenProvider = new DataProtectorTokenProvider<ApplicationUser>(dataProtectionProvider.Create("ASP.NET Identity"));
    }
    return manager;
}
```

Figura 44 - Excerto de código da classe *IdentityConfig,* referente à autenticação

5.2.2 Autorização

Já em relação à autorização, este é um conceito que pretende restringir o acesso a funcionalidades consoante o cargo/role que o utilizador autenticado desempenha no sistema. Assim, para a implementação deste requisito, foi também utilizada a ferramenta *Identity*, uma vez que a mesma, disponibiliza o conceito de "*Role provider*", como enunciado anteriormente. Assim, de maneira a restringir o acesso às funcionalidades do sistema através do cargo do utilizador, foi utilizada o mecanismo de [*Authorize* (Roles = "NomeCargo")], nos cabeçalhos dos métodos das classes *Controller*. A Figura 45 ilustra um excerto de código da classe *EquipasController*, que tem como objetivo remover utilizadores da sua equipa, e só poderá ser acedido por Treinadores.

```
[Route("RemoverUtilizadorEquipa")]
[Authorize(Roles = "Treinador")]
[HttpPost]
0 references | David Abreu, 111 days ago | 1 author, 1 change
public async Task<IHttpActionResult> RemoverUtilizadorEquipa(RemoverUserDTO objDTO)
{
```

Figura 45 - Excerto de código referente à autorização

5.2.3 Acesso a dados

Segundo a arquitetura proposta a aplicação servidora será responsável pelo acesso aos dados, assim sendo, a mesma tem de estabelecer uma ligação com a *SmartCoachDatabase*, essa ligação é realizada através da *connectionString* presente no ficheiro *WebConfig*. O **Anexo 1 – Tabelas da SmartCoachDatabase**, apresenta a ligação realizada e a lista de todas as tabelas presentes na *SmartCoachDatabase.*

Após existir uma conexão entre a base de dados e a aplicação servidora, torna-se necessário realizar a componente de migrações da respetiva base dados, este conceito visa tratar de técnicas e ferramentas que auxiliam no versionamento da base de dados durante todo o seu processo de desenvolvimento. Assim, de modo a implementar esta atividade, foi seguido o tutorial da *Microsoft* - "*Atualizar a aplicação com Migrações Code First*" [59], que através da inserção do código "*Enable-Migrations*" na *Package Manager Console*, resultou na geração automática da pasta *Migrations*, com o ficheiro de inicialização da base de dados (*Configuration.cs*). Após isso, torna-se possível de uma maneira simples realizar alterações à *SmartCoachDatabase* sem perder o conteúdo já guardado. De seguida, a Figura 46 apresenta a lista de migrações realizadas ao longo deste projeto.

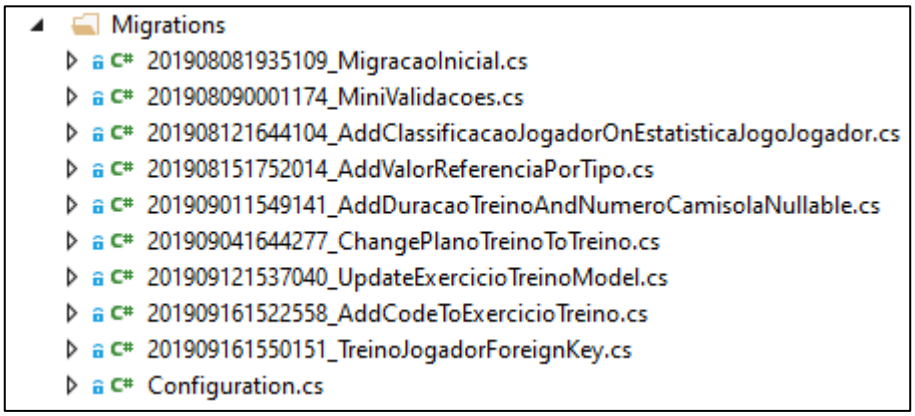


Figura 46 - Lista de migrações realizadas ao longo do projeto

De seguida, de modo, a demonstrar como se comporta a aplicação servidora, e todo fluxo do processo de chamadas entre os seus subcomponentes para a obtenção de dados, será apresentado na Figura 47 o diagrama de sequência de um exemplo de um pedido GET à *SmartCoachAPI*.

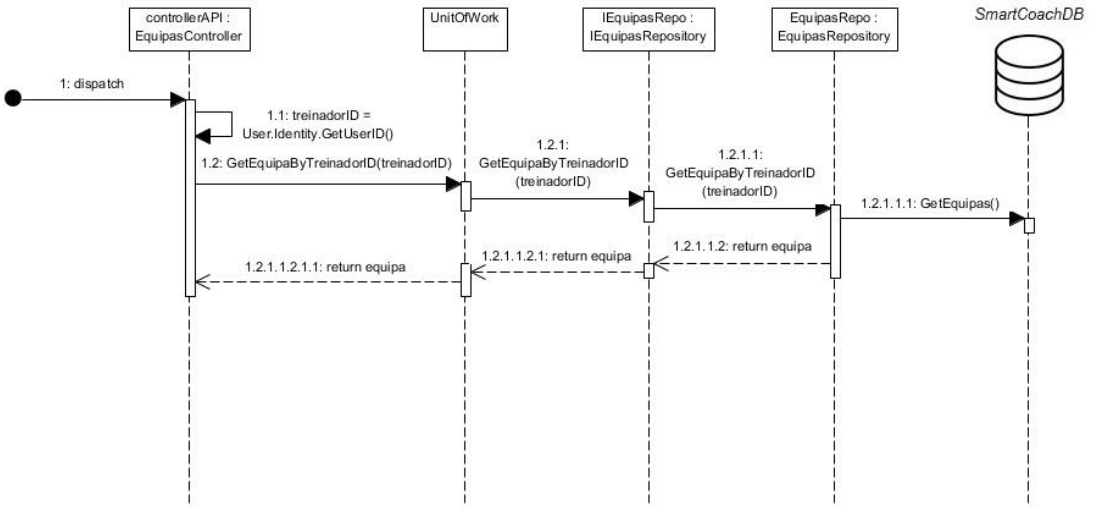


Figura 47 - Diagrama de sequência do pedido GetEquipaByTreinadorID()

5.3 Aplicação *Web*

A aplicação *Web SmartCoachApp*, consome os serviços disponibilizados pela aplicação servidora (*SmartCoachAPI*) e tem o objetivo de apresentar ao utilizador um conjunto de páginas *web* com as informações desejadas, estas podem ser acedidas via navegador da internet (*browser),* tanto por dispositivos móveis, assim como, dispositivo desktop. A aplicação *web*, foi mais uma vez produzida em C#, utilizando a *framework Asp.NET MVC* e o respetivo padrão de software *Model-View-Controller*. O componente principal divide-se em cinco subcomponentes, descritos e justificados na secção **4.3.1.3 - Arquitetura *Frontend***. É ainda de salientar que para a implementação de cada subcomponente identificado, foi respeitado ao máximo todo o design elaborado ao longo da secção **4 - Design da Solução**.

5.3.1 Autenticação e Autorização

Como já foi mencionado anteriormente, este requisito tem uma importância bastante elevada, uma vez se tratar da segurança de dados. Desse modo, visto se tratar de uma aplicação *web* sem acesso direto à base de dados, é necessário implementar um mecanismo de autenticação via *Tokens*. A obtenção do *Token (TokenResponse)* é realizado ao efetuar Login na aplicação. De seguida, o *Token* é guardado numa variável de sessão (*session)*, através do excerto do código presente na Figura 48. Assim sendo, até ao utilizador terminar sessão, as comunicações entre a aplicação *web* e a aplicação servidora são realizadas com o envio do *Token* da variável de sessão.

```
1 reference | David Abreu, 157 days ago | 1 author, 1 change
public static void storeToken(TokenResponse token)
{
    var session = HttpContext.Current.Session;
    session["token"] = token;
}
```

Figura 48 - Excerto de código responsável por guardar *Token*

Já em relação à autorização, foi utilizado o *ClaimsIdentity*, para guardar o nome, o e-mail e o(s) role(s) do utilizador corrente da aplicação, a Figura 49 apresenta o excerto de código referente a esta atribuição. Este conceito opera através de *Cookies* e pode-se concluir que as mesmas são removidas quando o utilizador termina sessão na aplicação, ou quando o tempo do *Token* guardado na variável de sessão expira.

```
var claims = new List<Claim>();
claims.Add(new Claim(ClaimTypes.Name, currentAppUser.GetName()));
foreach (var role in currentUserRoles) {
    claims.Add(new Claim(ClaimTypes.Role, role));
}
claims.Add(new Claim(ClaimTypes.Email, tokenResponse.Username));

var id = new ClaimsIdentity(claims, DefaultAuthenticationTypes.ApplicationCookie);
```

Figura 49 - Excerto de código referente à atribuição dos *Claims*

5.3.2 Responsividade

A aplicação *web* desenvolvida tem como um dos principais requisitos estar apta para ser acedida via dispositivo móvel. Desse modo, foram adotadas três práticas que vão ao encontro deste requisito. Primeiramente, foi definida a utilização do *bootstrap* para os estilos da página, de seguida, os tamanhos dos elementos das páginas HTML serão sempre trabalhados em percentagem (%) e não em tamanho real, e por último, foi retificado o modo de apresentação de certos elementos das páginas *web* quando as mesmas são acedidas via browser por um dispositivo mais pequeno. Esta alteração foi alcançada atualizando/adicionando novas classes no ficheiro *Site.css,* consoante a largura máxima do dispositivo do utilizador. De seguida, ao analisar a Figura 50, é possível verificar que o elemento "*dados-jogador-equipa*" é ocultado quando o dispositivo do utilizador tiver como largura máxima 700 pixéis.

```
.dados-jogador-equipa {
    display: table-cell;
    text-align: center;
}

@media (max-width: 700px) {
    .dados-jogador-equipa {
        display: none;
    }
}
```

Figura 50 - Excerto de código referente ao ficheiro *Site.css*

De seguida, de modo a comparar os resultados obtidos entre a componente móvel e a desktop, será apresentado o Painel de controlo inicial (*HomePage*), primeiramente, na Figura 51 para um dispositivo desktop, e de seguida, na Figura 52 para um dispositivo móvel.

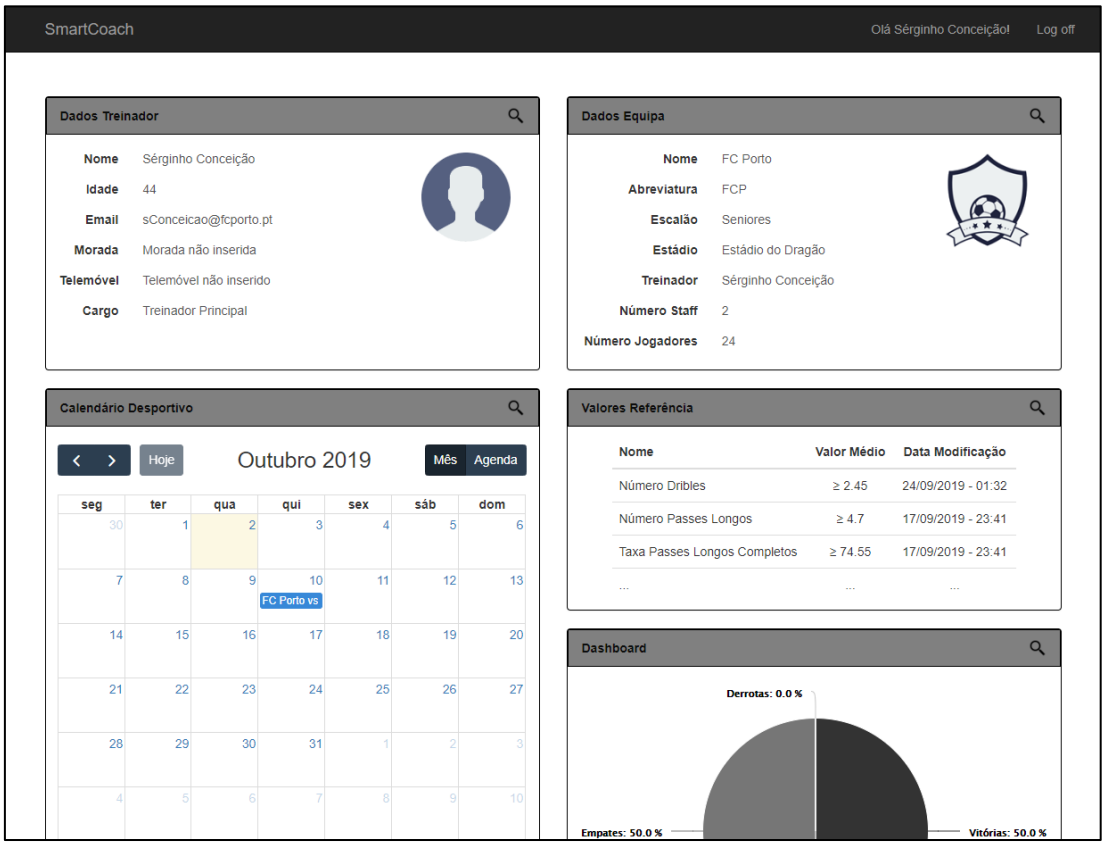


Figura 51 - Painel de controlo inicial da Aplicação (dispositivo desktop)

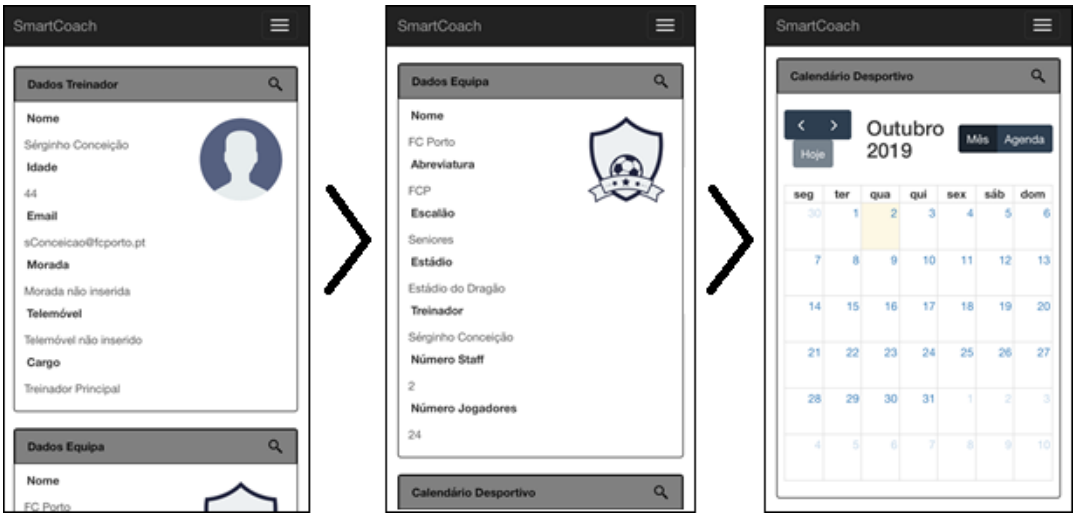


Figura 52 - Painel de controlo inicial da Aplicação (dispositivo móvel)

5.3.3 Plugins utilizados

Ao longo do desenvolvimento da aplicação *web*, foram importados diversos softwares livres (que não necessitam de aquisição de licença) para o auxílio das páginas *web*, mais concretamente, foram utilizados sete plugins *jquery* que tem o objetivo de melhorar o *output* da interface gráfica do utilizador ou possibilitar a apresentação de ferramentas visuais complementares. De seguida, serão descritos cada um destes e a sua aplicabilidade no sistema *SmartCoach*.

- **Chosen-select** [60]: Plugin de auxílio aos elementos de seleção (*selects* e *multi-selects)* presentes ao longo das páginas *web*, este tem o objetivo de tornar os mesmos, mais "*user-friendly*" e mais simples de usar. Exemplos da sua aplicabilidade são na escolha de jogadores para a inserção de estatísticas, e na escolha de jogo/jogador no componente *dashboard* estatístico;
- **CountyPicker** [61]: Software *open-source* que disponibiliza um *select*, com a lista de todos os países, incluindo a sua bandeira e o seu nome. Esta ferramenta é usada no registo do jogador, quando é solicitado a sua nacionalidade;
- **DataRangePicker** [62]: O *DataRangePicker*, trata-se de um plugin jquery para escolha de um período de tempo (entre duas datas) de forma simples numa página *web*. Este software é usado na funcionalidade de gerar treino recomendado, para a inserção do período de analise;
- **DataTables** [63]: *DataTables* é uma ferramenta que tem o objetivo de adicionar controles avançados de interação para tabelas HTML. Este plugin é utilizado em diversas páginas ao longo do projeto;
- **FullCalendar** [64]: Software *open-source*, que disponibiliza uma agenda em formato calendário com interação de eventos, com mais de 100 definições customizáveis. Este plugin foi usado para o desenvolvimento do calendário desportivo;
- **Highcharts** [65]: O software *Highcharts*, está disponível em versão "*trial*" para aplicações sem fins comerciais. Mais detalhadamente, *Highcharts* é uma biblioteca de gráficos multiplataforma baseada em SVG, através dela é possível a fácil adição de gráficos interativos otimizados para qualquer tipo de dispositivos. No presente projeto, foi utilizado esta ferramenta na componente do *dashboard* estatístico.
- **Pretty-Tabs** [66]: O plugin *Pretty-Tabs*, disponibiliza de forma simples, inteligente e personalizável separadores para as páginas *web*. Esta ferramenta foi então utilizada na componente da conta do utilizador para dividir em quatro separadores as subfuncionalidades dessa página, assim como, na página relacionada à gestão dos jogos.

5.4 Casos de Uso

De seguida, para complementar a argumentação deste capítulo serão apresentados os treze casos de uso únicos do sistema, estes dividem-se entre os vários atores da aplicação e apresentam todas as funcionalidades e subfuncionalidades do software. De modo a documentar os respetivos casos de uso, foi adotada a seguinte abordagem, primeiramente, será fornecida uma descrição do UC, indicando o que este faz e quem o executa, de seguida, serão disponibilizadas figuras alusivas ao resultado obtido, e por fim, será explicado como foi implementada a codificação do mesmo, com a possibilidade de inserção de excertos de código relevantes.

5.4.1 Login

O presente caso de uso é realizado por um utilizador não autenticado que pretende garantir a sua autenticação no sistema, através da inserção dos seus dados de início de sessão, como apresentado na Figura 53.

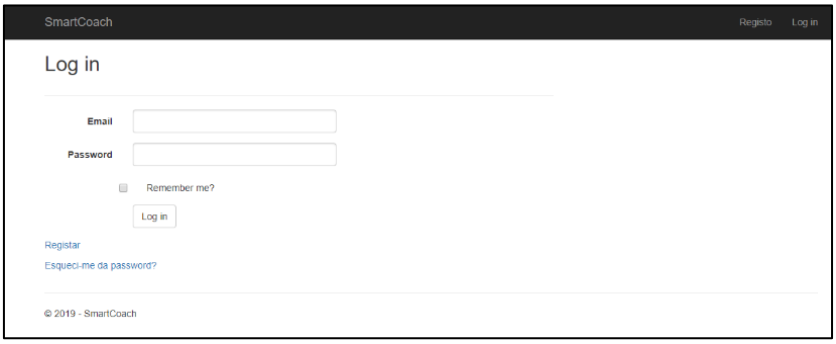


Figura 53 - Formulário de início de sessão

Após o utilizador inserir os seus dados, é invocada a aplicação servidora, esta tem a responsabilidade de validar a informação inserida e retornar um *Token* para o acesso futuro, por parte do utilizador *loggado*. É de salientar que o *Token* a ser gerado tem 14 dias para expirar, no entanto, esta e outras configurações relacionados com o *Token* podem ser facilmente atualizadas no método "*ConfigureAuth*" da classe *Startup.cs*, como apresentado de seguida, na Figura 54.

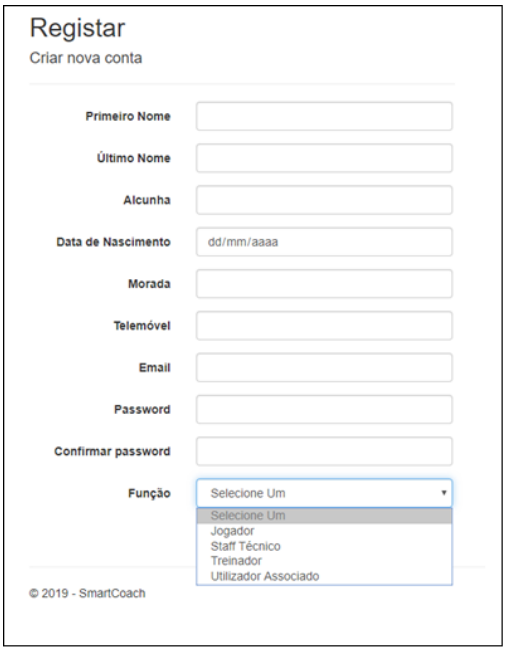
```
// Configure the application for OAuth based flow
PublicClientId = "self";
OAuthOptions = new OAuthAuthorizationServerOptions
{
    TokenEndpointPath = new PathString("/Token"),
    Provider = new ApplicationOAuthProvider(PublicClientId),
    AuthorizeEndpointPath = new PathString("/api/Account/ExternalLogin"),
    AccessTokenExpireTimeSpan = TimeSpan.FromDays(14),
    // In production mode set AllowInsecureHttp = false
    AllowInsecureHttp = true
};

// Enable the application to use bearer tokens to authenticate users
app.UseOAuthBearerTokens(OAuthOptions);
```

Figura 54 - Excerto de código referente à configuração do Token

Retornados os dados relativos ao *Token* gerado, torna-se necessário guardar o mesmo na variável de sessão da *SmartCoachApp* e atualizar os *Claims* deste componente, esta operação está detalhada na secção **5.3.1 - Autenticação e Autorização**. Por fim, caso o sistema não imita nenhuma mensagem de erro, pode-se dar o login como realizado.

5.4.2 Registo

O presente caso de uso é invocado por um utilizador não autenticado quando este pretende criar uma conta no sistema, de modo a possibilitar a realização desta ação foi produzido um formulário dinâmico, que opera consoante o Role/Função escolhido, como é ilustrado nas Figura 55 e Figura 56, ao analisar as mesmas, é possível verificar os novos campos a serem preenchidos quando o utilizador insere a sua função.



Formulário Geral

Formulário Jogador

Figura 55 - Formulário Registo (1)

Figura 56 - Formulário Registo (2)

Após a inserção do formulário acima, é invocado o método *Register()* da *SmartCoachAPI* com o DTO relativo aos dados inseridos. Esta função, primeiramente irá validar os dados recebidos, e de seguida, efetuar a criação da entidade *user*, após isso, caso não haja nenhum erro, será então processada a função inserida pelo utilizador através de um "*switch*". Este comando, tendo em conta o role existente no DTO irá criar uma nova instância de um dos quatro perfis de utilizador existentes (Jogador, Treinador, Staff Técnico ou Utilizador Associado), contudo, caso tratar-se de um Treinador, será também registada uma nova instância da entidade Equipa, assim como, os seus valores referência (por *default*) associados a esta Equipa. Por fim, serão então persistidos os objetos criados na base de dados, seguindo-se da invocação do *EmailService*, este terá o objetivo de informar o sucesso da operação, enviando um email de boas-vindas para o respetivo utilizador. Para complementar a descrição deste caso de uso, será exposta a função *Register()*, no entanto, pode-se verificar que foi ocultado o comando "*switch",* isto devido ao seu elevado comprimento.

```
public async Task<IHttpActionResult> Register(RegistoDTO model)
{
    if (!ModelState.IsValid)  {
        return BadRequest(ModelState);
    }

    var user = new ApplicationUser() {PrimeiroNome = model.PrimeiroNome, UltimoNome = model.UltimoNome, Alcunha = model.Alcunha,
        DataNascimento = model.DataNascimento, Morada = model.Morada, Telemovel = model.Telemovel, UserName = model.Email,
        Email = model.Email, DataRegisto = DateTime.Now};

    IdentityResult result = await UserManager.CreateAsync(user, model.Password);
    ErroDTO erroDTO = new ErroDTO() { Erros = result.Errors };


    if (!result.Succeeded) {
        var errorJson = JsonConvert.SerializeObject(erroDTO);
        //return BadRequest(errorJson);
        var resp = new HttpResponseMessage(HttpStatusCode.BadRequest)  {
            Content = new StringContent(errorJson),
            ReasonPhrase = "error"
        };
        throw new HttpResponseException(resp);
    }


    switch (model.Role)...
    EmailService.SendEmail("Bem-vindo", user.Email, "Olá "+user.PrimeiroNome + ", seja bem-vindo ao software SmartCoach!");
    return Ok();
}
```

Figura 57 - Método *Register()* da componente *SmartCoachAPI*

5.4.3 Gerir conta

A funcionalidade gerir conta está presente para todos os utilizadores do software através do acesso à sua página pessoal, esta divide-se em quatro subsecções, nomeadamente, configuração/atualização do seu perfil, opções associadas à equipa (como abandonar ou juntar-se a uma nova equipa), atualizar a sua password e por fim, apagar a sua conta. De seguida, na Figura 58 será ilustrada a interface relativa a este caso de uso.

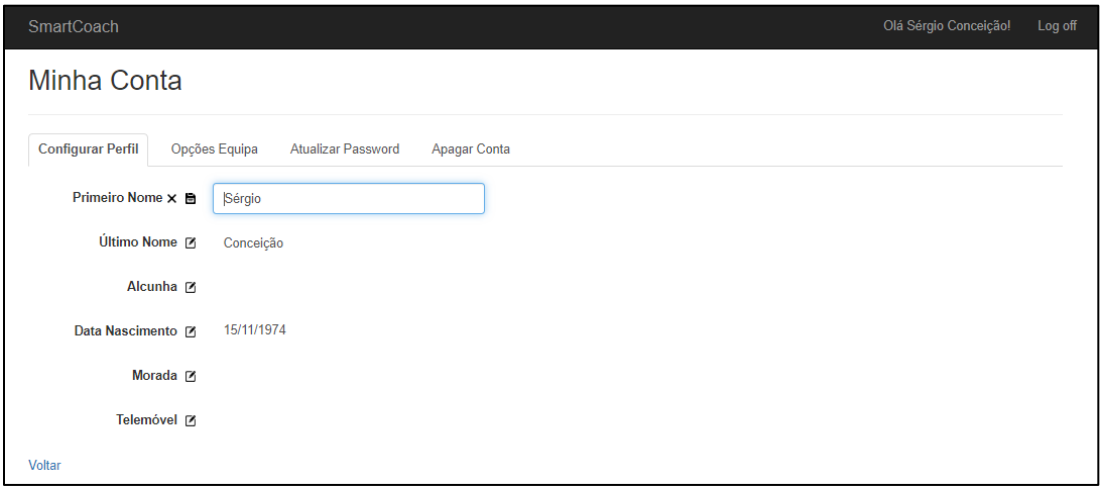


Figura 58 - Funcionalidade de gerir conta

Para a implementação deste caso de uso, foi primeiramente necessário preparar o layout da página "Minha Conta" para apresentar a informação do utilizador, assim como, exibir os separadores e o conteúdo das restantes subfuncionalidades deste caso de uso. Assim sendo, foi utilizado o plugin *pretty-tabs*, como mencionado na secção **5.3.3 - Plugins utilizados** para obter o resultado desejado. De seguida, será detalhada o desenvolvimento de cada uma das subfuncionalidades.

- **Configurar perfil**: De modo a facilitar a atualização de qualquer um dos campos relativos aos dados do utilizador, foi desenvolvido um pedido *AJAX* assíncrono que invoca o *ManageController* da aplicação *web* com o campo a atualizar e o seu novo valor, este após construir o *DTO* requerido tem a finalidade de via *AccountHandler* chamar a aplicação servidora, esta primeiramente validará a existência do utilizador, e de seguida, persistirá os novos dados na *SmartCoachDB,* como apresentado no excerto de código existente na Figura 59;
- **Opções equipa**: Neste separador o utilizador poderá abandonar a sua equipa corrente ao selecionar a respetiva opção, que irá invocar a aplicação servidora e remover a entidade equipa do utilizador corrente, ou enviar um pedido para se juntar a uma nova equipa, que através da inserção do código da equipa desejada, invoca a aplicação servidora que irá adicionar à lista de pedidos pendentes para a equipa relativa ao código inserido o respetivo utilizador;

- **Atualizar password**: Mediante a inserção da password antiga e uma nova password, será invocada a aplicação servidora, esta terá o objeito de primeiro validar os campos, e de seguida, atualizar a password do utilizador;
- **Apagar conta**: Ao selecionar esta ação, será requerida uma confirmação, assim, após a confirmação por parte do utilizador, será então invocada a *SmartCoachAPI*, esta terá o objetivo de remover os dados pessoais do utilizador, como o seu nome, email e data de nascimento, associadas à entidade utilizador e serão inseridos novos dados gerados para esses campos, por último será ainda atualizado o estado do utilizador para "Inativo".

```
0 references | David Abreu, 134 days ago | 1 author, 1 change
public async Task<IHttpActionResult> AtualizarMinhaConta(UpdateMinhaContaDTO minhaContaDTO)
{
    ApplicationUser user = await unitOfWork.UserRepository.GetUserByID(minhaContaDTO.UserID);
    if (user == null){
        return BadRequest();
    }
    List<Posicao> allPosicoes = await unitOfWork.PosicaoRepository.GetPosicoes();
    user.atualizarDados(minhaContaDTO, allPosicoes);
    return Ok(unitOfWork.UserRepository.UpdateUser(user));
}
```

Figura 59 - Método relativo à atualização dos dados do utilizador na *SmartCoachAPI*

5.4.4 Configurar equipa

A configuração da equipa é uma funcionalidade relativa à atualização dos dados da equipa e os mesmos só poderão ser atualizados pelo treinador. Por outras palavras, a página alusiva a esta funcionalidade permitirá alterar os dados da equipa, tais como, o nome, a abreviatura, o escalão, etc. E para além disso, será a partir desta página que o treinador poderá gerir e visualizar os detalhes dos utilizadores associados à equipa (esta operação será detalhada na secção **5.4.6 - Gestão de jogadores/membros do Staff Técnico**), como exibido na Figura 60. É ainda de salientar que os outros membros da equipa também possuirão acesso a esta página, mas não poderão alterar os dados apresentados.

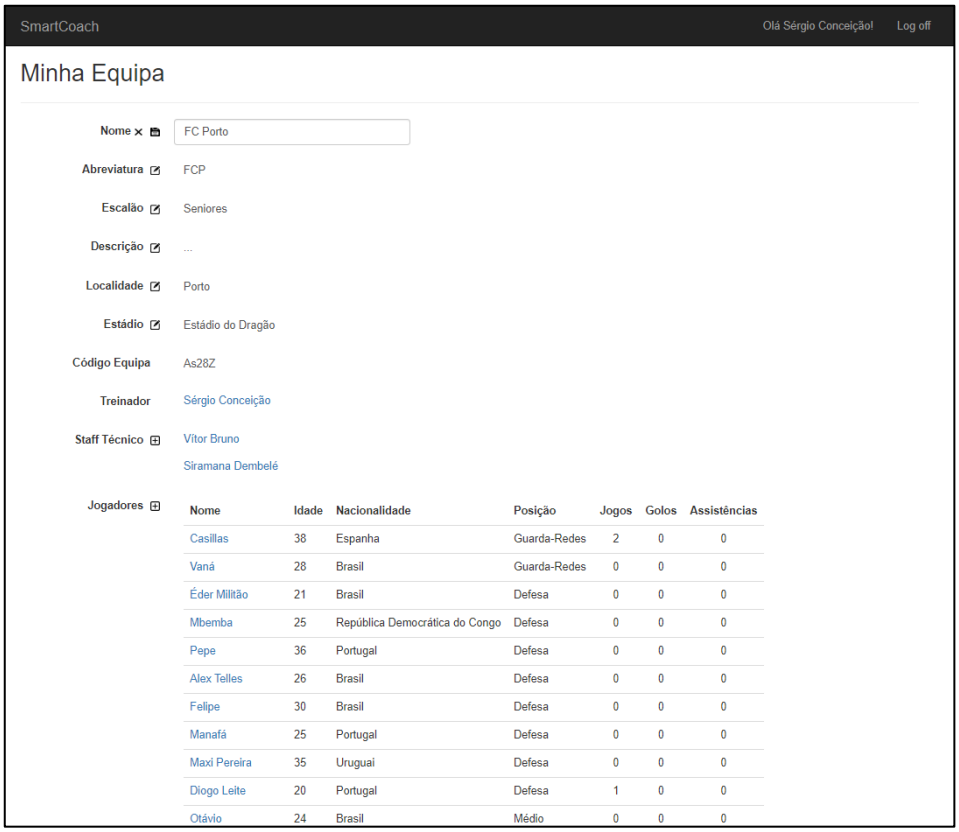


Figura 60 - Funcionalidade de configurar equipa

Para a implementação deste caso de uso, foi seguida a mesma abordagem que no caso de uso de gerir conta, mais especificamente, na secção configurar perfil. Desse modo, foi produzida a *view* com os dados da equipa do utilizador corrente, estes dados podem ser atualizados assincronamente (caso o utilizador seja o treinador da equipa), através da invocação do método *AtualizarEquipa*() da *SmartCoachApp*, que recebe como argumentos o ID da equipa, o ID do campo a atualizar e o novo valor. Esta função tem o objetivo de construir o DTO, para envio ao *controller* da aplicação servidora mediante o componente *EquipaHandler*, como ilustrado na Figura 61. Chegada ao *controller* da *SmartCoachAPI*, este validará o objeto recebido e de seguida, persistirá os novos dados na *SmartCoachDB.*

```
0 references | David Abreu, 17 days ago | 1 author, 3 changes
public async Task<ActionResult> AtualizarEquipa(int equipaID, string dados, int campoID){
    if (!SmartCoachAPIClient.verifySession()) {
        TempData["error"] = "A sua sessão expirou, por favor, faça login novamente!";
        return RedirectToAction("Erro", "Home");
    }
    var objDTO = new {
        EquipaID = equipaID,
        Dados = dados,
        CampoID = campoID
    };
    if (await EquipaHandler.AtualizarEquipa(objDTO)) {
        return Json(true);
    }
    return Json(false);
}
```

Figura 61 - Método da *SmartCoachApp* referente à atualização dos dados da equipa

5.4.5 Consultar calendário desportivo

A presente funcionalidade, tem como objetivo, apresentar para todos os utilizadores uma nova página com a sua agenda desportiva (em formato calendário). No calendário exibido devem existir os eventos relacionados com os jogos da equipa, distinguindo os agendados dos já realizados, contudo, estes últimos serão também diferenciados entre si pelo resultado (derrota, empate ou vitória), sendo a cor o seu elemento de diferenciação. Na Figura 62 será disponibilizado um exemplo de um calendário desportivo. É ainda de salientar que ao selecionar um jogo, será aberto um novo *pop-up* com a informação dos detalhes do jogo, apresentando ao utilizador no final uma série de ações possíveis de realizar, a Figura 63 demonstra um exemplo de um *pop-up* gerado.

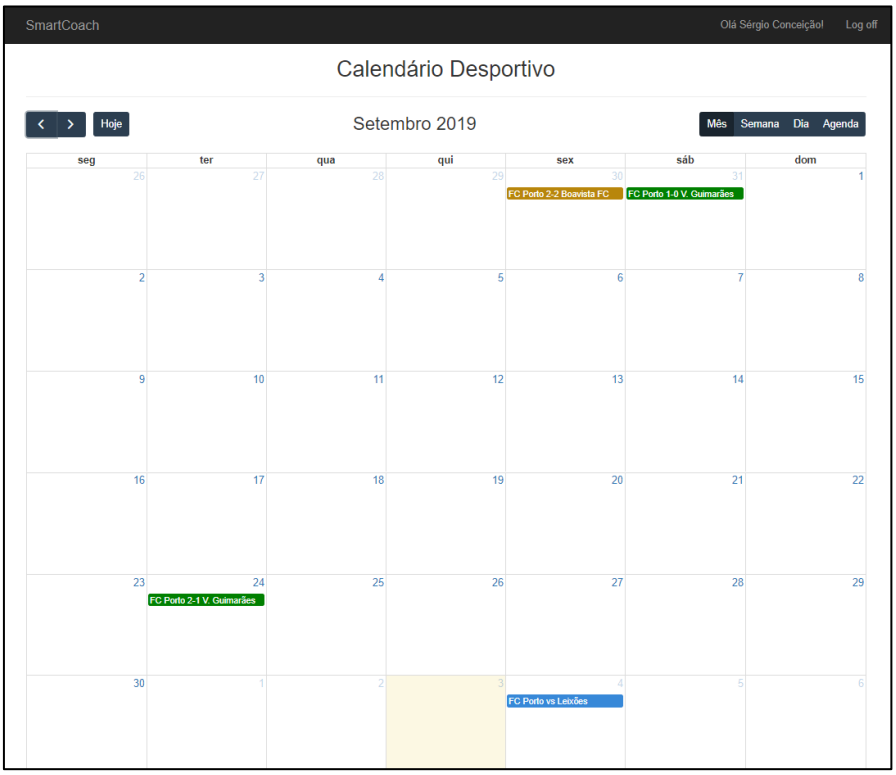


Figura 62 - Calendário desportivo com quatro eventos

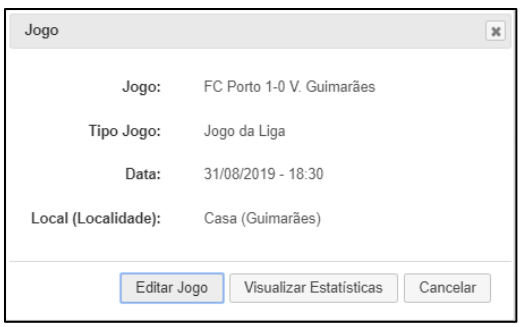


Figura 63 - *Pop-up* com os detalhes do jogo selecionado

Para a apresentação da agenda desportiva em formato calendário foi utilizado o plugin *FullCalendar*, documentado na secção **5.3.3 - Plugins utilizados**, permite a exibição de um calendário, com um conjunto de eventos instanciados por scripts. Desse modo, foi iniciado o plugin com as configurações necessárias. Já em relação aos eventos a serem criados, estes referem-se aos jogos da equipa, portanto, para a criação dos mesmos, foi enviado para a *view* a lista de todos os jogos existentes relativos à equipa do utilizador. De seguida, verifica-se que ao iterar a lista será realizada uma comparação do resultado do jogo, para diferenciar o jogo consoante o resultado do jogo obtido, atribuindo uma cor única para cada uma das possibilidades. Na Figura 64 será demonstrado o excerto de código que tem a responsabilidade da criação dos eventos associados ao calendário.

```
@foreach (var jogo in Model.JogosEquipa){
    <text>
    if (jogo.EstadoJogo == SmartCoachApp.Models.EstadoJogo.ConcluidoComResultado){
        if (jogo.ResultadoJogo.GetResultado() == SmartCoachApp.Models.Resultado.Derrota){
            var event = {
                id: '@Html.Raw(jogo.ID)',
                title: '@Html.Raw(jogo.GetNomeSimplesWithResult())',
                start: '@Html.Raw(jogo.GetDataInicio())',
                end: '@Html.Raw(jogo.GetDataFim())',
                color: 'red'
            };
            calendar.addEvent(event);
        } else if (jogo.ResultadoJogo.GetResultado() == SmartCoachApp.Models.Resultado.Vitoria) {
            var event = {
                id: '@Html.Raw(jogo.ID)',
                title: '@Html.Raw(jogo.GetNomeSimplesWithResult())',
                start: '@Html.Raw(jogo.GetDataInicio())',
                end: '@Html.Raw(jogo.GetDataFim())',
                color: 'green'
            };
            calendar.addEvent(event);
        }else{
            var event = {
                id: '@Html.Raw(jogo.ID)',
                title: '@Html.Raw(jogo.GetNomeSimplesWithResult())',
                start: '@Html.Raw(jogo.GetDataInicio())',
                end: '@Html.Raw(jogo.GetDataFim())',
                color: 'darkgoldenrod'
            };
            calendar.addEvent(event);
        }
    } else {
        var event = {
            id: '@Html.Raw(jogo.ID)',
            title: '@Html.Raw(jogo.GetNomeSimples())',
            start: '@Html.Raw(jogo.GetDataInicio())',
            end: '@Html.Raw(jogo.GetDataFim())'
        };
        calendar.addEvent(event);
    }
</text>
}
calendar.render();
```

Figura 64 - Código referente à inicialização dos eventos/jogos do calendário

5.4.6 Gestão de jogadores/membros do Staff Técnico

De seguida, devido à semelhança entre os casos de uso de gestão de jogadores e da gestão de membros do staff técnico estes serão unidos numa única secção, esta união tem o objetivo de não repetir informação desnecessária. Assim, pode-se dizer que estas funcionalidades estão englobadas na configuração da equipa, e dividem-se em três subfuncionalidades unicamente executadas pelo treinador, nomeadamente:

- **Convidar utilizadores**: O treinador poderá convidar novos jogadores/membros Staff através da inserção do email do utilizador desejado, é de destacar que o utilizador terá que existir no sistema para o convite ser enviado.

- **Remover utilizador**: O treinador poderá remover jogadores/membros staff técnico existentes na equipa, ao selecionar o utilizador desejado, e confirmando a sua remoção, como ilustrado na Figura 65 para o caso de um jogador;

Figura 65 - Confirmação da remoção de um jogador

- **Aceitar/Rejeitar utilizadores pendentes**: Por último, o treinador poderá aceitar ou rejeitar convites de adesão à sua equipa, de seguida, a Figura 66 a lista de convites enviados e recebidos.

Figura 66 - Gerir convites de adesão de jogadores

O presente caso de uso encontra-se analisado na secção **4.3.5.1 - TR04 – Gestão de Jogadores**, no entanto, de modo a complementar essa analise, será explicado de forma mais pormenorizada a implementação de cada uma das três subfuncionalidades inerentes a este caso de uso:

Convidar utilizadores: Para a implementação desta atividade, foi gerado no ficheiro relativo ao *layout* da página, um *pop-up* que solicita o email do utilizador a ser convidado, a criação do mesmo baseou-se no código disponibilizado pelo *jQuery Dialog* [67]. De seguida, é invocado via *Ajax* a aplicação *web*, que comunica por sua vez com a aplicação servidora, sendo esta responsável, por verificar a existência do email introduzindo e realizar a lógica necessária

para convidar o utilizador. Após isso, é retornado o resultado desta operação para a *view* que informa o sucesso ou insucesso da operação, de seguida a função *convidarJogadorEquipa*(), presente na Figura 67, mostra a componente da *view* responsável pela criação do *pop-up*, assim como, o envio e receção dos dados para o sistema.

```
function convidarJogadorEquipa() {
    var mensagem = $('<div id="conformBox" class="form-horizontal" style="overflow:hidden">'
        + "<div class='form-group' style='margin-top: 15px;'><label class='control-label col-md-2'>Email: </label>"
        + "<div class='col-md-10'><input id='inputEmail' type='text' class='form-control'></input></div></div>" + '</div>');
    mensagem.dialog({
        title: "Convidar Jogador",
        closeOnEscape: true,
        close: function (event, ui) {
            $(this).dialog('destroy').remove();
        },
        width: 400,
        modal: true,
        buttons: [{
            text: "Convidar",
            click: function () {
                if ($("#inputEmail").val().trim().length > 0 && isEmail($("#inputEmail").val().trim())) {
                    var data = {
                        email: $("#inputEmail").val().trim(),
                    };
                    var exists = checkEmailExists("../Manage/CheckJogadorHasEmail", data);
                    if (exists != null) {
                        if (exists) {
                            var data = {
                                equipaID: @Model.ID,
                                emailJog: $("#inputEmail").val().trim(),
                            };
                            var check = saveDataAjax("../Manage/ConvidarJogadorEquipa", data);
                            if (check) {
                                alert("Convite enviado, para mais detalhes visite o seu perfil!");
                                $(this).dialog("close");
                            } else {
                                alert("Erro, por favor tente mais tarde!");
                                $(this).dialog("close");
                            }
                        } else {
                            alert("Email inexistente no sistema, por favor, insira um Email válido!");
                            $("#inputEmail").focus();
                        }
                    } else {
                        alert("Erro, por favor tente mais tarde!");
                    }
                } else {
                    alert("Insira um Email válido!");
                    $("#inputEmail").focus();
                }
            }
        }, {
            text: "Cancelar",
            click: function () {
                $(this).dialog("close");
            }
        }]
    });
}
```

Figura 67 - Script responsável pela criação do *pop-up* e invocação da aplicação *web*

Remover utilizador: A remoção de um jogador da equipa, é realizada após a sua confirmação, para isso, foi gerado um *pop-up* de confirmação (como detalhado acima), que ao ser confirmado pelo treinador, invoca a aplicação servidora, de modo, a remover o respetivo utilizador da equipa. Para obter esse resultado, será atualizado a equipa do utilizador para o valor "*null*". A Figura 68, apresenta o método responsável por esta ação.

```
0 references | David Abreu, 115 days ago | 1 author, 1 change
public async Task<IHttpActionResult> RemoverUtilizadorEquipa(RemoverUserDTO objDTO) {
    ApplicationUser user = await unitOfWork.UserRepository.GetUserByID(objDTO.UserID);
    if (user == null){
        return BadRequest();
    }
    if (user.Jogador != null) {
        if (user.Jogador.EquipaID == null){
            return BadRequest();
        } else{
            user.Jogador.EquipaID = null;
            unitOfWork.JogoRepository.UpdateInformacaoJogoByJogadorID(user.Id);
            return Ok(unitOfWork.UserRepository.UpdateUser(user));
        }
    }
    if (user.StaffTecnico != null) {
        if (user.StaffTecnico.EquipaID == null){
            return BadRequest();
        } else{
            user.StaffTecnico.EquipaID = null;
            return Ok(unitOfWork.UserRepository.UpdateUser(user));
        }
    }
    return BadRequest();
}
```

Figura 68 - Método da aplicação servidora responsável pela remoção do utilizador

Aceitar/Rejeitar utilizadores pendente: Para a aceitação/rejeição de convites pendentes, foi implementado um mecanismo simples de redireccionamento para o método responsável por aceitar ou rejeitar o utilizador pretendido na aplicação servidora, este método, de uma forma simplificada, adiciona o jogador na equipa do treinador ou remove o convite de adesão. Atualizando de seguida a página, com as alterações realizadas.

5.4.7 Gestão de jogos

A componente de gestão de jogos, será operada pelo treinador e/ou membros do staff técnico e divide-se em três categorias, nomeadamente: Apresentação dos jogos da equipa e seus detalhes, diferenciando os jogos agendados dos já concluídos, como apresentado na Figura 69. Atualização de dados, como a inserção do resultado, para um certo jogo, como exibido na Figura 70. E por último, registo de novos jogos, como exemplificado na Figura 71.

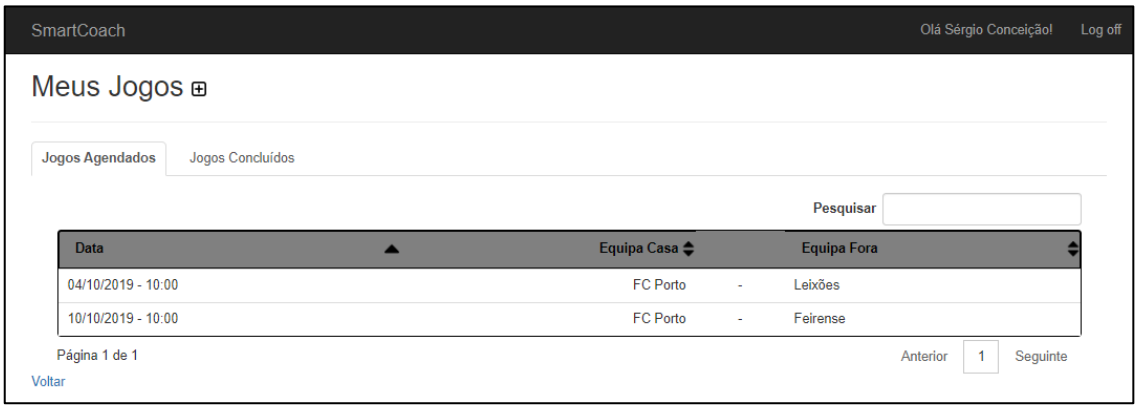


Figura 69 - Jogos da equipa divididos consoante o seu estado (concluído, agendado)

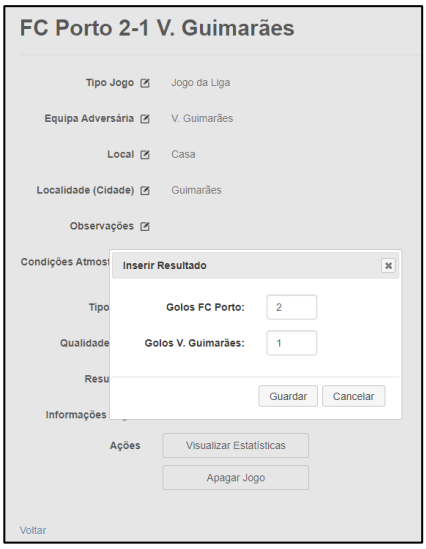


Figura 70 - Atualização de um jogo

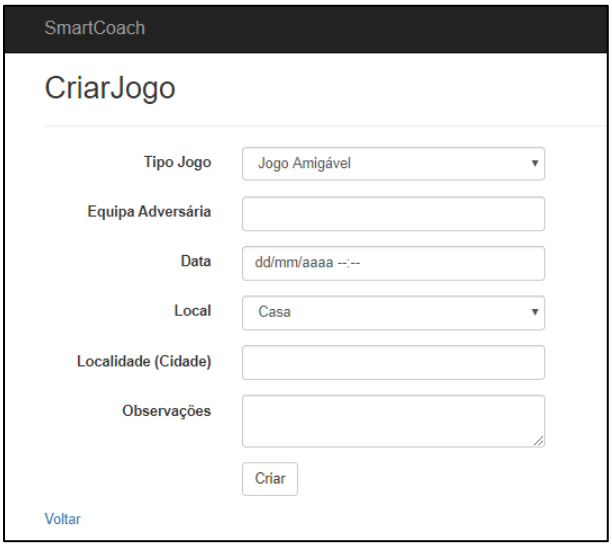


Figura 71 - Exemplo do registo de um novo jogo

Para a implementação deste caso de uso, mais concretamente, para a funcionalidade de visualizar a lista de jogos existentes da sua equipa, foi primeiramente invocada a função de AtualizarEstadoJogos() da *SmartCoachAPI*, este método, como o nome indica, tem o propósito de atualizar automaticamente o estado de cada jogo, consoante a sua data e o resultado (se o mesmo tiver sido inserido), a Figura 25, explicada anteriormente apresenta de forma mais percetível o diagrama de estados da entidade jogo. Após isso, torna-se necessário preparar a *view* para apresentar os jogos da equipa do utilizador, esta representação teve a assistência dos plugins *DataTables* e *Pretty-Tabs*, documentados na secção **5.3.3 - Plugins utilizados**. Estes plugins foram aplicados através de código HTML simples associado a scripts jQuery disponibilizados pelos respetivos softwares.

De seguida, para a atualização dos dados de um certo jogo, foi seguida a mesma abordagem usada nos UCs "Gerir Conta" e "Configurar Equipa", no entanto, para este caso foi necessário ter em atenção que é requerido a inserção do resultado do jogo, assim sendo, foi produzido um *pop-up* que solicita os golos marcados de ambas as equipa, e ao confirmar o *pop-up* é invocado o *controller* da aplicação *web* que por sua vez comunica com a *SmartCoachAPI*, que tem o objetivo de guardar os dados inseridos e atualizar a entidade jogo, como demonstrado no método *GuardarResultado*() presente na Figura 72.

```
0 references | David Abreu, 18 days ago | 1 author, 2 changes
public async Task<IHttpActionResult> GuardarResultado(GuardarResultadoDTO jogoDTO)
{
    Jogo jogo = await unitOfWork.JogoRepository.GetJogoByID(jogoDTO.JogoID);
    if (jogo == null) {
        return BadRequest();
    }
    jogo.guardarResultado(jogoDTO);
    return Ok(unitOfWork.JogoRepository.UpdateJogo(jogo));
}
```

Figura 72 - Método responsável por atualizar o resultado de um jogo

Por fim, já em relação à possibilidade do utilizador quer registar um novo jogo, foi desenvolvida um novo *layout* que contém o formulário necessário à criação do jogo como exibido na Figura 71, após preenchimento do formulário e respetiva submissão, a aplicação *web* comunica com a aplicação servidora invocando o método CriarJogo(), este recebe a entidade jogo por parâmetro, que é validada e retificada, de modo a adicionar o ID da equipa do utilizador corrente, para o objeto "jogo" ser posteriormente guardado na base de dados, a Figura 73 expõe a função mencionada.

```
[HttpPost]
[Authorize(Roles = "Treinador, Staff Técnico, Jogador Individual")]
[ResponseType(typeof(JogoDTO))]
0 references | David Abreu, 18 days ago | 1 author, 3 changes
public async Task<IHttpActionResult> CriarJogo(Jogo jogo)
{
    if (!ModelState.IsValid) {
        return BadRequest(ModelState);
    }
    try {
        Equipa equipa = await GetMyEquipaByUser();
        jogo.EquipaID = equipa.ID;
        unitOfWork.JogoRepository.CreateJogo(jogo);
        return Ok(JogosTypeAdapter.ToDTO(jogo));
    }catch {
        return BadRequest();
    }
}
```

Figura 73 - Método da *SmartCoachAPI* responsável pelo registo de um novo jogo

5.4.8 Configurar valores referência

Este caso de uso, refere-se à configuração de valores referência (estes foram gerados aquando do registo da equipa), por parte do treinador, estes valores têm relevância na análise de recomendação, uma vez que são valores, de acordo com o que o treinador pretende da performance da sua equipa. Mais concretamente, estes valores estão relacionados com cada uma das estatísticas a recolher e existem para todas as possíveis posições do jogador. A Figura 74 apresenta a tabela dos valores referência para uma certa equipa, nesta é possível verificar para cada estatística o valor médio de referência, assim como ao visualizar os detalhes de uma certa estatística, o seus valores específicos, de acordo com cada posição, para além disso, a tabela exibida permite a fácil atualização de um ou mais valores referência, ou pela alteração do valor/operador, ou pela inativação/ativação do respetivo valor referência.

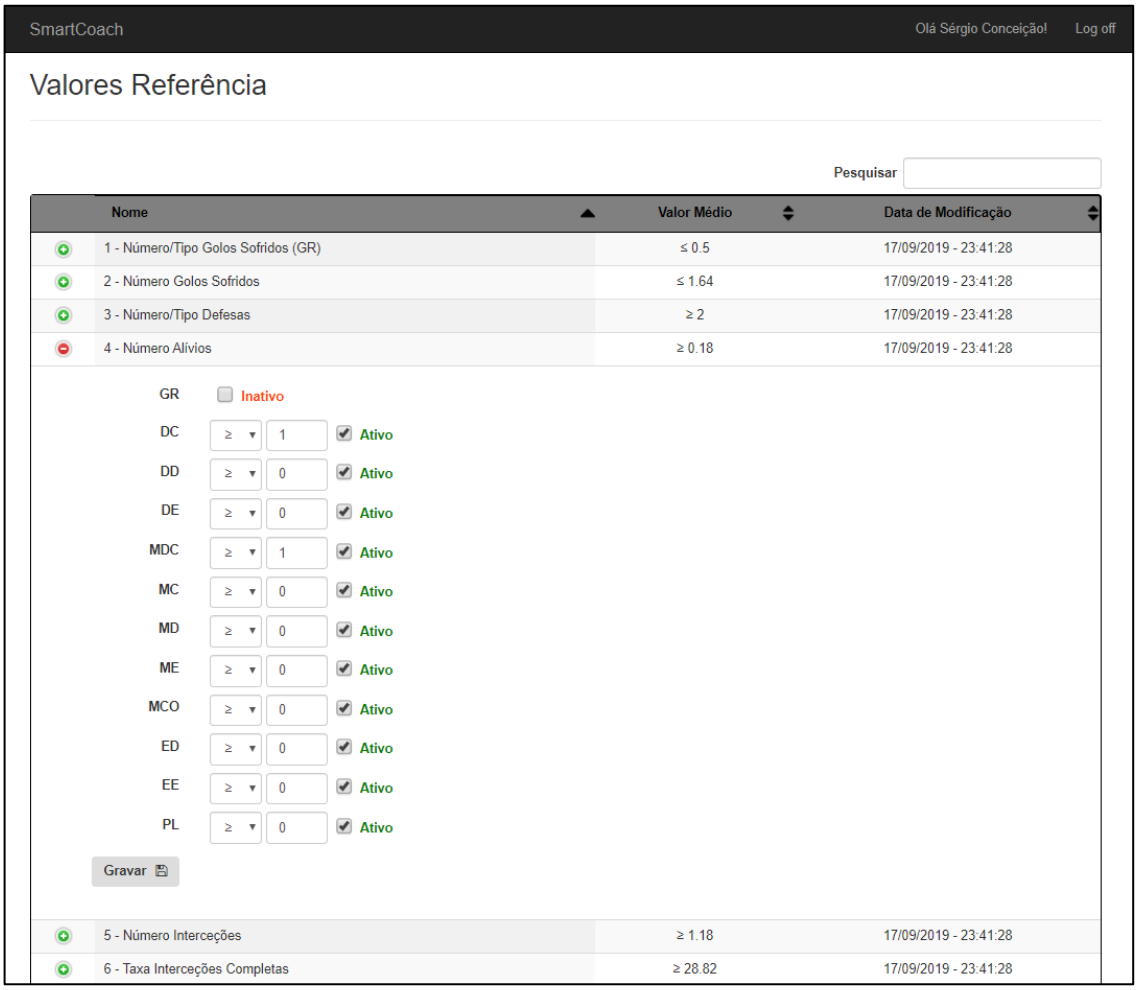


Figura 74 - Tabela dos valores referência com detalhe do número de alívios por posição

Para a realização deste caso de uso, foi primeiramente necessário obter através do método "*GetValoresReferenciaByEquipaID*()" da camada *repositories* da aplicação servidora, todos os valores referência associados à equipa do treinador, assim como suas dependências, esta operação é realizada de modo assíncrono e pode ser consultada na Figura 75.

```
2 references | David Abreu, 39 days ago | 1 author, 1 change
public async Task<List<ValorReferencia>> GetValoresReferenciaByEquipaID(int equipaID)
{
    return await context.ValoresReferencia.Where(c => c.EquipaID == equipaID)
        .Include(x=>x.EstatisticaParaPerformance.DetalhesEstatisticaPerformance)
        .Include(x=>x.ValoresReferenciaPorTipo.Select(y=>y.DetalheEstatisticaPerformance))
        .Include(x => x.Posicao).ToListAsync();
}
```

Figura 75 - Método responsável pela obtenção dos valores referência de uma equipa

De seguida, após a existência dos dados, foi trabalhada a apresentação destes valores, através do plugin *DataTables*. Como resultado, foi obtido uma tabela inteligente que apresenta para as estatísticas existentes o resumo dos valores referência associados, como, o seu valor médio (calculado através da realização da média dos valores referência ativos para a respetiva estatística) e a data da última modificação. Para além disso, foi implementado um mecanismo que ao selecionar uma estatística invocará o *controller,* de modo a ir buscar os detalhes dos valores referência relacionados com cada uma das posições existentes para a estatística selecionada. Após ser exibido na *view* estes detalhes (em formato editável), é dado a possibilidade de o treinador atualizar estes valores. Depois do treinador realizar as alterações desejadas deve selecionar o botão "Guardar", este quando pressionado invoca a *SmartCoachAPI*, que irá receber e iterar a lista de valores referência associados aos detalhes da estatística selecionada, atualizando os novos valores na base de dados. Por fim, a Figura 76, apresenta o código relativo à atualização dos valores referência por parte da *SmartCoachAPI*.

```
0 references | David Abreu, 18 days ago | 1 author, 2 changes
public async Task<IHttpActionResult> AtualizarValoresReferencia(List<ValorReferenciaDTO> objDTOs)
{
    Equipa equipa = null;
    var userID = User.Identity.GetUserId();
    if (await unitOfWork.RoleRepository.CheckUserHasRole(userID, "Jogador Individual")) {
        equipa = await unitOfWork.EquipaRepository.GetEquipaByJogadorID(userID);
    } else if (await unitOfWork.RoleRepository.CheckUserHasRole(userID, "Treinador"))  {
        equipa = await unitOfWork.EquipaRepository.GetEquipaByTreinadorID(userID);
    }
    if (equipa == null){
        return BadRequest();
    }
    foreach (var vrDTO in objDTOs) {
        ValorReferencia valorReferencia = await unitOfWork.ValoresReferenciaRepository.GetValorReferenciaByID(vrDTO.ID);
        if (valorReferencia.EquipaID != equipa.ID){
            return BadRequest();
        }
        valorReferencia.EstadoReferencia = vrDTO.EstadoReferencia;
        valorReferencia.Valor = vrDTO.Valor;
        valorReferencia.OperacaoValor = vrDTO.OperacaoValor;
        valorReferencia.DataAtualizacao = DateTime.Now;
        if (vrDTO.ValoresReferenciaPorTipo != null){
            foreach (var vrptDTO in vrDTO.ValoresReferenciaPorTipo) {
                var vrpt = valorReferencia.ValoresReferenciaPorTipo.FirstOrDefault(x=>x.ID == vrptDTO.ID);
                vrpt.Valor = vrptDTO.Valor;
            }
        }
        if (!unitOfWork.ValoresReferenciaRepository.UpdateValorReferencia(valorReferencia)){
            return BadRequest();
        }
    }
    return Ok();
}
```

Figura 76 - Método de atualização de um conjunto de valores referencia

5.4.9 Inserir estatísticas jogo

O objetivo desta funcionalidade é permitir a recolha de estatística do jogo, ao vivo ou pós jogo, por parte do staff da equipa, ou então utilizadores associados aos jogadores em campo. Assim, para executar esta funcionalidade, será necessário primeiro introduzir os jogadores, que o são pretendidos levantar estatística, como demonstrado na Figura 77. E de seguida, será disponibilizado um painel de recolha dos dados do jogo, este irá conter vários elementos, entre eles: os dados do jogo, um cronometro para iniciar o tempo do jogo, um campo para determinar, opcionalmente, onde ocorreu a ação, e por último, separadores para os jogadores selecionados com a lista de estatísticas a recolher, divididas em atributos primários e secundários, consoante a posição que o jogador está a jogar. A Figura 78, apresenta o painel descrito.

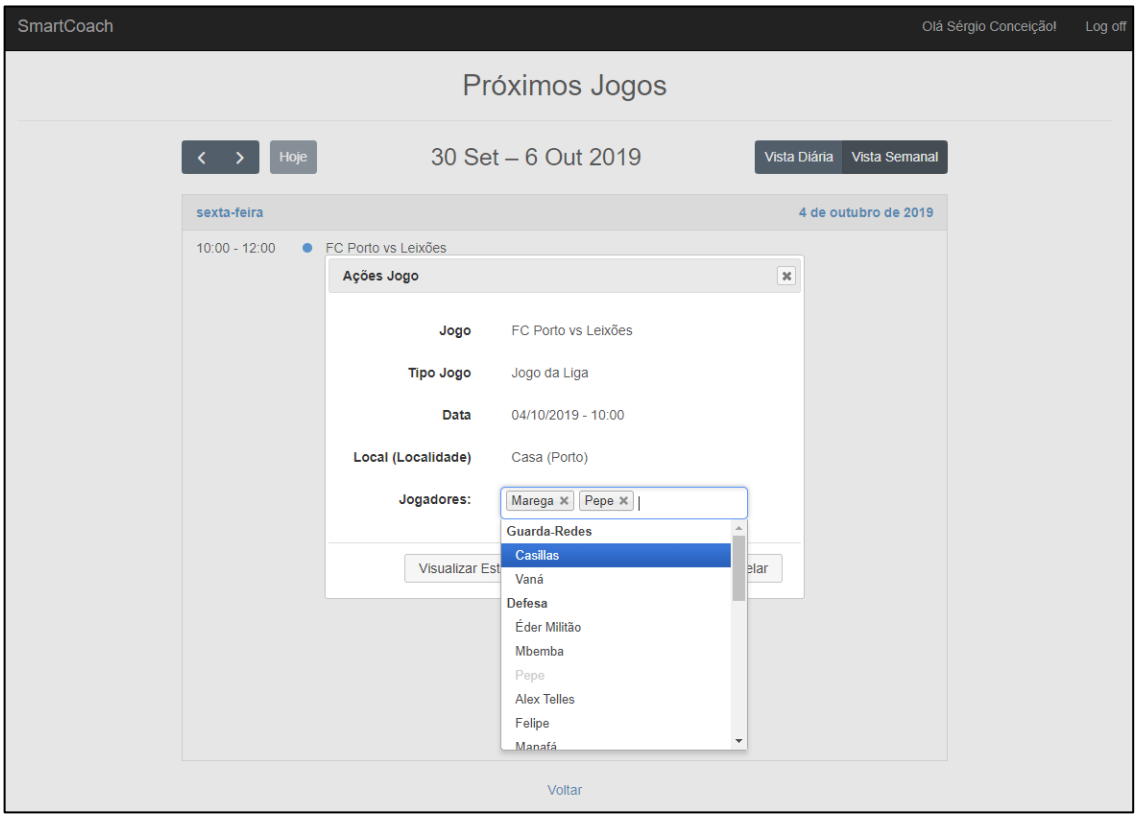


Figura 77 - Seleção dos jogadores para levantamento de estatísticas

Figura 78 - Painel de recolha de estatísticas

O desenvolvimento deste caso de uso, está relacionado com a produção de duas componentes. A primeira, que diz respeito à criação de um *pop-up* para a seleção dos jogadores aos quais o utilizador quer inserir estatística. E a segunda, destinada à recolha dos dados estatísticos do jogo, através de um painel intuitivo, como apresentado acima.

Assim, primeiramente, foram obtidos os jogadores possíveis de inserir estatísticas, consoante o Role do utilizador (Utilizador Associado ou membro do Staff Técnico), de seguida, com o auxílio de scripts associados ao plugin *jQuery-Chosen*, descrito na secção **5.3.3 - Plugins utilizados**, apresenta no *pop-up* criado um elemento de seleção que contém a lista dos jogadores possíveis de selecionar (estes, ordenados pela sua posição). Após isso, e de acordo com o diagrama de sequência analisado na secção **4.3.5.3 - UA01 – Inserir estatísticas de jogo**, é possível verificar que o sistema procede à inicialização da *view* invocando a *SmartCoachAPI*, esta tem a responsabilidade de verificar se o utilizador já inseriu dados do respetivo jogador no jogo selecionado e retornar para a *view* os valores já inseridos para cada uma das estatísticas (ou o valor 0, caso não tenha inserido). De seguida, a uma nova página é criada, com as informações do jogo e os dados retornados. É dada a possibilidade de iniciar o cronômetro do jogo, que através da função *startTimer*(), analisa intervalos de 60 em 60 segundos, atualizando o tempo de jogo, como demonstrado na Figura 79.

```
function startTimer() {
    timer = setInterval(function () {
        if (PrimeiraParteLive) {
            if (min == 45) {
                extraMin++;
                $("#minJogo").text(min + "' +" + extraMin);
            } else {
                min++
                $("#minJogo").text(min + "'");
            }
        } else if (IntervaloLive) {
            $("#minJogo").text("Meio-Tempo");
        } else if (SegundaParteLive) {
            if (min == 90) {
                extraMin++;
                $("#minJogo").text(min + "' +" + extraMin);
            } else {
                min++
                $("#minJogo").text(min + "'");
            }
        } else if (FimJogoLive) {
            $("#minJogo").text("Terminado");
        }

    }, 60000);
}
```

Figura 79 - Função responsável pelo cronômetro do tempo do jogo

Para além disso, o utilizador deverá inserir a posição a que jogador está a jogar (por via de uma caixa de seleção com as posições disponíveis), de modo, a invocar um pedido *Ajax* à *SmartCoachAPI,* com vista a serem apresentadas na *view* as estatísticas para performance que poderão ser recolhidas, dando a opção de incrementar ou desincrementar cada uma das mesmas. Por fim, caso o utilizador incremente uma certa estatística, a *view* realiza um pedido *Ajax ao Controller da App,* com vista a comunicar com a *SmartCoachAPI*, de modo, a atualizar o valor atual e os detalhes do valor estatístico incrementado, tais como, o minuto e a posição do campo onde ocorreu a ação, assim como, a técnica de execução (pé direito, pé esquerdo ou cabeça), como demonstrado na Figura 80, após isso, devem ser persistidas as alterações realizadas na *SmartCoachDatabase,* por parte da camada *Repositories* da aplicação servidora.

```
1 reference | David Abreu, 58 days ago | 1 author, 2 changes
public void incrementarValor(AtualizarValorEstatisticaDTO estatisticasDTO){
    ValorAtual = ValorAtual + 1;
    if (DetalhesValoresEstatisticaJogador == null){
        DetalhesValoresEstatisticaJogador = new List<ValorEstatisticaJogador>();
    }
    int min = -1;
    PosicaoCampo posicaoCampo = PosicaoCampo.NA;
    if (estatisticasDTO.Minutos != null){
        min = estatisticasDTO.Minutos.Value;
    }
    if (estatisticasDTO.PosicaoCampoID != null){
        posicaoCampo = (PosicaoCampo) estatisticasDTO.PosicaoCampoID;
    }
    TecnicaExecucao? tecnica = null;
    if (estatisticasDTO.TecnicaExecucao != null){
        tecnica = (TecnicaExecucao)estatisticasDTO.TecnicaExecucao;
    }
    ValorEstatisticaJogador novoValor = new ValorEstatisticaJogador() {
        DataInsercao = DateTime.Now,
        IndiceInsercao = ValorAtual,
        EstatisticaJogadorID = ID,
        Minuto = min,
        PosicaoCampo = posicaoCampo,
        TecnicaExecucao = tecnica
    };
    DetalhesValoresEstatisticaJogador.Add(novoValor);
}
```

Figura 80 - Método responsável por incrementar uma estatística do jogador

5.4.10 Gerar um treino recomendado

A criação de um treino recomendado trata-se da funcionalidade principal do sistema. A execução da mesma é da responsabilidade do treinador ou de um jogador da aplicação e está dividida em duas etapas, a primeira, que passa pelo preenchimento do formulário para a recomendação de atributos de treino, por parte do sistema de recomendação. E a segunda, relativa à escolha dos atributos de treino (provenientes do mecanismo de recomendação) que quer ver presentes na recomendação da lista de exercícios de treino, como apresentado na Figura 81. Após o utilizador confirmar, será então gerado um treino recomendado, este contém um conjunto de exercícios específicos, para a duração inserida, a Figura 82, ilustra a lista de treinos existentes no sistema, incluindo o que foi recomendado. De seguida, é ainda possível apagar o treino recomendado, ou então, visualizar os detalhes de um exercício específico do treino selecionado, esta ação é realizada com o auxílio da plataforma *SmartCoachManager*, como demonstrado na Figura 83.

Figura 81 - Apresentação da recomendação de atributos de treino por parte do SR

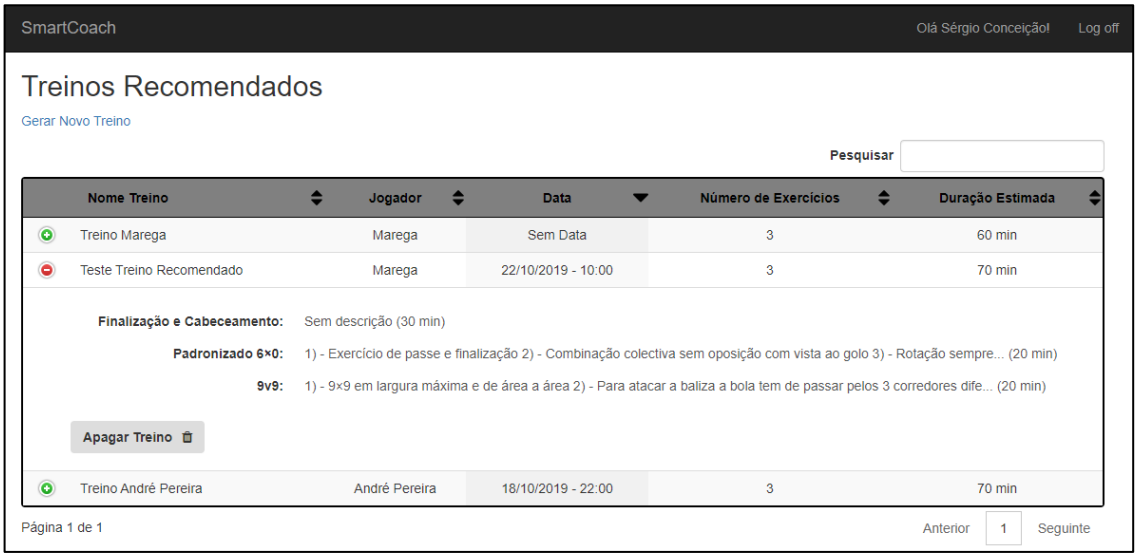


Figura 82 - Treinos recomendados existentes no sistema

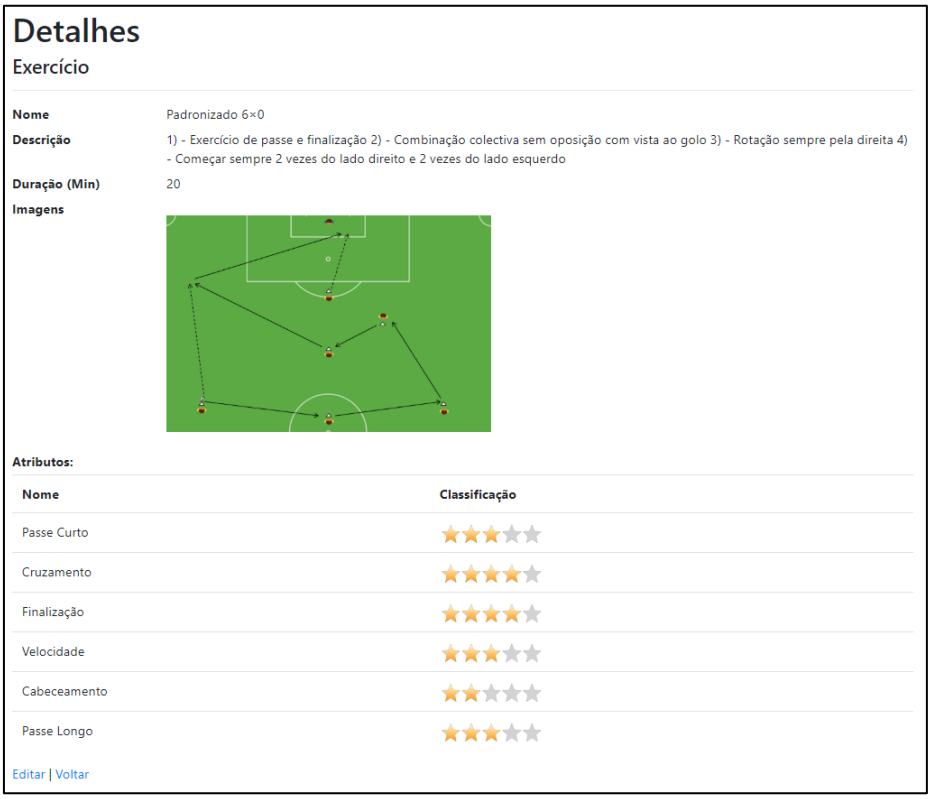


Figura 83 - Detalhes de um exercício de treino na plataforma *SmartCoachManager*

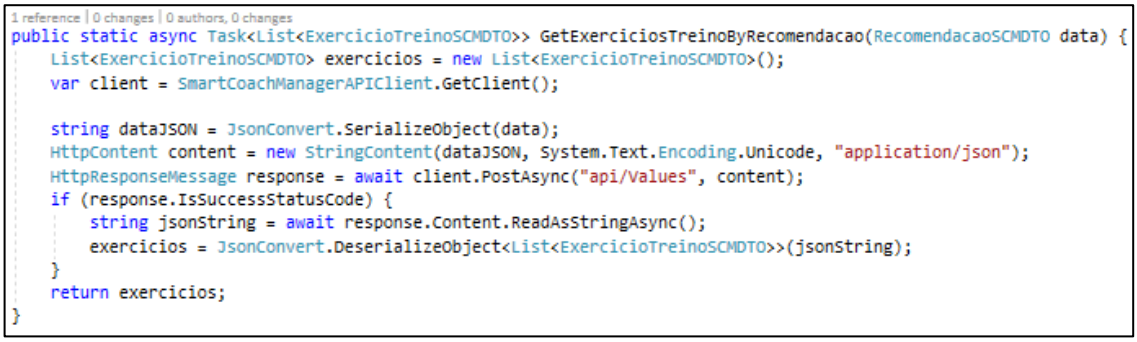
A implementação desta funcionalidade foi previamente detalhada do ponto de vista de design na secção **4.3.5.2 - JG06 – Gerar um plano de treino recomendado**. Tendo isso em conta, ao analisar o diagrama de sequência presente nessa secção, torna-se possível verificar que o caso de uso dá início após o utilizador preencher os dados requeridos à recomendação, assim sendo, em termos de desenvolvimento, foi primeiramente necessário produzir um formulário, que solicite o nome, a data e a duração do treino, bem como, um período temporal (entre duas datas) para análise de estatísticas. Esta última foi alcançada através do uso do plugin *DataRangePicker*, também documentado na secção **5.3.3 - Plugins utilizados.**

De seguida, depois do preenchimento e submissão do formulário apresentado é invocado o sistema de recomendação (SR), pelo *TreinosController* da *SmartCoachAPI*. Este mecanismo tem o objetivo de analisar as estatísticas obtidas do respetivo jogador no período de análise inserido e identificar possíveis atributos de treino a melhorar, como abordado anteriormente no capitulo **5.1 - Sistema de Recomendação**. Após esta análise, é então instanciada uma recomendação, a Figura 84 demonstra o código referente a esta atividade.

```
0 references | David Abreu, 19 days ago | 1 author, 1 change
public async Task<IHttpActionResult> GerarRecomendacaoJogador(GerarTreinoDTO objDTO) {
    List<RecomendacaoDTO> recomendacoesCriadas = new List<RecomendacaoDTO>();
    var userID = User.Identity.GetUserId();
    var jogID = User.Identity.GetUserId();

    Equipa equipa = await GetMyEquipaByUser();
    Jogador jogador = await unitOfWork.JogadorRepository.GetJogadorByID(jogID);
    var est = await unitOfWork.EstatisticasJogoRepository.GetEstatisticasByJogadorIDAndDatePeriod(jogID, objDTO.DataInicioAnalise, objDTO.DataFimAnalise.AddDays(1));
    //Mecanismo de Recomendação
    List<AtributoRecomendacao> result = await Helpers.SistemaRecomendacao.GerarRecomendacao(equipa, jogador, est, unitOfWork);

    Recomendacao r = new Recomendacao();
    r.DataInicioAnalise = objDTO.DataInicioAnalise;
    r.DataFimAnalise = objDTO.DataFimAnalise;
    r.Jogador = await unitOfWork.JogadorRepository.GetJogadorByID(jogID);
    r.JogadorID = jogID;
    r.UserRegisto = await unitOfWork.UserRepository.GetUserByID(userID);
    r.UserRegistoID = userID;
    r.DataRegisto = DateTime.Now;
    r.Nome = objDTO.NomeTreino;
    r.DuracaoTreino = objDTO.DuracaoTreino;
    r.DataTreino = objDTO.DataTreino;
    r.AtributosRecomendacao = result;
    recomendacoesCriadas.Add(RecomendacaoTypeAdapter.ToDTO(r));
    return Ok(recomendacoesCriadas);
}
```

Figura 84 - Método responsável por invocar o SR e instanciar a Recomendação

De seguida, através do desenvolvimento de um *pop-up* da página *web,* é apresentada a recomendação gerada ao jogador, dando a possibilidade a este de selecionar os atributos de treino que quer aperfeiçoar, permitindo assim, a comunicação com a aplicação servidora, que tem o objetivo de invocar o serviço externo *SmartCoachManagerAPI* para obtenção dos exercícios de treino mais aptos para a recomendação gerada, como ilustrado na Figura 85. Após validado pelo sistema, é realizada a persistência do treino na *SmartCoachDatabase* e apresentado ao jogador o respetivo treino criado.



```
1 reference | 0 changes | 0 authors, 0 changes
public static async Task<List<ExercicioTreinoSCMDTO>> GetExerciciosTreinoByRecomendacao(RecomendacaoSCMDTO data) {
    List<ExercicioTreinoSCMDTO> exercicios = new List<ExercicioTreinoSCMDTO>();
    var client = SmartCoachManagerAPIClient.GetClient();

    string dataJSON = JsonConvert.SerializeObject(data);
    HttpContent content = new StringContent(dataJSON, System.Text.Encoding.Unicode, "application/json");
    HttpResponseMessage response = await client.PostAsync("api/Values", content);
    if (response.IsSuccessStatusCode) {
        string jsonString = await response.Content.ReadAsStringAsync();
        exercicios = JsonConvert.DeserializeObject<List<ExercicioTreinoSCMDTO>>(jsonString);
    }
    return exercicios;
}
```

Figura 85 - Método responsável por comunicar com a *SmartCoachManager* para a obtenção dos exercícios de treino

Por último, de modo a apresentar os detalhes de um exercício existente no treino gerado, o utilizador é redirecionado para a plataforma *SmartCoachManager*, através de um "*href*" relacionado com o ID do exercício selecionado.

5.4.11 Visualizar *dashboard* estatístico

O *dashboard* trata-se de um painel estatístico aonde um utilizador associado a uma equipa poderá visualizar um conjunto de gráficos sobre a sua equipa, tais como, o número de jogos realizados por mês e ano, o registo da equipa (% vitórias, % empates, % derrotas), entre outros, conforme ilustrado na Figura 86. Por outro lado, é possível visualizar indicadores estatísticos individuais para um certo jogador, estes indicadores tem o objetivo de apresentar o resumo da performance do jogador num jogo específico, assim como, o resumo da evolução do mesmo, comparando a média da estatísticas do jogador, com os valores referencia e valores médios da equipa (posição semelhante), a Figura 87, apresenta os indicadores referidos.

Figura 86 - Componentes da estatística coletiva do *dashboard*

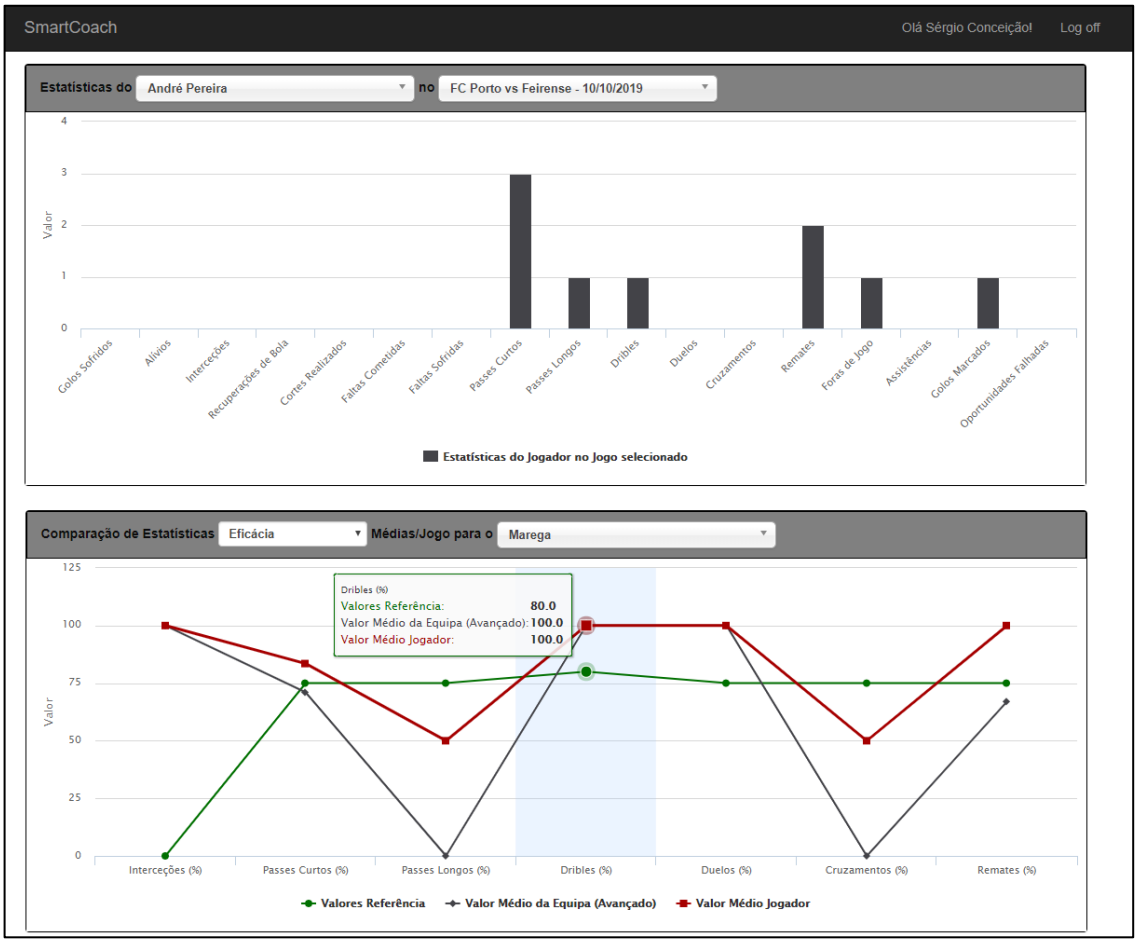


Figura 87 - Componentes da estatística individual do *dashboard*

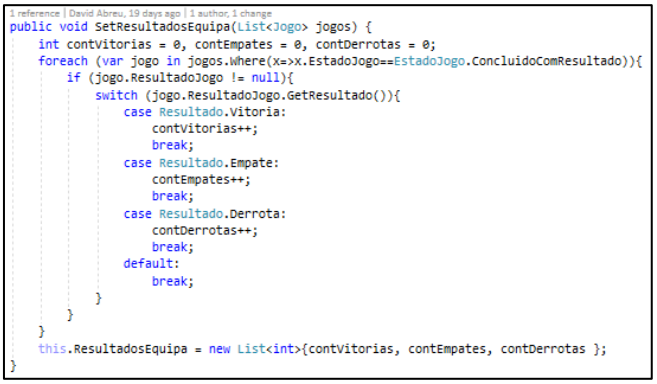
Para o desenvolvimento deste caso de uso, foi inicialmente necessário realizar um estudo sobre que gráficos estatísticos devem estar presentes nesta componente, após essa determinação, seguiu-se criação da *ViewModel* associada ao *dashboard*, como apresentado na Figura 88.

```
4 references | David Abreu, 19 days ago | 1 author, 1 change
public class DashboardViewModel
{
    8 references | David Abreu, 19 days ago | 1 author, 1 change
    public Equipa Equipa { get; set; }
    7 references | David Abreu, 19 days ago | 1 author, 1 change
    public List<int> ResultadosEquipa { get; set; }
    3 references | David Abreu, 19 days ago | 1 author, 1 change
    public IDictionary<GrupoPosicao, int> JogadoresPorPosicaoPrincipal { get; set; }
    19 references | David Abreu, 19 days ago | 1 author, 1 change
    public IDictionary</* Year */ int, IDictionary</* Month */ int, int>> JogosRealizadosPorAnoMes { get; set; }
    4 references | David Abreu, 19 days ago | 1 author, 1 change
    public IDictionary<string, int> NumeroEstatisticasJogoRecolhidasByJogador { get; set; }
```

Figura 88 - Excerto de código referente à classe *DashboardViewModel*

Após isso, é necessário preencher cada um dos objetos, assim sendo, para cada campo do *ViewModel* é associado um "*Set*" específico, este deve determinar o valor a ser adicionado à variável consoante os dados disponibilizados. De seguida, na Figura 89 é exibido o método

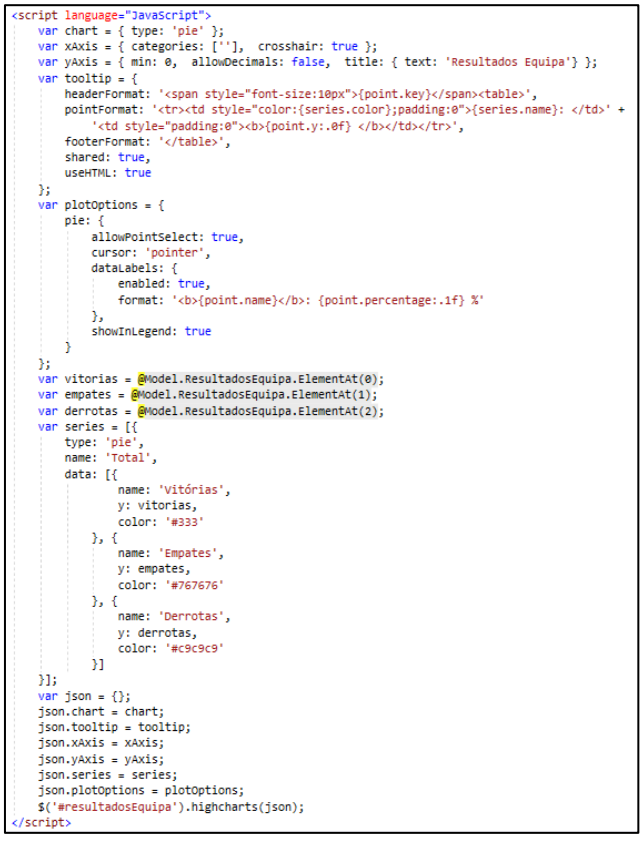
SetResultadosEquipa(), responsável pelo cálculo do número de vitórias/empates/derrotas dos jogos realizados da equipa.



```
1 reference | David Abreu, 19 days ago | 1 author, 1 change
public void SetResultadosEquipa(List<Jogo> jogos) {
    int contVitorias = 0, contEmpates = 0, contDerrotas = 0;
    foreach (var jogo in jogos.Where(x=>x.EstadoJogo==EstadoJogo.ConcluidoComResultado)){
        if (jogo.ResultadoJogo != null){
            switch (jogo.ResultadoJogo.GetResultado()){
                case Resultado.Vitoria:
                    contVitorias++;
                    break;
                case Resultado.Empate:
                    contEmpates++;
                    break;
                case Resultado.Derrota:
                    contDerrotas++;
                    break;
                default:
                    break;
            }
        }
    }
    this.ResultadosEquipa = new List<int>{contVitorias, contEmpates, contDerrotas };
}
```

Figura 89 - Método responsável pelo cálculo do número de vitórias/empates/derrotas

Por fim, posteriormente à inicialização dos dados do *ViewModel*, é necessário apresentá-los em formato de gráfico, para isso, foi utilizado o plugin *HighCharts*, documentado na secção **5.3.3 - Plugins utilizados**. Este software, consoante um script de inicialização, como demonstrado na Figura 90, proporciona o gráfico desejado, de acordo com os dados disponibilizados.



```
<script language="JavaScript">
    var chart = { type: 'pie' };
    var xAxis = { categories: [''],  crosshair: true };
    var yAxis = { min: 0,  allowDecimals: false,  title: { text: 'Resultados Equipa'} };
    var tooltip = {
        headerFormat: '<span style="font-size:10px">{point.key}</span><table>',
        pointFormat: '<tr><td style="color:{series.color};padding:0">{series.name}: </td>' +
            '<td style="padding:0"><b>{point.y:.0f} </b></td></tr>',
        footerFormat: '</table>',
        shared: true,
        useHTML: true
    };
    var plotOptions = {
        pie: {
            allowPointSelect: true,
            cursor: 'pointer',
            dataLabels: {
                enabled: true,
                format: '<b>{point.name}</b>: {point.percentage:.1f} %'
            },
            showInLegend: true
        }
    };
    var vitorias = @Model.ResultadosEquipa.ElementAt(0);
    var empates = @Model.ResultadosEquipa.ElementAt(1);
    var derrotas = @Model.ResultadosEquipa.ElementAt(2);
    var series = [{
        type: 'pie',
        name: 'Total',
        data: [{
                name: 'Vitórias',
                y: vitorias,
                color: '#333'
            }, {
                name: 'Empates',
                y: empates,
                color: '#767676'
            }, {
                name: 'Derrotas',
                y: derrotas,
                color: '#c9c9c9'
            }]
    }];
    var json = {};
    json.chart = chart;
    json.tooltip = tooltip;
    json.xAxis = xAxis;
    json.yAxis = yAxis;
    json.series = series;
    json.plotOptions = plotOptions;
    $('#resultadosEquipa').highcharts(json);
</script>
```

Figura 90 - Script de inicialização do gráfico circular do registo da equipa

5.4.12 Gerar relatórios

Este caso de uso diz respeito à geração de relatórios individuais, por parte do treinador, e deve conter o resumo de toda a informação associada a um jogador desejado em formato *PDF*, mais concretamente, deve ser analisado para cada estatística a sua média, assim como, os valores que ocorreram em cada jogo, como apresentado na tabela da Figura 91, para além disso, o relatório deverá conter os dados estatísticos recolhidos no último jogo do jogador, conforme a Figura 92.

É importante referir que este caso de uso não está totalmente completo, uma vez que falta a componente da importação dos dados obtidos para ficheiro *PDF*, no entanto, a visualização do conteúdo associado a esta funcionalidade está disponível.

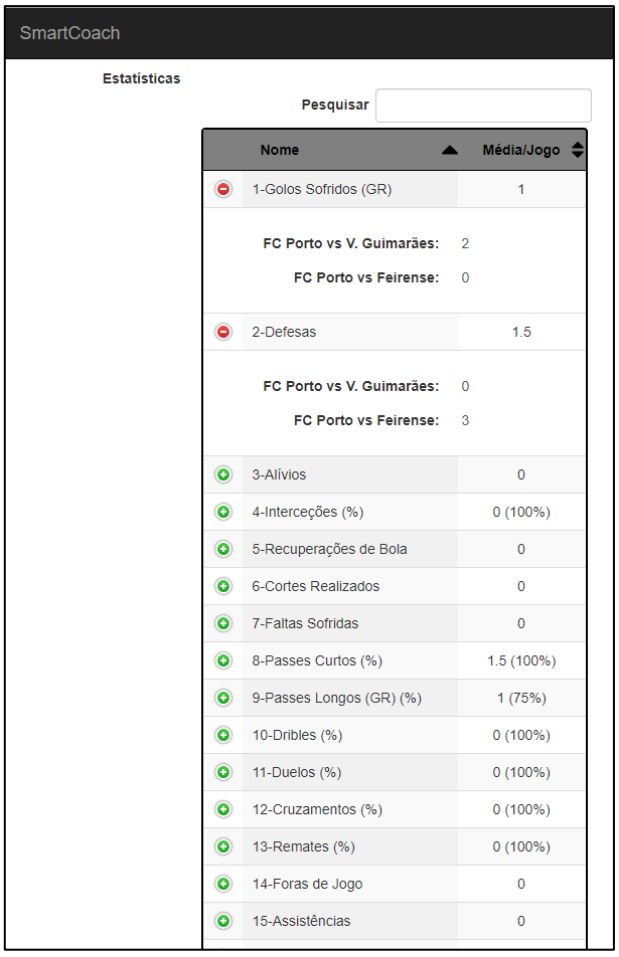


Figura 91 - Resumo das estatísticas para um jogador

Figura 92 - Estatísticas levantadas no último jogo para um jogador

Como mencionado acima, a implementação desta funcionalidade não foi totalmente concluída, no entanto, será esclarecido o processo de obtenção dos dados exibidos. Dessa forma, começando pelo cálculo das estatísticas médias, foi realizado um pedido à *SmartCoachAPI*, com vista a retornar a lista de todas as estatísticas do jogador, estas foram retornadas para a *view* e agrupadas pela classe *EstatisticaParaPerformance*, assim, foi possível calcular a média para cada uma das estatísticas e apresentada ao utilizador o resumo da performance do jogador em formato tabela. No caso, do utilizador quer saber detalhadamente os valores dessa estatística consoante os jogos, é invocada a função *GetDetalhesEstatistica*(), documentada na Figura 93, que calcula a média da respetiva estatística para cada um dos jogos em que o jogador esteve presente.

```
0 references | David Abreu, 19 days ago | 1 author, 1 change
public async Task<ActionResult> GetDetalhesEstatisticas(int eppID, string jogadorID) {
    if (!SmartCoachAPIClient.verifySession()){
        TempData["error"] = "A sua sessão expirou, por favor, faça login novamente!";
        return RedirectToAction("Erro", "Home");
    }

    string detalhes = "Erro ao obter detalhes!";
    ApplicationUser user = await EquipaHandler.GetUtilizadorByID(jogadorID);

    detalhes = "<div class='form-horizontal home-form-horizontal'>";
    foreach (var item in user.Jogador.EstatisticasJogoJogador.Where(x => x.PosicaoJogador != null)
        .SelectMany(x => x.EstatisticasJogador.Where(z=>z.DetalheEstatisticaPerformance.EstatisticaParaPerformanceID == eppID))
        .GroupBy(x => x.EstatisticaJogoJogador.EstatisticaJogoAppUser.Jogo)){
        foreach (var epps in item.GroupBy(x => x.DetalheEstatisticaPerformance).GroupBy(x => x.Key.EstatisticaParaPerformance).OrderBy(c => c.Key.ID)){
            var detalhesEficacia = "";
            if (epps.Key.Analise == AnaliseEstatisticaPerformance.Eficacia){
                double avgValueEficacia = 100;
                double avgTotal = epps.Sum(x => x.Average(z => z.ValorAtual));
                if (avgTotal > 0) {
                    double avgPositivos = epps.Where(x => x.Key.ClasseEficacia == SmartCoachApp.Models.ClasseEficacia.Positiva).Sum(x => x.Average(z => z.ValorAtual));
                    avgValueEficacia = Math.Round((avgPositivos / avgTotal) * 100,0);
                }
                detalhesEficacia = " ("+avgValueEficacia+"%)";
            }

            detalhes += "<div class='form-group home-form-group'><label class='control-label col-md-8' style='cursor:pointer'>" +
                "<a href='/InserirEstatisticas/VisualizarEstatisticasJogador?jogoID=" + item.Key.ID+ "&amp;jogadoresIDs=%5B"+ jogadorID + "%5D' style='color: #333;'>"
                + item.Key.GetNomeSimples()+":</a></label>";
            detalhes += "<div class='col-md-3 home-form-info'><span class='toggle-info'>" + epps.Sum(y => y.Average(x => x.ValorAtual)) + detalhesEficacia+ "</span></div>";
            detalhes += "</div>";
        }
    }
    detalhes += "</div>";

    return Content(detalhes, "text/html");
}
```

Figura 93 - Método que cálcula a média de uma estatística para cada jogo

Por fim, em relação à visualização das estatísticas recolhidas no último jogo de um certo jogador, foi seguida a mesma abordagem que a funcionalidade "Inserir estatísticas jogo", no entanto, para este caso específico, foram só mantidas as permissões de visualização da respetiva *view*.

5.4.13 Gestão de utilizadores/jogadores associados

Por fim, devido à semelhança entre os casos de uso de gestão de utilizadores associados e da gestão de jogadores associados estes serão unidos numa única secção. O objetivo destas funcionalidades, passa pela gestão da associação entre jogadores e outros utilizadores do sistema. Por outras palavras, um jogador deverá puder convidar utilizadores do sistema para serem seus associados (via inserção do e-mail), e vice-versa, como ilustrado na Figura 94. Por outro lado, deve ser possível um utilizador aceitar ou rejeitar esta associação, sendo que se a mesma for aceite ficará visível para ambas as entidades, na secção "Utilizadores/Jogadores Associados", conforme a Figura 95. Por fim, caso desejado é possível remover esta associação por qualquer um dos lados, selecionando a opção "Remover".

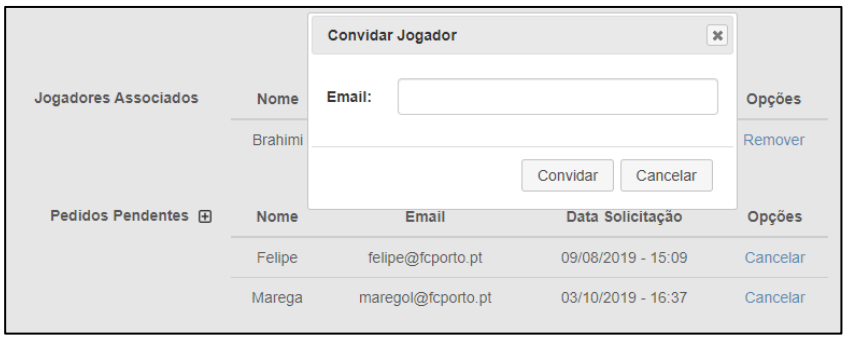


Figura 94 - *Pop-up* de convidar um jogador para ser seu associado

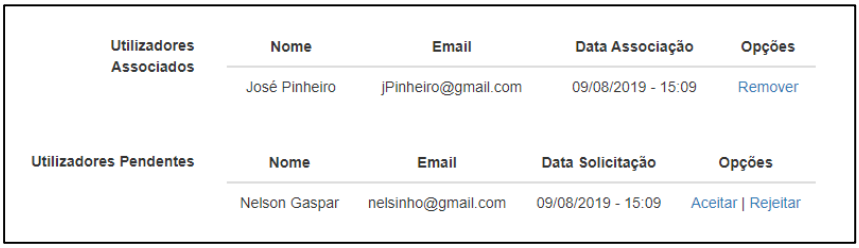


Figura 95 - Exemplo dos utilizadores associados e pendentes para um jogador do sistema

Para documentar da melhor forma o desenvolvimento deste caso de uso, será dividido a sua implementação sobre as três subfuncionalidades, nomeadamente:

Convidar associado: Para a implementação desta atividade, foi gerado no ficheiro relativo à *view*, um *pop-up* que solicita o email do utilizador a ser convidado. De seguida, é invocado via *Ajax* a aplicação *web*, que comunica por sua vez com a aplicação servidora, sendo esta responsável, por verificar a existência do email introduzindo e realizar a lógica necessária para convidar o jogador a ser seu associado. Após isso, é retornado o resultado desta operação para a *view*, a função *convidarJogadorAssociado*(), está presente na Figura 96 e mostra a

componente da *view* responsável pela criação do *pop-up*, assim como, o envio e receção dos dados para o sistema.

```
function convidarJogadorAssociado() {
    var mensagem = $('<div id="conformBox" class="form-horizontal" style="overflow:hidden">'
        + "<div class='form-group' style='margin-top: 15px;'><label class='control-label col-md-2'>Email: </label>"
        + "<div class='col-md-10'><input id='inputEmail' type='text' class='form-control'></input></div></div>" + '</div>');
    mensagem.dialog({
        title: "Convidar Jogador",
        closeOnEscape: true,
        close: function (event, ui) {
            $(this).dialog('destroy').remove();
        },
        width: 400,
        modal: true,
        buttons: [{
            text: "Convidar",
            click: function () {
                if ($("#inputEmail").val().trim().length > 0 && isEmail($("#inputEmail").val().trim())) {
                    var data = {
                        email: $("#inputEmail").val().trim(),
                    };
                    var exists = checkEmailExists("/Manage/CheckJogadorHasEmail", data);
                    if (exists != null) {
                        if (exists) {
                            var data = {
                                userID: '@Model.Id',
                                emailJog: $("#inputEmail").val().trim(),
                            };
                            var check = saveDataAjax("/Manage/ConvidarJogadorAssociado", data);
                            if (check) {
                                location.reload();
                            } else {
                                alert("Erro, por favor tente mais tarde!");
                            }
                        } else {
                            alert("Email inexistente no sistema, por favor, insira um Email válido!");
                            $("#inputEmail").focus();
                        }
                    } else {
                        alert("Erro, por favor tente mais tarde!");
                    }
                } else {
                    alert("Insira um Email válido!");
                    $("#inputEmail").focus();
                }
            }
        }, {
            text: "Cancelar",
            click: function () {
                $(this).dialog("close");
            }
        }]
    });
}
```

Figura 96 - Script da *view* responsável pela realização de convites de associação

Remover associação: A remoção de um jogador/utilizador associado, é possível quando o utilizador corrente seleciona a opção “Remover” na associação existente, esta ação irá invocar a aplicação servidora, de modo, a remover a associação entre os dois utilizadores. A Figura 97, apresenta o método responsável por esta ação.

```
0 references | David Abreu, 136 days ago | 1 author, 1 change
public async Task<IHttpActionResult> RemoverJogadorAssociado(GerirUtilizadorAssociadoJogadorDTO gerirDTO)
{
    ApplicationUser user = await unitOfWork.UserRepository.GetUserByID(gerirDTO.UserID);
    if (user == null || user.UtilizadorAssociado == null || user.UtilizadorAssociado.JogadoresAssociados == null){
        return BadRequest();
    }
    var jogadorAssociado = user.UtilizadorAssociado.JogadoresAssociados.Where(x => x.JogadorID == gerirDTO.JogadorID && x.UtilizadorAssociadoID == gerirDTO.UserID).FirstOrDefault();
    if (jogadorAssociado == null){
        return BadRequest();
    }
    user.UtilizadorAssociado.JogadoresAssociados.Remove(jogadorAssociado);
    return Ok(unitOfWork.UserRepository.UpdateUser(user));
}
```

Figura 97 - Método da aplicação servidora responsável pela remoção da associação

Aceitar/Rejeitar associação: Para a aceitação/rejeição de associações pendentes, foi implementado um mecanismo simples de redireccionamento para o método responsável por aceitar ou rejeitar o utilizador pretendido na aplicação servidora, este método, de uma forma simplificada, adiciona à lista de utilizadores associados ou remove o convite de associação.

6 Avaliação

Neste capítulo é exposto como será realizado o processo de avaliação para as principais características a avaliar do projeto. Assim sendo, primeiramente são identificadas as métricas e grandezas que devem ser avaliadas. Após isso, são definidas as hipóteses das grandezas identificadas anteriormente. De seguida, é apresentada e justificada a metodologia a utilizar na avaliação destas hipóteses tendo em conta os grupos de controlo que serão abordados. E, por fim, é realizado a análise dos resultados obtidos tendo em conta a metodologia descrita.

6.1 Grandezas

Considerando o tipo de projeto em questão e o conjunto de funcionalidades que a solução idealizada propõe, as grandezas a considerar prendem-se com o tempo dos cálculos realizados no módulo de análise de recomendação de treinos e a satisfação do utilizador em duas vertentes: utilização geral da aplicação e satisfação obtida aquando da utilização do módulo de recomendação.

Serão então, utilizadas duas grandezas diferentes para a avaliação de todo o projeto:

- **Avaliar internamente o sistema**: É sempre importante avaliar as especificidades de um produto, principalmente, quando se trata de um software informático que pode estar suscetível a diversos tipos de falha, desse modo, deverá ser analisado e avaliado, a precisão e o tempo dos valores provenientes do algoritmo de recomendação, assim como, das outras funcionalidades relevantes do sistema.
- **Avaliar a satisfação do utilizador**: Por outro lado, a satisfação do utilizador para o projeto em questão é talvez o fator mais importante a ter em consideração, isto devido a ser uma aplicação de suporte para múltiplos utilizadores e a opinião dos mesmos poderá condicionar o futuro a dar a este projeto.

6.2 Hipóteses

Com base nas grandezas identificadas anteriormente, torna-se necessário testar as mesmas. Para isso, serão expostos testes de hipóteses. Estes, são considerados um procedimento estatístico que permite tomar uma decisão (aceitar ou rejeitar a hipótese nula H_0) entre duas ou mais hipóteses (hipótese nula H_0 ou hipótese alternativa H_1).

A primeira, é referente à grandeza "**Avaliar internamente o sistema"**, desse modo, são apresentados dois testes de hipótese:

Teste 1: Avaliar internamente o sistema (Através da realização de testes de software)

H_0: Todos os testes são concluídos com sucesso.

H_1: Ocorre uma falha num ou mais testes.

Teste 2: Avaliar internamente o sistema (Tempo do algoritmo de recomendação)

H_0: O sistema demora menos que 1 segundo para apresentar recomendações de treino.

H_1: O sistema demora mais que 1 segundo para apresentar recomendações de treino.

De seguida, são então apresentados os testes de hipótese relativos à grandeza "**Avaliar satisfação do utilizador**":

Teste 3: Avaliar satisfação do utilizador (Através de inquéritos de satisfação)

H_0: O inquérito de satisfação realizado tem uma média geral de satisfação de quatro ou mais.

H_1: O inquérito de satisfação realizado tem uma média geral de satisfação abaixo de quatro.

6.3 Metodologia

Com vista a testar e avaliar as hipóteses definidas anteriormente, foram identificadas duas metodologias a pôr em prática, isto devido às grandezas a avaliar terem responsabilidades totalmente distintas. Assim sendo, são de seguida, apresentadas ambas as metodologias:

Na sequência da grandeza **"Avaliar internamente o sistema"**, podemos concluir que o sistema em geral deverá ser testado e avaliado, para isso deverá ser adotado o modelo V, que rege sobre as normas de verificação e validação do software ao longo do seu desenvolvimento. Já no caso, de apurar o tempo despendido no algoritmo de recomendação irá ser invocado o algoritmo de recomendação vinte cinco vezes, juntamente com uma função de cronômetro disponibilizada através da classe *Stopwatch*, com isso, o sistema irá registar os vários tempos

obtidos (desde a chamada até ao resultado final da recomendação), importando os mesmos para um ficheiro *CSV*, como apresentado no excerto de código da Figura 98.

É ainda de salientar que a importação destes dados para um ficheiro *Excel,* irá permitir o cálculo dos valores da média e do desvio padrão associados, assim como, um gráfico ilustrativo dos tempos obtidos.

```
string csvpath = "D:\\TempoAlgoritmo.csv";

Boolean fileExists = System.IO.File.Exists(csvpath);
StringBuilder csvcontent = new StringBuilder();
System.IO.File.AppendAllText(csvpath, csvcontent.ToString());

var watch2 = System.Diagnostics.Stopwatch.StartNew();
//algortimo recomendação
List<AtributoRecomendacao> result = await Helpers.SistemaRecomendacao.GerarRecomendacao(equipa, jogador, listAllEstatisticas, unitOfWork);
watch2.Stop();

string content = watch2.ElapsedMilliseconds.ToString();
csvcontent.AppendLine(content);
System.IO.File.AppendAllText(csvpath, csvcontent.ToString());
```

Figura 98 - Excerto de código relativo ao processo de cálculo do tempo do algoritmo

Já em relação à grandeza **"Avaliar a satisfação do utilizador"**, será primeiramente, disponibilizado o software produzido para um certo número de indivíduos que de certa forma estão ligados ao futebol. Após um período de utilização, será obtida a opinião dos utilizadores através de inquéritos para concluir a satisfação geral do utilizador, estes deverão avaliar a usabilidade, a eficiência, e claro a utilidade da aplicação desenvolvida. Mais detalhadamente, o inquérito partilhado deverá dispor de um vasto número de perguntas tendo em conta o papel do utilizador na aplicação e das funcionalidades existentes. Para cada pergunta será apresentada uma escala de possíveis respostas que vão de 1 até 5, com a descrição "Discordo completamente", até "Concordo completamente", como apresentado de seguida na Tabela 6. Após essas perguntas, irá existir uma pergunta final opcional, na qual o utilizador poderá dar a sua opinião sobre o respetivo software, assim como, possíveis melhorias que gostaria de ver implementadas no futuro.

Tabela 6 - Escala utilizada nos inquéritos

Descrição	Escala
Discordo completamente	1
Discordo	2
Sem opinião	3
Concordo	4
Concordo completamente	5

6.4 Análise de Resultados

Na presente secção será detalhada a análise de resultados para cada uma das hipóteses a verificar tendo em conta a metodologia abordada acima. Assim sendo, primeiramente, serão apresentados os resultados dos testes realizados, de seguida, é validado o tempo de execução do algoritmo, e por fim, é analisado a satisfação dos utilizadores que dispuseram do software em causa.

6.4.1 Testes de software realizados

Com o objetivo de assegurar o total funcionamento da aplicação, ao longo do desenvolvimento do projeto foram realizados testes de software. Considera-se que a elaboração e execução de testes de software é o processo de avaliar um sistema com a intenção de descobrir um erro [68]. Com esse propósito foi adotado o modelo V, que rege sobre as normas de verificação e validação do software ao longo do seu desenvolvimento, no mesmo são identificados quatro tipos de testes, para as quatro fases do desenvolvimento do software, como ilustrado na Figura 99. De seguida, serão analisados cada um dos quatro tipos de teste elaborados.

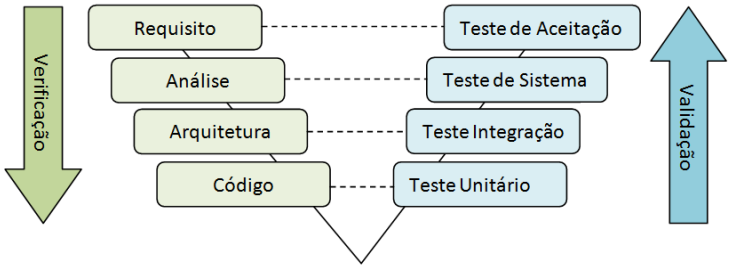


Figura 99 - Modelo V (Fases do Desenvolvimento X Fases dos Testes) [69]

6.4.1.1 Testes Unitários

Os testes unitários têm por objetivo verificar cada unidade que compõe o software, de modo isolado, assim sendo, os mesmos pretendem validar os dados retornados de cada método do domínio, verificando assim, o correto funcionamento da aplicação.

Para a implementação dos testes unitários, foi primeiramente necessário importar a *Unit Testing Framework*. Assim, com o auxílio desta ferramenta foram então decididos testar todos os métodos relacionados com o mecanismo de recomendação, a Figura 100 apresenta um exemplo de teste realizado a esse módulo. O mesmo, diz respeito à componente do sistema de recomendação que tem a responsabilidade de calcular o valor do peso do jogo, consoante as variáveis de contexto.

```
[TestMethod()]
0 references | 0 changes | 0 authors, 0 changes
public void caclularPesoJogoTest()
{
    Jogo jogo = new Jogo() {
        TipoJogo = TipoJogo.Amigavel,
        Data = DateTime.Now,
        EstadoJogo = EstadoJogo.ConcluidoComResultado,
        QualidadePiso = QualidadePiso.Bom,
        TipoPiso = TipoPiso.RelvadoSintetico,
        CondicoesAtmosfericas = CondicoesAtmosfericas.TempoLimpo
    };
    double resultadoEsperado = 0.925;
    double resultadoObtido = Math.Round(SistemaRecomendacao.caclularPesoJogo(jogo), 3);
    Assert.AreEqual(resultadoEsperado, resultadoObtido);
}
```

Figura 100 - Teste unitário ao cálculo do peso do jogo

6.4.1.2 Testes de Integração

Os testes de integração têm como objetivo encontrar falhas de integração entre as unidades, neste caso, entre a aplicação servidora e a base de dados. Desse modo, foi então necessário simular as operações da base de dados, através da *framework Moq*. Com o auxílio desta ferramenta, implementou-se o método "*Initialize*()", este tem a finalidade, de formular valores predefinidos para as operações da BD existentes nos repositórios. De seguida, na Figura 101 é ilustrado esta inicialização, assim como, o teste realizado ao método "*GetMinhaEquipa*()".

```
[TestInitialize]
0 references | 0 changes | 0 authors, 0 changes
public void Initialize()
{
    mock = new Mock<IEquipaRepository>();

    equipas = new List<Equipa>
    {
        new Equipa{ CodigoEquipa = "As28Z", Abreviatura = "FCP", Nome = "FC Porto",
            Escalao = EscalaoEquipa.Seniores, Localidade="Porto", Estadio = "Estádio do Dragão"
        },
        new Equipa{ CodigoEquipa = "Zk2F2", Abreviatura = "SLB", Nome = "SL Benfica",
            Escalao = EscalaoEquipa.Seniores, Localidade="Lisboa", Estadio = "Estádio da Luz"
        },
        new Equipa{ CodigoEquipa = "2X31A", Abreviatura = "SCP", Nome = "Sporting CP",
            Escalao = EscalaoEquipa.Seniores, Localidade="Lisboa", Estadio = "Estádio de Alvalade"
        },
    };

    mock.Setup(m => m.GetEquipas()).ReturnsAsync(equipas);
}

[TestMethod()]
0 references | 0 changes | 0 authors, 0 changes
public async Task GetMinhaEquipaTestAsync()
{
    EquipasController controller = new EquipasController();
    var result = await controller.GetMinhaEquipa();
    var expResult = equipas.ElementAt(0);
    Assert.Equals(result, expResult);
}
```

Figura 101 - Teste de integração ao *controller* das equipas

6.4.1.3 Testes de Sistema

Por outro lado, foram também realizados testes de sistema. Estes, representam a avaliação do software como um utilizador final da aplicação, com o objetivo de testar certos conjuntos de funcionalidades de forma informal.

Desse modo, foram realizados seis testes de sistema à aplicação, mais detalhadamente, a lista dos testes realizados encontra-se disponível no **Anexo 2 – Testes de Sistema**, no entanto, de seguida, será apresentado na Tabela 7, um exemplo de teste de sistema realizado às funcionalidades de gerar treino recomendado e visualizar os detalhes do respetivo treino.

Tabela 7 - Exemplo teste de sistema realizado

Teste de Sistema Nº5	
Descrição:	Gerar um treino recomendado e visualizar os detalhes de cada um dos exercícios presentes no treino.
Resultado esperado:	Criação de um treino recomendado consoante os valores provenientes do sistema de recomendação e visualização no serviço externo *SmartCoachManager* dos detalhes dos exercícios de treino gerados.
Resultado obtido:	PASSOU

6.4.1.4 Testes de Aceitação

Por último, os testes de aceitação caracterizam-se por testar e aceitar de forma individual cada uma das funcionalidades desenvolvidas, para isso, foi decidido testar todas as funcionalidades únicas do sistema através do utilizador com a função de treinador.

Assim sendo, foram realizados doze testes de aceitação, mais detalhadamente, a lista dos testes realizados encontra-se disponível no **Anexo 3 – Testes de Aceitação**, no entanto, de seguida, será apresentado na Tabela 8, um exemplo de teste de sistema realizado à funcionalidade de gestão de jogadores.

Tabela 8 - Exemplo teste de aceitação realizado

Teste de Aceitação Nº5 – Gerir Jogadores	
Descrição:	O treinador convida um novo jogador (via e-mail) para ingressar na sua equipa.
Resultado esperado:	É enviado um convite para o e-mail do jogador inserido para adesão à equipa do treinador.
Resultado obtido:	PASSOU

6.4.1.5 Conclusão

Por fim, verificando que todos os testes realizados são bem-sucedidos, é possível concluir que a hipótese nula (H_0), referente ao teste de hipótese número um (presente na secção **6.2 - Hipóteses**) é aceite, o que demonstrando que o resultado geral sobre avaliação interna do sistema é positiva.

6.4.2 Tempo algoritmo recomendação

Para obtenção do tempo de execução do algoritmo, foi seguida a metodologia descrita na secção **6.3 - Metodologia**, assim sendo, foram realizadas 25 execuções do algoritmo de recomendação e guardados os tempos obtidos num ficheiro *Excel,* o **Anexo 4 – Resultados tempo do algoritmo**, contém o conteúdo do respetivo ficheiro.

Posteriormente, de modo a analisar os dados relativos ao tempo de execução do algoritmo, será apresentado o Gráfico 3, este tem o objetivo de mostrar a função do tempo gasto em milissegundos (*ms)* pelo seu número de execução.

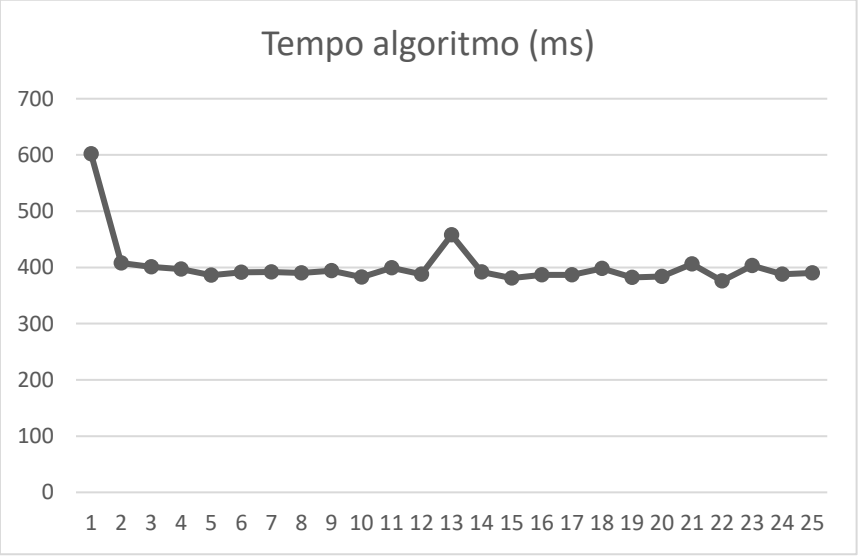


Gráfico 3 - Tempo de execução do algoritmo em milissegundos (*ms)* por tentativa

Após isso e ainda de acordo com dados obtidos, foi calculado quatro elementos estatísticos relevantes que condicionam a análise do tempo de execução, nomeadamente, o tempo máximo, o tempo mínimo, o valor médio e o desvio padrão, como apresentados de seguida, na Tabela 9.

Tabela 9 - Valores gerais da execução do algoritmo em milissegundos (*ms)*

Máximo	Mínimo	Média	Desvio Padrão
602	376	402,5	43,5

Ao visualizar a tabela acima, é possível verificar que os tempos de execução do algoritmo variam entre os 376 *ms* e os 602 *ms*, o que ainda apresenta uma variação considerável, no entanto, ao analisar o gráfico verifica-se que o valor máximo (602 *ms*), ocorre na primeira execução, o que é esperado, uma vez que esta inclui o tempo que o compilador *Just In Time* (JIT) gasta a converter o código da *Microsoft Intermediate Language* (MSIL) no código de

máquina executável. Subsequentemente, todas as chamadas realizadas estão a reutilizar esse código já compilado [68] [69].

Por outro lado, considerando o valor médio (402,5 *ms*) e do desvio padrão (43,5 *ms*), podemos concluir que estes representam da melhor forma a execução do algoritmo, apresentando resultados positivos para o contexto em questão.

Por último, ao comparar a média dos tempos obtidos com o teste de hipótese número dois (presente na secção **6.2 - Hipóteses**), é possível verificar que a hipótese nula (H_0) é aceite, demonstrando que o resultado sobre este tópico é positivo.

6.4.3 Satisfação do utilizador

Nesta secção será detalhada o grau de satisfação obtido aos utilizadores que avaliaram a usabilidade da presente aplicação, para isso, primeiramente, foi dada uma introdução teórica sobre a aplicação, e de seguida, disponibilizada a aplicação para os utilizadores testarem a sua usabilidade geral. Após isso, é dada a possibilidade de o utilizador responder a um inquérito, com vista, a obter a sua opinião e experiência para com o software fornecido. Mais detalhadamente, o inquérito é de cariz anônimo e contém dezoito perguntas relacionadas com as duas aplicações, nomeadamente, a *SmartCoach* e a *SmartCoachManager*. No entanto, para este caso de estudo, só serão avaliadas as respostas às perguntas gerais, assim como, à componente *SmartCoach*.

O presente questionário foi desenvolvido com o auxílio do *google forms* [70], encontra-se disponível no **Anexo 5 – Inquérito de Satisfação**, o mesmo foi respondido por dez utilizadores, sendo que os mesmos se dividem em quatro cargos distintos. Assim sendo, para uma primeira análise é determinado a função de distribuição dos utilizadores inquiridos em questão, o Gráfico 4 apresenta essa distinção, através do cálculo do número de utilizadores associados a cada cargo possível.

Gráfico 4 - Número dos inquiridos por cargo no seu clube

De seguida, de modo a analisar os restantes dados do questionário, foram convertidos os resultados obtidos para uma tabela onde são apresentadas as frequências relativas por resposta. Assim, a Tabela 10 apresenta a percentagem de escolha de cada resposta face às perguntas existentes, e por fim, é calculada ainda a média final ponderada da respetiva resposta [1-5].

Tabela 10 - Frequência relativa de cada resposta do inquérito

Pergunta	**1**	**2**	**3**	**4**	**5**	**Média**
1) A aplicação tem todas as funcionalidades desejadas?	0%	0%	0%	80%	20%	4,2
2) Considera a aplicação simples e intuitiva?	0%	0%	30%	50%	20%	3,9
3) A aplicação possui uma boa apresentação e design?	0%	10%	10%	80%	0%	3,7
4) Considera a aplicação com tempo de resposta curto?	0%	10%	0%	50%	40%	4,2
5) A gestão da equipa tem as funcionalidades pretendidas?	0%	0%	0%	60%	40%	4,4
6) Considera a configuração dos valores referência uma mais-valia?	0%	0%	0%	50%	50%	4,5
7) Considera a recolha de estatísticas de jogo completa e intuitiva?	0%	10%	0%	60%	30%	4,1
8) As recomendações de treino geradas são precisas face às necessidades do jogador?	0%	0%	0%	90%	10%	4,1
9) O *dashboard* apresenta os gráficos/indicadores estatísticos desejados?	0%	0%	10%	70%	20%	4,1
15) A aplicação poderia ser útil na sua equipa?	0%	0%	0%	80%	20%	4,2
16) Como classificaria a aplicação?	0%	0%	0%	70%	30%	4,3
17) Recomendaria a aplicação a outros utilizadores?	0%	0%	10%	0%	90%	4,8
Distribuição total	**0%**	**2,5%**	**5%**	**61,5%**	**31%**	**4,2**

Ao visualizar a tabela acima, é inicialmente distinguido que os inquiridos estão satisfeitos com as funcionalidades apresentadas pela aplicação, obtendo uma classificação média de 4.2 para a primeira questão. De seguida, podemos verificar que perante as perguntas de avaliação visual da aplicação, tais como, avaliar se a aplicação é simples e intuitiva e se a mesma apresenta bom design. Os utilizadores respondem um pouco mais negativamente, obtendo valores de 3.9 e 3.7 respetivamente. Por fim, face à pergunta relacionada com a rapidez da aplicação os inquiridos avaliam esta, com um valor médio de 4.2, o que demonstra de uma maneira geral que a aplicação tem uma boa performance nos dispositivos do utilizador.

Por outro lado, as perguntas exclusivas da vertente *SmartCoach* (perguntas 5, 6, 7, 8 e 9), tem o objetivo de averiguar se as funcionalidades da aplicação desenvolvida são úteis e eficazes para o utilizador em estudo. Ao analisar a tabela, verificamos que de uma maneira geral os utilizadores deram boas classificações nestas questões (4.4, 4.5, 4.1, 4.1 e 4.1, respetivamente), concluindo assim, que estas funcionalidades desenvolvidas apresentam bons indicadores em função da amostra dos inquiridos. Por fim, de modo, a concluir esta vertente, é salientada a pergunta número oito: "As recomendações de treino geradas são precisas face às necessidades do jogador?" Esta caracteriza-se por ser talvez a pergunta específica mais importante do questionário, uma vez que se trata do sucesso ou insucesso do mecanismo de recomendação. Assim sendo, ao consultar a tabela, podemos verificar que esta pergunta teve uma classificação de 4.1, o que demonstra uma boa satisfação dos utilizadores face ao sistema de recomendação desenvolvido.

Em relação às perguntas que visam determinar satisfação geral do utilizador, conforme: "A aplicação poderia ser útil na sua equipa?", "Como classificaria a aplicação?" e "Recomendaria a aplicação a outros utilizadores?", estas foram respondidas positivamente, obtendo valores de 4.2, 4.3 e 4.8, respetivamente. O que indica, de um ponto de vista generalizado, que os inquiridos consideram a aplicação útil e classificam a mesma como boa-muito boa, estando todos os inquiridos de acordo na sua recomendação para com outros utilizadores.

Por fim, analisando a pergunta número dezoito, referente a possíveis melhorias a realizar na aplicação obteve-se três respostas que são de seguida apresentadas na Tabela 11, ao observar mais cuidadosamente a mesma, é possível concluir que estas apontam para a melhoria da interface/design da aplicação, desenvolvimento de uma abordagem mais centrada na equipa e não no jogador e por último, a existência de exercícios de treinos mais variados.

Tabela 11 - Respostas à pergunta relacionada com possíveis melhorias

Possíveis melhorias	
1º	Design, exercícios mais variados, intuitividade
2º	Melhor design
3º	Com a individualidade ser capaz de criar estatística coletiva, exercícios específicos para a equipa e não individualmente.

Face aos resultados positivos obtidos nos parágrafos anteriores é decido calcular a distribuição total para cada uma das possíveis respostas, assim como, a média final do questionário analisado. Desse modo, foi obtido os valores de 0% para a resposta "Discordo completamente", 2.5% para "Discordo", 5% para "Sem opinião", 61.5% para "Concordo" e finalmente, 31% para "Concordo completamente", resultando numa média final de 4.2, o que aparenta ser um ótimo indicador para o projeto realizado.

Relacionando o valor da média final obtida de 4.2, com o teste de hipótese número três (presente na secção **6.2 - Hipóteses**), é possível verificar que a hipótese nula (H_0) é aceite, o que comprova que o resultado geral sobre a satisfação do utilizador é bastante positiva.

7 Conclusão

A elaboração da presente dissertação teve como principal objetivo o estudo e desenvolvimento de um protótipo da aplicação desportiva *SmartCoach*, destinada ao futebol de formação. Esta plataforma, visa auxiliar o treinador a gerir a sua equipa, assim como, ajudar os jovens atletas em questão a melhorar a sua performance desportiva, através de um mecanismo inteligente de recomendação de treinos.

Assim numa primeira fase, foi contextualizado o tema da dissertação, relacionando as novas tecnologias com o conceito do desporto, mais concretamente o futebol. Permitindo concluir que nos dias de hoje, os softwares desta área são cada vez mais importantes, oferecendo aos seus utilizadores inúmeras vantagens face aos seus adversários. Por fim, é ainda possível verificar que esta área tem uma margem de progressão bastante alta.

Após esta contextualização, foi produzido um estudo sobre a componente de recomendação a ser implementada. Nesta, foi possível concluir que existem duas técnicas de recomendação possíveis de ser adotadas para o sistema a desenvolver. Com base numa análise realizada, foi determinando que será adotada a técnica baseada em conteúdo, deixando a abordagem híbrida para trabalho futuro.

Por outro lado, foi igualmente produzido um estudo sobre a existência de aplicações similares de auxílio à gestão da equipa para treinadores de futebol. Neste estudo foi verificado que dos cinco softwares analisados, nenhum apresenta um mecanismo de recomendação de treinos, o que torna o projeto a desenvolver único nesta vertente.

De modo a terminar a face de investigação do projeto, foi ainda efetuada uma análise ao tema que resulta na elaboração de uma proposta de valor. Esta resumidamente, passa pela oferta de uma aplicação desportiva, com a finalidade de melhorar a performance dos resultados da sua equipa.

Após o fim da fase de investigação, foi então dado inicio ao processo de design da solução, neste são identificados todos os requisitos do sistema a serem implementados, e também, constatado que será adotada uma arquitetura monolítica (baseada na produção de uma API), que irá conter toda a lógica de negócio, assim como, comunicação com os restantes componentes do sistema, nomeadamente, a base de dados, a aplicação *web* e o serviço externo para obtenção de exercícios de treino.

Relativamente à implementação da solução, esta foi realizada ao longo de um período total de cerca de sete meses e apresenta a elaboração de todos os requisitos identificados na fase anterior. Para isso, primeiramente, foram desenvolvidos os três componentes do sistema, nomeadamente, a aplicação *web SmartCoachApp*, a aplicação servidora *SmartCoachAPI* e a base dados *SmartCoachDatabase*, e de seguida, realizadas as funcionalidades gerais da aplicação, tais como, gestão da equipa e seus utilizadores, gestão dos jogos, entre outras. Posto isto, e mediante o *feedback* obtido das reuniões de acompanhamento do projeto, foram implementadas as restantes funcionalidades, ligadas à recomendação de treinos e à componente da análise estatística. Estas, como documentado tem o objetivo de invocar o sistema de recomendação e comunicar com o serviço externo *SmartCoachManager*, para a obtenção dos exercícios de treino relacionados com os atributos calculados pelo SR.

Por fim, foi necessário avaliar a aplicação desenvolvida. Esta avaliação foi dividida em duas vertentes, a primeira, relacionada com a avaliação interna do sistema face à realização de testes de software, assim como, da análise ao tempo do algoritmo de recomendação e a segunda, relativa à satisfação dos utilizadores perante a resposta a um inquérito de satisfação. Analisando ambas as grandezas, é possível concluir que o sistema foi bem desenvolvido, apresentando resultados positivos na componente dos testes e no tempo de execução do algoritmo de recomendação, já em relação à opinião dos inquiridos considera-se que a aplicação foi um sucesso, uma vez que foi obtida a classificação final de 4.2, na escala de 1 a 5.

Pode-se assim concluir que a realização deste projeto foi considerada uma mais-valia, visto que a nível pessoal, para além da experiência adquirida, foram atingidas competências técnicas através da aplicação dos conceitos estudados, bem como, da investigação realizada. Inclusive será oportuno referir que com o resultado da investigação efetuada foram publicados três artigos científicos, entre eles:

- **Artigo da ISAMI 2019**: *Smart Coach - A Recommendation System for Young Football Athletes*;
- **Artigo CAPSI 2019**: *Information System for Young Football Athletes Customized Training;*
- **Artigo da FICC 2020***: Hybrid Recommendation System for Young Football Athletes Customized Training.*

Por outro lado, ao analisar o resultado final do presente projeto considera-se que o mesmo é bastante positivo.

7.1 Objetivos Alcançados

Como citado anteriormente, o principal objetivo deste projeto relacionou-se com desenvolvimento de uma aplicação desportiva, destinada à formação. De seguida, serão apresentados mais detalhadamente através da Tabela 12, todos os objetivos relacionados com a aplicação desenvolvida, associadas ao seu grau de realização. É de salientar que os objetivos identificados estão mencionados na secção **1.3 - Objetivos**, bem como, ao longo do capítulo **4 - Design da Solução**.

Tabela 12 - Objetivos do projeto e seu grau de realização

Objetivo	Grau Realização
Existência de quatro tipo de perfis únicos na aplicação	Completo
Possibilidade de uso da aplicação coletivamente ou individualmente	Completo
Gestão do seu perfil de utilizador	Completo
Possibilitar o registo e a configuração da sua equipa	Completo
Ter numa só plataforma informações de todos os jovens jogadores registados	Completo
Possuir uma interface de recolha de dados estatísticos	Completo
Existência e configuração de valores referência associados à equipa	Completo
Gestão e apresentação em formato de calendário dos jogos associados à equipa	Completo
Análise e organização dos dados recolhidos graficamente	Completo
Possibilitar a recomendação de treinos, consoante as necessidades dos diferentes jogadores	Completo
Possibilitar a criação de relatórios baseados nas performances dos jogadores	Parcialmente Completo
Oferta de uma aplicação responsiva multiplataforma	Completo

Como apresentado na tabela acima, todos os objetivos foram completados com sucesso à exceção do requisito relacionado com a componente de geração de relatórios informativos, que se encontra parcialmente completo, com cerca de 50% do trabalho realizado. Assim sendo, foi obtido uma aplicação 100% funcional que vai em conta aos requisitos impostos pelo cliente. Desse modo e à vista do mencionado na secção anterior pode-se considerar o presente projeto como concluído.

7.2 Limitações e Trabalho Futuro

Como mencionado anteriormente, a solução desenvolvida pretende ser apenas um protótipo de lançamento, desse modo, durante a fase de desenvolvimento do software e da sua avaliação identificaram-se algumas limitações e melhorias a realizar, assim como, a possibilidade de adição de novas funcionalidades, estas teriam o objetivo principal de melhorar o desempenho e usabilidade geral da aplicação.

Tendo isso em conta, apresenta-se como limitações atuais e trabalho futuro a desenvolver os seguintes pontos:

- Terminar a funcionalidade de criação de relatórios relacionados com a performance dos jogadores;
- Possibilidade de configuração dos pesos das variáveis de contexto para o jogo, assim como, do valor mínimo de recomendação;
- Atualizar o sistema de recomendação para o uso de uma técnica híbrida, como mencionado na secção, **5.1 - Sistema de Recomendação**;
- Mecanismo de feedback para os treinos recomendados;
- Possibilidade de recomendação de treinos para a equipa/posição;
- Adicionar novos indicadores estatísticos;
- Melhorar o design gráfico da aplicação;
- Desenvolvimento de um *back-office* para gestão das entidades gerais da aplicação (posições, estatísticas para performance, entre outras).

Referências

[1] S. J. Haake, "The impact of technology on sporting performance in Olympic sports," *Journal of Sports Sciences ,* pp. 1421-1431, 2009.

[2] J. Ramos, "Como a tecnologia invadiu o desporto," 15 Outubro 2015. [Online]. Available: http://visao.sapo.pt/exame/2015-10-15-Como-a-tecnologia-invadiu-o-desporto. [Acedido em 10 Fevereiro 2019].

[3] P. Matos, J. Rocha, R. Gonçalves, C. Martins, F. Santos e D. Abreu, "Hybrid Recommendation System for Young Football Athletes," Fevereiro 2019.

[4] U. Essays, "Role Of Sport In Modern Society Cultural Studies Essay," November 2018. [Online]. Available: https://www.ukessays.com/essays/cultural-studies/role-of-sport-in-modern-society-cultural-studies-essay.php. [Acedido em 2 Fevereiro 2019].

[5] T. Solutions, "Sports App Development: Grand Slam Market Opportunities and Pitfalls," 7 Fevereiro 2018. [Online]. Available: https://medium.com/swlh/sports-app-development-grand-slam-market-opportunities-and-pitfalls-33872dcc01bf. [Acedido em 2 Fevereiro 2019].

[6] M. I. Coelho, "Inteligência Artificial já sabe quem vai ser o grande vencedor do Mundial 2018," 17 Junho 2018. [Online]. Available: https://pplware.sapo.pt/informacao/ia-vencedor-do-mundial-2018/. [Acedido em 21 Fevereiro 2019].

[7] G. Giblin, E. Tor e L. Parrington, "The impact of technology on elite sports performance," Novembro 2016.

[8] O. Tevet, "A Friendly Introduction to Recommender Systems," 9 Fevereiro 2018. [Online]. Available: https://www.linkedin.com/pulse/friendly-introduction-recommender-systems-oren-tevet. [Acedido em 18 Novembro 2018].

[9] H. Reikeras, "How to Build a Content-Based Recommender System For Your Product," 1 Agosto 2018. [Online]. Available: https://www.offerzen.com/blog/how-to-build-a-content-based-recommender-system-for-your-product. [Acedido em 21 Novembro 2018].

[10] R. Souza, “Sistemas de Recomendação,” Maio 2014. [Online]. Available: https://www.ibm.com/developerworks/br/local/data/sistemas_recomendacao/index.html. [Acedido em 19 Novembro 2018].

[11] J. Bobadilla, F. Ortega, A. Hernando e A. Gutiérrez, “Recommender systems survey,” *Elsevier,* vol. 46, pp. 109-132, 2013.

[12] ReportsnReports, “Recommendation Engine Market to Grow at 40.7% CAGR to 2022,” 19 Março 2018. [Online]. Available: https://www.prnewswire.com/news-releases/recommendation-engine-market-to-grow-at-407-cagr-to-2022-677272563.html. [Acedido em 6 Dezembro 2018].

[13] J. P. Lucas, N. Luz, M. N. Moreno, R. Anacleto, A. A. Figueiredo e C. Martins, “A hybrid recommendation approach for a tourism system,” *Elsevier,* vol. 40, nº 9, pp. 3532-3550, 2013.

[14] M. J. Pazzani e D. Billsus, “Content-Based Recommendation Systems,” em *The Adaptive Web*, Heidelberg, Springer, 2007, pp. 325-341.

[15] D. Majhee, “What is the difference between content based filtering and collaborative filtering?,” 20 Junho 2017. [Online]. Available: https://www.quora.com/What-is-the-difference-between-content-based-filtering-and-collaborative-filtering. [Acedido em 22 Novembro 2018].

[16] C. Pinela, “Recommender Systems — User-Based and Item-Based Collaborative Filtering,” 6 Novembro 2017. [Online]. Available: https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f. [Acedido em 20 Novembro 2018].

[17] L. Reis, “Sistema de Recomendação Baseado em Conhecimento,” 2012.

[18] P. N. V. Kumar e D. V. R. Reddy, “A Survey on Recommender Systems (RSS) and Its Applications,” *International Journal of Innovative Research in Computer,* vol. 2, nº 8, pp. 5254-5260, 2014.

[19] L. Reis, “Sistema de Recomendação Baseado em Conhecimento,” Coimbra, 2012.

[20] G. Guo, “Resolving Data Sparsity and Cold Start,” em *User Modeling, Adaptation, and Personalization*, Berlin, Heidelberg, Springer, 2012, pp. 361-364.

[21] P. Kordík, “Machine Learning for Recommender systems — Part 1 (algorithms, evaluation and cold start),” 2018 Junho 3. [Online]. Available: https://medium.com/recombee-blog/machine-learning-for-recommender-systems-

part-1-algorithms-evaluation-and-cold-start-6f696683d0ed. [Acedido em 25 Novembro 2018].

[22] I. Medeiros, "Estudo sobre sistemas de recomendação colaborativos," Recife, 2013.

[23] D. Bokdea, S. Giraseb e D. Mukhopadhyay, "Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey," *Elsevier,* vol. 49, pp. 136-146, 2015.

[24] Sigmoidal, "Recommendation Systems – How Companies are Making Money," [Online]. Available: https://sigmoidal.io/recommender-systems-recommendation-engine. [Acedido em 30 Novembro 2018].

[25] Netflix, "O que é a Netflix?," [Online]. Available: https://help.netflix.com/pt-pt/node/412. [Acedido em 25 Novembro 2018].

[26] N. McAlone, "Why Netflix thinks its personalized recommendation engine is worth $1 billion per year," 14 Junho 2016. [Online]. Available: https://www.businessinsider.com/netflix-recommendation-engine-worth-1-billion-per-year-2016-6. [Acedido em 25 Novembro 2018].

[27] I. MacKenzie, C. Meyer e S. Noble, "How retailers can keep up with consumers," Outubro 2013. [Online]. Available: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers. [Acedido em 25 Novembro 2018].

[28] N. H. Carlos A. Gomez-Uribe, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," *ACM Transactions on Management Information Systems (TMIS),* vol. 6, nº 4, pp. 1-19, 2016.

[29] G. Linden, B. Smith e J. York, "Item-to-Item Collaborative Filtering," *Amazon.com,* pp. 76-79.

[30] T. Krawiec, "The Amazon Recommendations Secret to Selling More Online," [Online]. Available: http://rejoiner.com/resources/amazon-recommendations-secret-selling-online/. [Acedido em 25 Novembro 2018].

[31] J. Zeleznikow, C. MacMahon e T. Barnett, "Providing automated decision support for elite," *International Symposium of Computer Science in Sport,* pp. 240-248, 2009.

[32] I. A. Fariñas, "Who is going to win the next World Cup?," 24 Abril 2018. [Online]. Available: https://www.minsait.com/sites/default/files/newsroom_documents/quien_va_a_ganar_el_proximo_mundial_de_futbol._los_datos_nos_dan_las_opciones_en.pdf. [Acedido em 30 Janeiro 2019].

[33] G. Kumar, “Machine Learning for Soccer Analytics,” Cambridge University Press, 2013.

[34] R. E. Wright, J. Silva e I. Kaynar-Kabul, “Shot Recommender System for NBA Coaches,” *Shot Recommender System for NBA Coaches,* p. 4.

[35] B. Smyth e P. Cunningham, “A Novel Recommender System for helping Marathoners to,” *Insight Centre for data Analytics,* 5 Julho 2017.

[36] TacticalSoccer, “Tactical Soccer,” [Online]. Available: https://www.tacticalsoccer.pt/. [Acedido em 19 Janeiro 2019].

[37] MyCoachFootball, “MyCoachFootball,” [Online]. Available: https://www.mycoachfootball.com/en/. [Acedido em 21 Dezembro 2018].

[38] D. d. Treinador, “Dossier do Treinador,” [Online]. Available: http://www.dossierdotreinador.com/. [Acedido em 19 Janeiro 2019].

[39] SportsEasy, “SportsEasy,” [Online]. Available: https://www.sporteasy.net/pt/home/. [Acedido em 21 Janeiro 2019].

[40] T. S. Manager, “Soccer Coach,” Team Sports Manager, [Online]. Available: http://www.teamsportsmanager.com. [Acedido em 21 Janeiro 2019].

[41] DRM Associates, “Value Analysis and Function Analysis System Technique,” 2016. [Online]. Available: http://www.npd-solutions.com/va.html. [Acedido em 16 Fevereiro 2019].

[42] Koen, Peter et al, “Fuzzy Front End: Effective Methods, Tools, and Techniques,” pp. 5-36, 2002.

[43] S. Nicola, E. P. Ferreira e J. J. P. Ferreira, “International Journal of Information Technology &,” vol. III, nº 11, pp. 661-703, 2012.

[44] T. Woodall, “Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis,” *Academy of Marketing Science Review,* p. 2, Janeiro 2003.

[45] V. A. Zeithaml, “Consumer Perceptions of Price, Quality and Value: A Means-End Model and Synthesis of Evidence,” *Journal of Marketing,* p. 14, 1988.

[46] A. Lindgreena e F. Wynstra, “Value in business markets: What do we know? Where are we going?,” *Industrial Marketing Management,* pp. 732-748, 2005.

[47] M. Gabry, “O que é proposta de valor e por que é tão importante para seu negócio?,” 11 Janeiro 2016. [Online]. Available:

http://www.administradores.com.br/artigos/negocios/o-que-e-proposta-de-valor-e-por-que-e-tao-importante-para-seu-negocio/92725/. [Acedido em 11 Fevereiro 2019].

[48] L. Cruz, “Proposta de Valor,” 25 Março 2018. [Online]. Available: http://knoow.net/cienceconempr/marketing/proposta-de-valor/. [Acedido em 12 Fevereiro 2019].

[49] SaldoPositivo, “O que é o Business Canvas Model?,” 9 Setembro 2016. [Online]. Available: http://saldopositivo.cgd.pt/empresas/business-canvas-model/. [Acedido em 16 Fevereiro 2019].

[50] U. Eriksson, “Functional vs Non Functional Requirements,” 5 Abril 2012. [Online]. Available: https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/. [Acedido em 9 Fevereiro 2019].

[51] P. Eeles, “Capturing Architectural Requirements,” 15 Novembro 2005. [Online]. Available: https://www.ibm.com/developerworks/rational/library/4706.html. [Acedido em 14 Fevereiro 2019].

[52] Quatinetwork, “Modelo de domínio,” [Online]. Available: https://quatinetwork.files.wordpress.com/2012/04/modelo_dominio.pdf. [Acedido em 21 Fevereiro 2019].

[53] BPM Flow, “4+1 Architecture Models,” Julho 2018. [Online]. Available: https://bpmflow.gitbook.io/project/engineering-models/4+1-architecture-models. [Acedido em 15 Fevereiro 2019].

[54] E. Williams, “What is Deployment in Software,” 29 Agosto 2019. [Online]. Available: https://pdf.wondershare.com/business/what-is-software-deployment.html. [Acedido em 22 Setembro 2019].

[55] L. S. Inc., “O que é um modelo de banco de dados?,” Lucid Software Inc., [Online]. Available: https://www.lucidchart.com/pages/pt/o-que-e-um-modelo-de-banco-de-dados?a=0. [Acedido em 23 Fevereiro 2019].

[56] M. Castro, “Resumo: Architectural Blueprints — The “4+1” View Model of Software Architecture,” 5 Janeiro 2016. [Online]. Available: https://medium.com/@matheus_ortsac/resumo-architectural-blueprints-the-4-1-view-model-of-software-architecture-f03adf3724d2. [Acedido em 22 Fevereiro 2019].

[57] SmartBear, “What is Microservices Architecture?,” [Online]. Available: https://smartbear.com/learn/api-design/what-are-microservices/. [Acedido em 23 Fevereiro 2019].

[58] OpusSoftware, “Micro Serviços: Qual a diferença para a Arquitetura Monolítica?,” [Online]. Available: https://www.opus-software.com.br/micro-servicos-arquietura-monolitica/. [Acedido em 23 Fevereiro 2019].

[59] P. Matos, J. Rocha, R. Gonçalves, A. Almeida, F. Santos, D. Abreu e C. Martins, “Smart Coach—A Recommendation System for Young Football Athletes,” *International Symposium on Ambient Intelligence,* pp. 171-178, 23 Junho 2019.

[60] Microsoft, “Introduction to ASP.NET Identity,” 1 Janeiro 2019. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity. [Acedido em 1 Outubro 2019].

[61] Microsoft, “Tutorial: Criar uma aplicação ASP.NET no Azure com a Base de Dados SQL,” 25 Janeiro 2018. [Online]. Available: https://docs.microsoft.com/pt-pt/azure/app-service/app-service-web-tutorial-dotnet-sqldatabase#update-app-with-code-first-migrations. [Acedido em 1 Outubro 2019].

[62] P. Filler, “Chosen,” 11 Julho 2017. [Online]. Available: https://harvesthq.github.io/chosen/. [Acedido em 1 Outubro 2019].

[63] Saganic, “Countrypicker,” 2017. [Online]. Available: https://github.com/Saganic/country-picker. [Acedido em 2 Outubro 2019].

[64] D. Grossman, “Date Range Picker,” 2012. [Online]. Available: http://www.daterangepicker.com/. [Acedido em 2 Outubro 2019].

[65] SpryMedia, “DataTables,” 2017. [Online]. Available: https://datatables.net/. [Acedido em 2 Outubro 2019].

[66] FullCalendar, “The most popular full-sized JavaScript Calendar,” 2009. [Online]. Available: https://fullcalendar.io/. [Acedido em 2 Outubro 2019].

[67] Highcharts, “Make your data come alive - Highcharts,” 2009. [Online]. Available: https://www.highcharts.com/. [Acedido em 2 Outubro 2019].

[68] Swimmwatch, “Smart Customizable jQuery Tabs Plugin - Pretty Tabs,” 2016. [Online]. Available: https://www.jqueryscript.net/other/Smart-Customizable-jQuery-Tabs-Plugin-Pretty-Tabs.html. [Acedido em 2 Outubro 2019].

[69] jQuery, “Dialog,” [Online]. Available: https://jqueryui.com/dialog/#modal-form. [Acedido em 3 Outubro 2019].

[70] T. d. Software, “Processo de Testes,” 22 Abril 2011. [Online]. Available: https://testesw.wordpress.com/processo-de-testes/. [Acedido em 5 Outubro 2019].

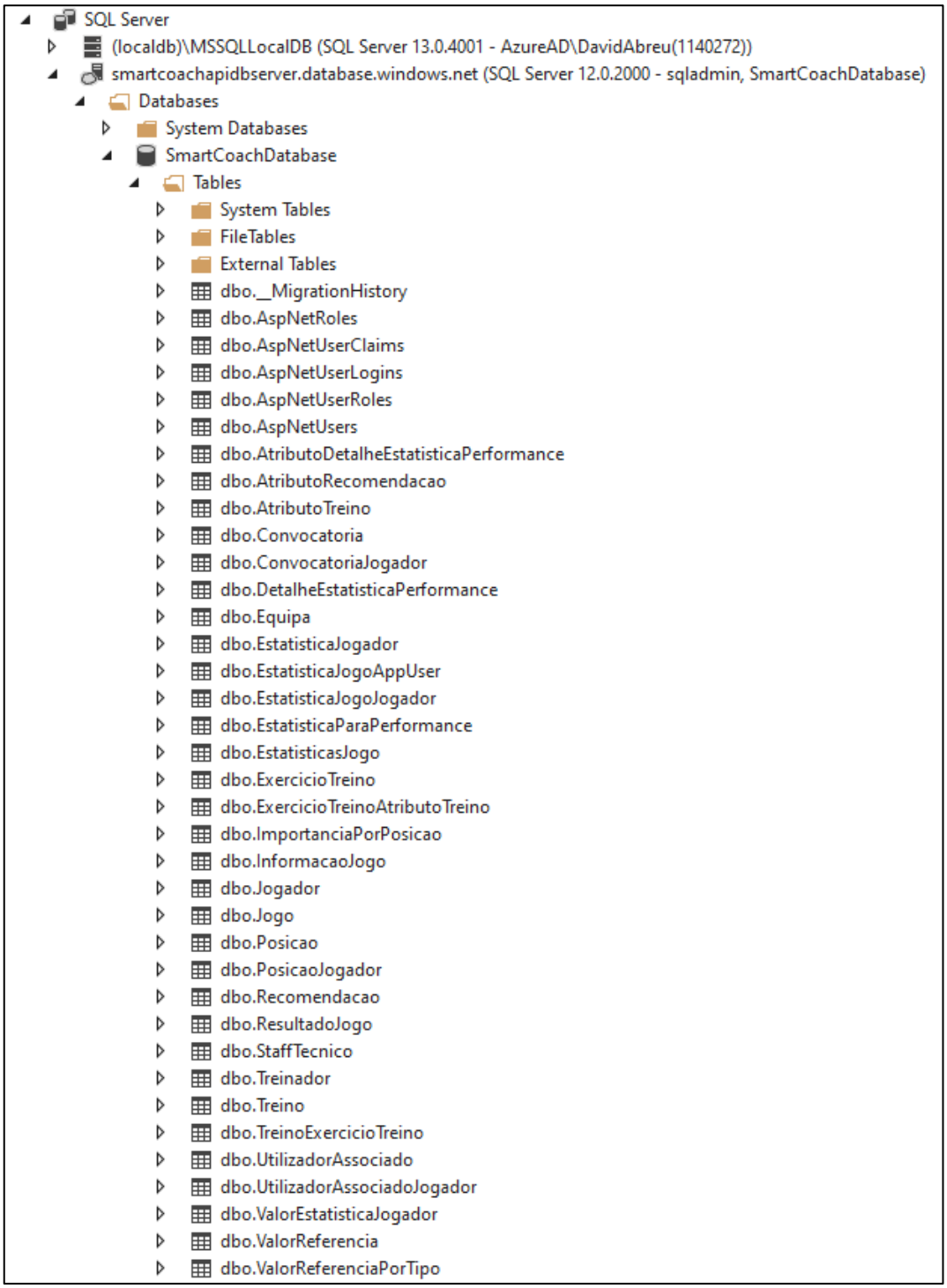
[71] J. Aparecido, “Teste de integração na prática,” 2014. [Online]. Available: https://www.devmedia.com.br/teste-de-integracao-na-pratica/31877. [Acedido em 4 Outubro 2019].

[72] Servy, “c# takes more time to do first execution,” 15 Março 2015. [Online]. Available: https://stackoverflow.com/questions/28950581/c-sharp-takes-more-time-to-do-first-execution. [Acedido em 9 Outubro 2019].

[73] M. Aboullaite, “Understanding JIT compiler (just-in-time compiler),” 31 Agosto 2017. [Online]. Available: https://aboullaite.me/understanding-jit-compiler-just-in-time-compiler/. [Acedido em 9 Outubro 2019].

[74] Google, “Criar fabulosos formulários,” [Online]. Available: https://www.google.com/forms/about/. [Acedido em 6 Outubro 2019].

Anexo 1 – Tabelas da *SmartCoachDatabase*



Anexo 2 – Testes de Sistema

Teste de Sistema Nº1	
Descrição:	Registar-se na aplicação e configurar os valores referência.
Resultado esperado:	Um treinador regista-se no sistema e configura os valores referencias para a sua equipa.
Resultado obtido:	PASSOU

Teste de Sistema Nº2	
Descrição:	Jogador realiza o Login na aplicação e de seguida, visualiza o seu calendário desportivo.
Resultado esperado:	Login é realizado com sucesso identificando que o utilizador é um jogador, e de seguida, é apresentado o seu calendário desportivo.
Resultado obtido:	PASSOU

Teste de Sistema Nº3	
Descrição:	Treinador atualiza os dados da sua equipa e convida novos jogadores para pertencer à mesma.
Resultado esperado:	Os dados da equipa são atualizados e são convidados dois jogadores para a equipa.
Resultado obtido:	PASSOU

Teste de Sistema Nº4	
Descrição:	Registar um jogo e inserir estatísticas para o jogo registado.
Resultado esperado:	O treinador regista um jogo na aplicação, e depois, recolhe estatísticas individuais de um jogador para o jogo criado.
Resultado obtido:	PASSOU

Teste de Sistema Nº5	
Descrição:	Gerar um treino recomendado e visualizar os detalhes de cada um dos exercícios presentes no treino.
Resultado esperado:	Criação de um treino recomendado consoante os valores provenientes do sistema de recomendação e visualização no serviço externo *SmartCoachManager* dos detalhes dos exercícios de treino gerados.
Resultado obtido:	PASSOU

Teste de Sistema Nº6	
Descrição:	Gerenciar os jogadores associados e visualizar *dashboard* estatístico.
Resultado esperado:	Um utilizador do sistema elimina e adiciona novos jogadores associados, e de seguida, visualiza estatísticas sobre os mesmos, na componente do *dashboard*.
Resultado obtido:	PASSOU

Anexo 3 – Testes de Aceitação

Teste de Aceitação Nº1 – Registo	
Descrição:	Um utilizador não autenticado pretende-se registar como treinador na aplicação com sucesso.
Resultado esperado:	O utilizador não autenticado regista-se no sistema como treinador com sucesso.
Resultado obtido:	PASSOU

Teste de Aceitação Nº2 – Login	
Descrição:	Um utilizador do sistema pretende realizar o Login com sucesso na aplicação.
Resultado esperado:	O utilizador não autenticado torna-se utilizador do sistema.
Resultado obtido:	PASSOU

Teste de Aceitação Nº3 – Gerir Conta	
Descrição:	O treinador pretende atualizar certos campos do seu perfil.
Resultado esperado:	O utilizador atualiza a sua data de nascimento, assim como, a sua password.
Resultado obtido:	PASSOU

Teste de Aceitação Nº4 – Configurar Equipa	
Descrição:	O treinador pretende configurar certos campos da sua equipa.
Resultado esperado:	O treinador atualiza o nome e descrição da sua equipa.
Resultado obtido:	PASSOU

Teste de Aceitação Nº5 – Gerir Jogadores	
Descrição:	O treinador convida um novo jogador (via e-mail) para ingressar na sua equipa.
Resultado esperado:	É enviado um convite para o e-mail do jogador inserido para adesão à equipa do treinador.
Resultado obtido:	PASSOU

Teste de Aceitação Nº6 – Gerir Staff Técnico	
Descrição:	O treinador convida um novo membro do staff técnico (via e-mail) para ingressar na sua equipa.
Resultado esperado:	É enviado um convite para o e-mail do membro do staff inserido para adesão à equipa do treinador.
Resultado obtido:	PASSOU

Teste de Aceitação Nº7 – Consultar Calendário	
Descrição:	O treinador pretende visualizar o calendário desportivo da sua equipa.
Resultado esperado:	O sistema apresenta a agenda dos jogos da equipa do treinador em formato calendário.
Resultado obtido:	PASSOU

Teste de Aceitação Nº8 – Gestão de jogos	
Descrição:	O treinador pretende registar um novo jogo para a sua equipa.
Resultado esperado:	O sistema consoante os dados introduzidos pelo treinador regista um novo jogo.
Resultado obtido:	PASSOU

Teste de Aceitação Nº9 – Configurar Valores Referência	
Descrição:	O treinador pretende atualizar valores referência associados a certas estatísticas.
Resultado esperado:	O sistema permite a atualização dos valores referência desejados.
Resultado obtido:	PASSOU

Teste de Aceitação Nº10 – Inserir Estatísticas Jogo	
Descrição:	O treinador pretende inserir dados estatísticos sobre um jogador para o jogo selecionado.
Resultado esperado:	O sistema guarda os dados que o treinador inseriu para o conjunto jogador/jogo.
Resultado obtido:	PASSOU

Teste de Aceitação Nº11 – Gerar Treino Recomendado	
Descrição:	O treinador pretende gerar um treino recomendado para um jogador específico.
Resultado esperado:	O sistema através do mecanismo de recomendação gere um treino específico face ao jogador selecionado.
Resultado obtido:	PASSOU

Teste de Aceitação Nº12 – Visualizar *Dashboard*	
Descrição:	O treinador pretende visualizar gráficos e indicadores estatísticos coletivos e individuais.
Resultado esperado:	O sistema apresenta através da página do *dashboard*, gráficos coletivos e individuais.
Resultado obtido:	PASSOU

Anexo 4 – Resultados tempo do algoritmo

Tempo obtidos ao executar o algoritmo de recomendação, em milissegundos (ms)

Nº Execução	Tempo (*ms*)
1	602
2	408
3	401
4	397
5	386
6	391
7	392
8	390
9	394
10	383
11	399
12	388
13	458
14	392
15	381
16	387
17	387
18	398
19	382
20	384
21	406
22	376
23	403
24	388
25	390

Anexo 5 – Inquérito de Satisfação

Inquérito de satisfação SmartCoach/SmartCoachManager

Este questionário é realizado de forma anónima, e tem o objetivo de recolher a opinião sobre a experiência de utilização com os sistemas SmartCoach e SmartCoachManager

*Obrigatório

1. **Qual o seu cargo no clube?** *
   Marcar apenas uma oval.
   - ◯ Treinador
   - ◯ Jogador
   - ◯ Membro Staff Técnico
   - ◯ Outra: _____

2. **A aplicação tem todas as funcionalidades desejadas?** *
   Marcar apenas uma oval.
   - ◯ Discordo completamente
   - ◯ Discordo
   - ◯ Sem opinião
   - ◯ Concordo
   - ◯ Concordo completamente

3. **Considera a aplicação simples e intuitiva?** *
   Marcar apenas uma oval.
   - ◯ Discordo completamente
   - ◯ Discordo
   - ◯ Sem opinião
   - ◯ Concordo
   - ◯ Concordo completamente

4. **A aplicação possui uma boa apresentação e design?** *
   Marcar apenas uma oval.
   - ◯ Discordo completamente
   - ◯ Discordo
   - ◯ Sem opinião
   - ◯ Concordo
   - ◯ Concordo completamente

5. **Considera a aplicação com tempo de resposta curto?** *
Marcar apenas uma oval.

- ◯ Discordo completamente
- ◯ Discordo
- ◯ Sem opinião
- ◯ Concordo
- ◯ Concordo completamente

Componente SmartCoach

Plataforma associada à gestão da equipa, recolha de dados estatísticos e recomendação de treinos.

6. **A gestão da equipa tem as funcionalidades pretendidas?** *
Marcar apenas uma oval.

- ◯ Discordo completamente
- ◯ Discordo
- ◯ Sem opinião
- ◯ Concordo
- ◯ Concordo completamente

7. **Considera a configuração dos valores referência uma mais-valia?** *
Marcar apenas uma oval.

- ◯ Discordo completamente
- ◯ Discordo
- ◯ Sem opinião
- ◯ Concordo
- ◯ Concordo completamente

8. **Considera a recolha de estatísticas de jogo completa e intuitiva?** *
Marcar apenas uma oval.

- ◯ Discordo completamente
- ◯ Discordo
- ◯ Sem opinião
- ◯ Concordo
- ◯ Concordo completamente

9. **As recomendações de treino geradas são precisas face às necessidades do jogador?** *
Marcar apenas uma oval.

- ◯ Discordo completamente
- ◯ Discordo
- ◯ Sem opinião
- ◯ Concordo
- ◯ Concordo completamente

10. **O dashboard apresenta os gráficos/indicadores estatísticos desejados?** *
Marcar apenas uma oval.

- Discordo completamente
- Discordo
- Sem opinião
- Concordo
- Concordo completamente

Componente SmartCoachManager

Plataforma associada à gestão e classificação de treinos e exercícios de treino.

11. **A criação de exercícios de treino tem todos os parâmetros pretendidos?** *
Marcar apenas uma oval.

- Discordo completamente
- Discordo
- Sem opinião
- Concordo
- Concordo completamente

12. **Considera a classificação do exercício com base em atributos uma mais-valia?** *
Marcar apenas uma oval.

- Discordo completamente
- Discordo
- Sem opinião
- Concordo
- Concordo completamente

13. **Considera o processo de criação de um treino (manual ou por recomendação) simples e intuitivo?** *
Marcar apenas uma oval.

- Discordo completamente
- Discordo
- Sem opinião
- Concordo
- Concordo completamente

14. **Considera a calendarização dos planos de treino adequada?** *
Marcar apenas uma oval.

- Discordo completamente
- Discordo
- Sem opinião
- Concordo
- Concordo completamente

Satisfação geral

15. **A aplicação poderia ser útil na sua equipa?** *

Marcar apenas uma oval.

- ◯ Discordo completamente
- ◯ Discordo
- ◯ Sem opinião
- ◯ Concordo
- ◯ Concordo completamente

16. **Como classificaria a aplicação?** *

Marcar apenas uma oval.

- ◯ Muito má
- ◯ Má
- ◯ Sem opinião
- ◯ Boa
- ◯ Muito boa

17. **Recomendaria a aplicação a outros utilizadores?** *

Marcar apenas uma oval.

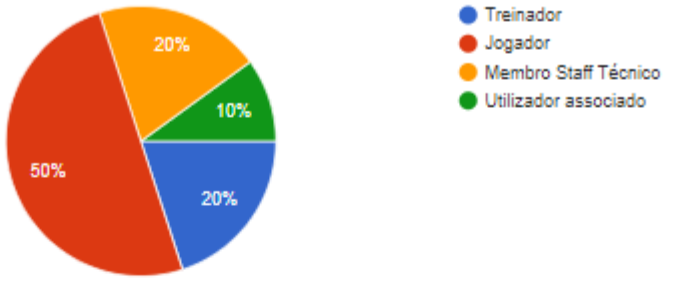
- ◯ Não
- ◯ Talvez
- ◯ Sim

18. **Possíveis melhorias?**

Anexo 6 – Respostas ao Inquérito

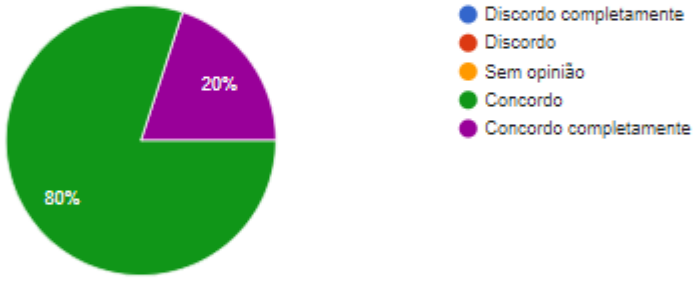
Qual o seu cargo no clube?

10 respostas



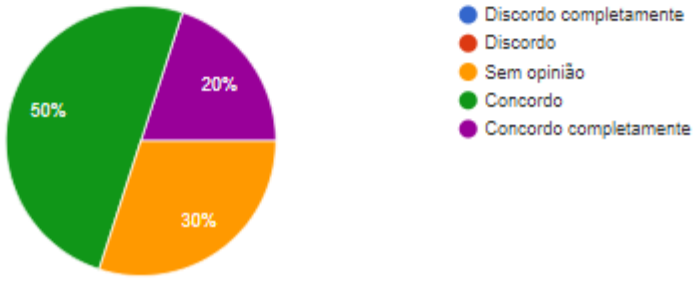
A aplicação tem todas as funcionalidades desejadas?

10 respostas



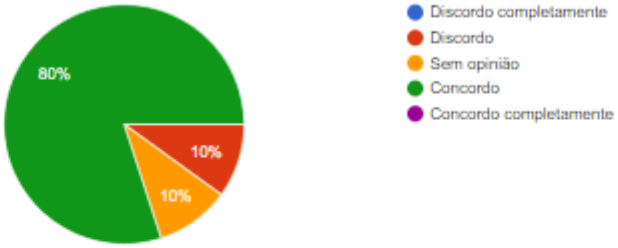
Considera a aplicação simples e intuitiva?

10 respostas



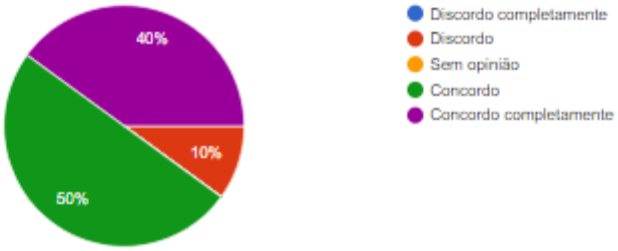
A aplicação possui uma boa apresentação e design?

10 respostas



Considera a aplicação com tempo de resposta curto?

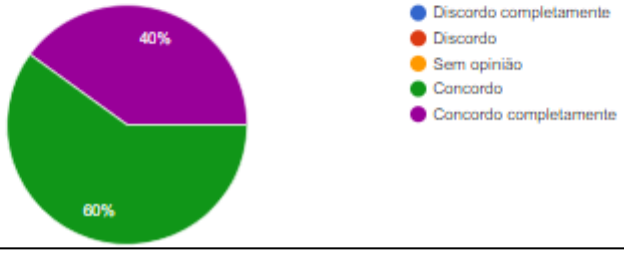
10 respostas



Componente SmartCoach

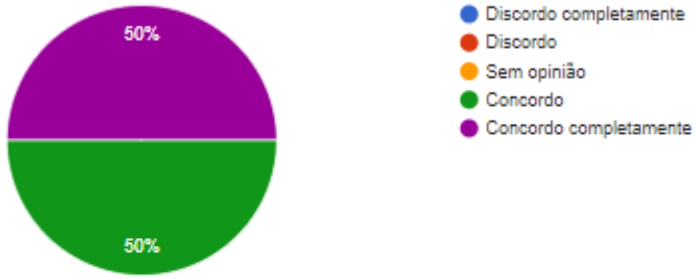
A gestão da equipa tem as funcionalidades pretendidas?

10 respostas



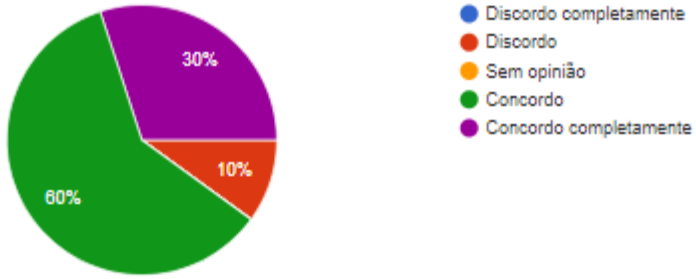
Considera a configuração dos valores referência uma mais-valia?

10 respostas



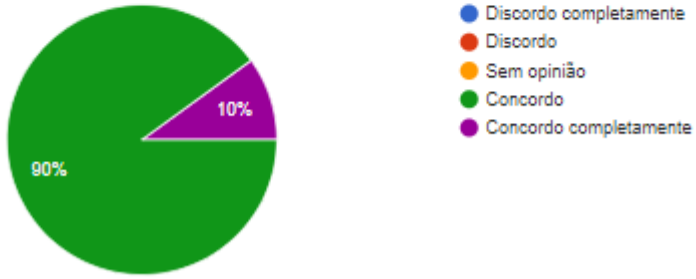
Considera a recolha de estatísticas de jogo completa e intuitiva?

10 respostas



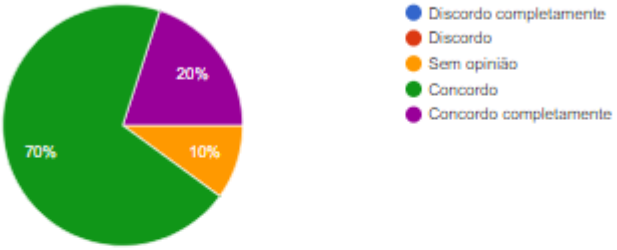
As recomendações de treino geradas são precisas face às necessidades do jogador?

10 respostas



O dashboard apresenta os gráficos/indicadores estatísticos desejados?

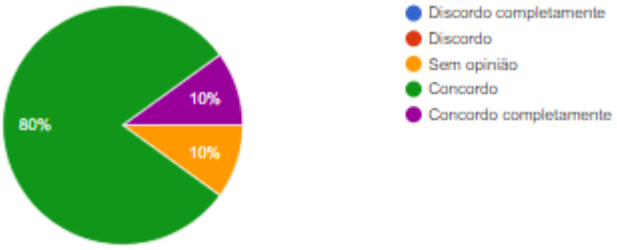
10 respostas



Componente SmartCoachManager

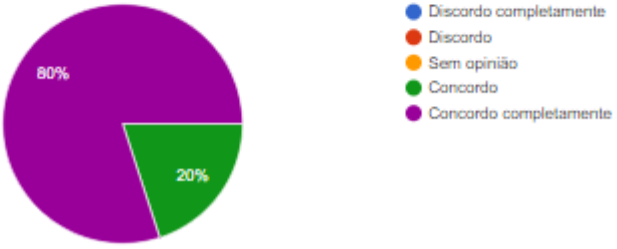
A criação de exercícios de treino tem todos os parâmetros pretendidos?

10 respostas



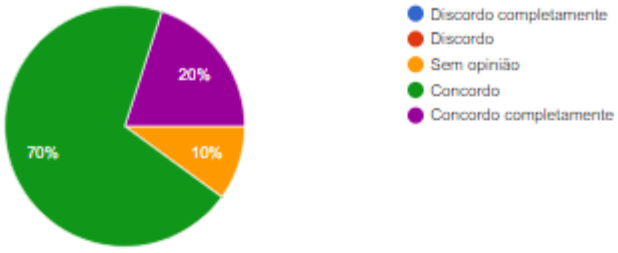
Considera a classificação do exercício com base em atributos uma mais-valia?

10 respostas



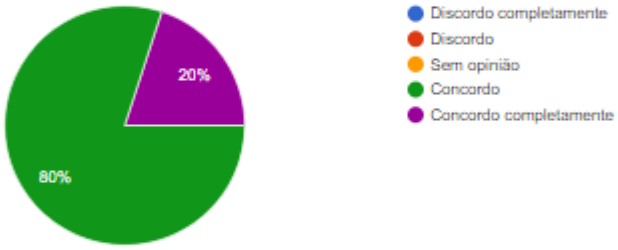
Considera o processo de criação de um treino (manual ou por recomendação) simples e intuitivo?

10 respostas



Considera a calendarização dos planos de treino adequada?

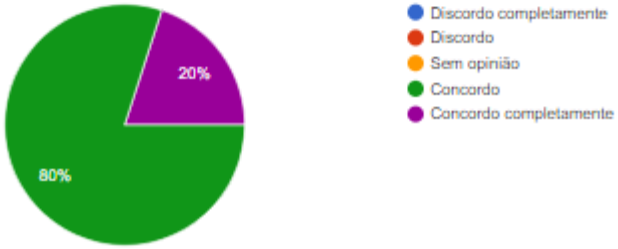
10 respostas



Satisfação geral

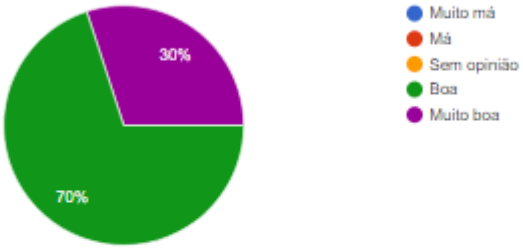
A aplicação poderia ser útil na sua equipa?

10 respostas



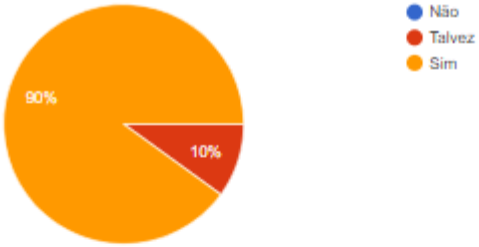
Como classificaria a aplicação?

10 respostas



Recomendaria a aplicação a outros utilizadores?

10 respostas



Possíveis melhorias?

3 respostas

Design, exercícios mais variados, intuitividade.
Melhor design
Com a individualidade ser capaz de criar estatística coletiva,exercícios específicos para a equipa e não individualmente.