**Relatório Final**

PlayersNet

Disciplina: Projeto e Seminário Curso: Licenciatura em Engenharia Informática e de Computadores

Arnaldo Tema No aluno: 39342 Lisboa, 07 de setembro de 2018 Orientador: Eng.o Carlos Martins

0

1

Índice

Resumo ........................................................................................................................................................... 2

Agradecimentos .............................................................................................................................................. 3

Introdução ...................................................................................................................................................... 4

Solução Funcional ........................................................................................................................................... 6

MEAN Stack .................................................................................................................................................... 8

Web Crawler ................................................................................................................................................. 10

Gamification ................................................................................................................................................. 12

Arquitetura ................................................................................................................................................... 15

Descrição técnica .......................................................................................................................................... 17

Interface da plataforma ................................................................................................................................ 24

Conclusões .................................................................................................................................................... 42

Referências ................................................................................................................................................... 46

2

Resumo

A Players Net é uma plataforma nova no mercado do desporto em Portugal. É uma rede social semelhante ao LinkedIn cujo objetivo é proporcionar a todos os clubes em Portugal uma ferramenta online que permita encontrar atletas, treinadores e fisioterapeutas, em qualquer contexto, permitindo a personalização da procura, tendo em conta os seus critérios de deteção de talento.

Para os atletas fornece uma ferramenta que proporciona uma forma de autopromoção perante os clubes, bem como uma vertente interativa e competitiva com os restantes atletas que utilizam a Players Net.

A visão deste projeto é desenvolver o desporto em Portugal fornecendo as ferramentas necessárias de promoção e deteção de talentos aos principais intervenientes do Desporto: Clubes, Atletas, Treinadores e Fisioterapeutas. Como se trata de um projeto de caris académico, vamos usar apenas o futebol como desporto “pivot”.

A propósito essencial é proporcionar a atletas e clubes um meio de interação onde os atletas possam promover as suas aptidões e competências, e os clubes possam detetar todos os talentos existentes em Portugal e futuramente no estrangeiro.

3

Agradecimentos

Quero agradecer ao meu grande amigo Diogo Pires, que me ajudou de forma notória no desenvolvimento da parte servidora desta aplicação, bem como na conceção da lógica e modelo de negócio da mesma. Deixo também os meus mais sinceros agradecimentos ao meu orientador - professor Carlos Martins- , por me proporcionar as melhores e mais confortáveis condições desde o início do projeto, sendo que foi difícil conseguir um orientador cujo core tecnológico se assemelhasse ao máximo com o da plataforma, voluntariando- se sempre para me ajudar no que fosse preciso.

4

Introdução

Num mundo onde a automaticidade e rapidez de processos nunca teve tanto valor, há a recorrente necessidade agilizar processos que carecem de um bom investimento de tempo e trabalho. Foi nesse sentido que cresceu a conhecida plataforma LinkedIn e é igualmente nesse sentido que nasce a Players Net. A plataforma ataca duas grandes vertentes:

Scouting

Para um clube desportivo, fazer scouting de um atleta é um processo trabalhoso e utiliza imensos recursos. Existem base de dados com informação dos atletas federados das variadas modalidades desportivas existentes, no entanto, essa informação não só raramente está atualizada para os escalões menos seniores ou modalidades menos conhecidas, como não há uma constante validação dos atletas desses mesmos dados (porque não é do seu interesse) nem possui dados que não se consiga obter numa típica “folha de avaliação” desportiva.

Com isto, os clubes têm de ter recursos exclusivamente alocados para scouting. Viajam pelo país para testemunhar presencialmente a performance dos atletas, enviam emails, fazem telefonemas, encontram-se com pessoas que possam também dar a sua opinião acerca do atleta. Tudo para, no final, terem um feedback mais real possível.

Com a Players Net, os atletas atualizam os seus dados, sendo eles responsáveis pela informação introduzida na plataforma. Para além disso, outros atletas, treinadores ou clubes podem dar o seu parecer acerca da informação introduzida pelo atleta, através de recomendações escritas, votos, confirmações de “skills”, entre outros. O que faz com que a informação alcance o nível mais alto possível de veracidade.

Os dados estatísticos são automaticamente atualizados pela plataforma, que recorre a outras fontes para o fazer, mas o utilizador pode e deve confirmar esses dados. Esses dados não só são confirmados pelo jogador como, sempre que há incoerência com outros dados obtidos através das fontes nas quais a Players Net confia, há um sistema de validação que só o rotula (através de um símbolo que representa um dado validado pela Players Net) como válido quando todos os intervenientes desse acontecimento também o validarem.

Por fim, há um sistema avançado de pesquisa de atletas que permite que o responsável pelo scouting de jogadores de um determinado clube - juntamente com o nível de validação de dados que a plataforma fornece, a atualização constante de informação por parte dos jogadores e outros intervenientes e o feedback de outras entidades através de recomendações escritas ou votos - faça o seu trabalho de scouting em meia dúzia de cliques.

Promoção dos atletas

Da mesma forma que o LinkedIn permite que os seus utilizadores se promovam perante os recrutadores de outras empresas, a Players Net permite que os seus atletas se promovam perante os recrutadores de outros clubes. Especialmente atletas com menos “visibilidade automática”.

A possibilidade de os jogadores introduzirem vídeos ou outro tipo de informação digital, terem em destaque nas suas páginas as suas grandes conquistas individuais ou coletivas, os seus dados estatísticos e não estatísticos e partilharem com toda a comunidade do seu setor desportivo uma informação constantemente atualizada, são tudo fatores fundamentais para a sua promoção como atleta. Não há outra plataforma que permita este tipo de promoção pessoal a nível desportivo.

**Conteúdo do relatório**

Neste documento encontrar-se-á uma explicação detalhada do processo de desenvolvimento deste projeto. Não estando organizado de uma forma cronológica, o relatório aborda os aspetos essenciais para a compreensão da metodologia de trabalho utilizada, abordagem técnica, workflow funcional da plataforma, explicação da escolha das tecnologias, abordagem algorítmica e também uma partilha de opinião pessoal no desenvolvimento de software e plataformas web.

5

No 2o Capítulo, para além da terminologia do relatório, está enunciado em síntese o processamento de dados automático que a Players Net oferece, bem como os aspetos funcionais e enumeração de páginas da plataforma.

No Capítulo seguinte é explicada a escolha da abordagem tecnológica para o desenvolvimento deste projeto. Numa vasta gama de linguagens e “Stacks de desenvolvimento” possíveis, aqui encontram-se as razões pelas quais a “MEAN Stack” foi a escolhida, incluindo as suas vantagens num ambiente que efetua “Web crawling”, benefícios em utilizar um sistema de base de dados não relacional (NoSQL) e o porquê de se decidir apresentar toda essa informação através de uma linguagem que essencialmente sustenta as “SPAs” - Angular.

Um “Web Crawler”, por ser tipicamente concebido para adquirir dados que não nos são diretamente fornecidos (por exemplo, através de um web service), é uma ferramenta algo polémica no desenvolvimento de software. Esta questão e o seu conceito geral encontram-se melhor aprofundados no 4o Capítulo, seguido duma abordagem ao conceito de “Gamification” utilizado na Players Net presente no Capítulo seguinte.

O 6o e 7o Capítulos falam sobre a arquitetura e descrição técnica respetivamente. Neste último está descrito com detalhe a solução técnica da Players Net, com exemplos práticos, definição e descrição dos componentes de serviço e da interface de utilizador, sendo que o seu detalhe funcional, acompanhado de um exemplo de manual de utilizador ilustrado com imagens da plataforma, encontra-se no penúltimo Capítulo deste relatório.

Por fim, no último Capítulo, estão as conclusões tiradas no final da elaboração deste projeto, onde são descritas as lições aprendidas com o seu desenvolvimento, o que poderia ter sido diferente, em que medida foram atingidos os objetivos deste projeto e as perspetivas para o seu trabalho futuro.

6

Solução Funcional

A plataforma oferece a seguinte solução funcional:

**• Ações de utilizador**

- Criação e edição de uma conta pessoal. - Criação e edição de um perfil de utilizador (jogador) - Criação e edição de recomendações escritas nas páginas de perfil de jogador, treinador ou clube. - Votar em “skills” no perfil de um utilizador. - Adicionar conteúdo de multimédia (vídeos ou imagens) ao próprio perfil de utilizador. - Editar informação estatística dos jogos. - Validar informação estatística dos jogos. - Seguir um determinado perfil de utilizador. - Fazer uma pesquisa geral de todos os perfis de utilizador ativos na plataforma. - Fazer uma pesquisa filtrada de todos os jogadores ativos ativos na plataforma. - Como clube, criar um atleta privado e atribuir avaliações simples ou avançadas para esse mesmo atleta. - Como clube, criar um evento de treinos de captação. - Como clube, filtrar quais os perfis de utilizador podem ter acesso ao evento de captação e/ou recebem uma notificação do evento. - Comentar, gostar ou partilhar um conteúdo de “Media” na página de perfil de um utilizador ou na página de feed principal.

**• Ações automaticamente feitas pela plataforma:**

- Obtenção de todas as “Competições” de futebol nacional, independentemente do escalão. - Obtenção de todas os clubes presentes nas competições obtidas. - Obtenção de todos os jogadores presentes nas equipas obtidas. - Obtenção de todos os jogos cuja “Competição” tenha sido previamente obtida - Obtenção constante de novos jogos e respetivos dados estatísticos. - Atualização de todos os dados estatísticos detetados nos jogos obtidos. - Atribuição de títulos (“Badges”) consoante um determinado acontecimento estatístico de um jogador. - Obtenção constante de informação de jogos futuros. - Atualização de todas as entidades com base na informação obtida acerca dos jogos futuros. - Atualização constante de rankings estatísticos e não estatísticos dos utilizadores.

**• Páginas da plataforma:**

- Inscrição/ Login. - Criação de conta. - Feed de notícias. - Perfil de jogador. - Edição de jogador. - Conteúdo “Media” de jogador. - Estatísticas de jogador. - Perfil de treinador. - Conteúdo “Media” de treinador. - Estatísticas de treinador. - Perfil de clube. - Conteúdo “Media” de clube.

7

- Estatísticas de clube. - Plantel de clube. - Jogo. - Pesquisa filtrada de jogadores. - Criação/ Edição de jogador privado de clube. - Lista de títulos (“Badges”). - Lista de recomendações.

8

MEAN Stack

O desenvolvimento de aplicativos usando MEAN é simples e rápido porque os developers podem programar numa só linguagem, ou seja, JavaScript para client-side e server-side - uma linguagem para back-end e front-end com as seguintes características:

o Flexibilidade

MEAN permite que se teste facilmente um aplicativo na plataforma de nuvem após a conclusão bem-sucedida de um processo de desenvolvimento. As aplicações podem ser facilmente desenvolvidas, testadas e introduzidas no servidor. Permite também adicionar informações extra simplesmente adicionando um novo campo ao seu formulário. O MongoDB, criado especificamente para cloud, fornece suporte completo a cluster e replicação automática, algo fundamental numa rede social, onde a concorrência pode chegar às centenas de milhar.

o JSON

A estrutura JSON (JavaScript Object Notation) é usada em todas as frentes da MEAN Stack (NodeJS, Angular e MongoDB).

o Alta velocidade e reutilização

O NodeJS é rápido e acessível devido à sua arquitetura “non-blocking”. O Angular 5 é uma estrutura JavaScript “Open Source” que oferece manutenção, testabilidade e reusabilidade.

o Código aberto e compatível com nuvem

Todas as tecnologias de MEAN Stack são Open Source. Ajuda no processo de desenvolvimento visto poder-se usar bibliotecas e repositórios públicos. O MongoDB ajuda a implementar funcionalidades na cloud ou servidor da aplicação, reduzindo o custo do espaço em disco.

o MongoDB

Devido ao “approach” tomado desde o início do desenvolvimento, “front-to-back”, seria displicente justificá-lo sem considerar a interface visual com as seguintes características principais:

▪ Todas as páginas principais são altamente relacionais, no sentido em que, na maioria das entidades, temos presentes informações de outras;

▪ A procura customizada de entidades, muitas vezes envolve características compostas (ex.: somatório de golos, número de golos por jogo (de jogadores e equipas)).

▪ Conteúdo dinâmico em que a sua escala é pouco previsível por se tratar de uma rede social.

▪ O desenvolvimento em Angular5, uma “Single-Page Application Framework”.

A MEAN Stack é de rápido desenvolvimento, fácil aprendizagem e combinação. Qualquer tecnologia disponível na Stack pode ser usada facilmente em integração com a outra, dependendo do requisito. É definitivamente uma nova alta tecnologia de ponta e inovação que possivelmente governará o mercado em breve.

**• Modelos não relacionais versus Modelos relacionais**

Face á emergência de base de dados NoSQL, criadas com redes sociais e outros casos de uso “Web 2.0”, optamos pela escolha de MongoDB, uma das DB’s NoSQL mais maturas no mercado.

9

o Como funciona MongoDB?

Como uma Base de dados orientada a “Documentos”, utiliza o conceito de dados e documentos autocontidos e auto descritivos, e isso implica que o documento em si tenha de definir como deverá ser apresentado e qual o significado dos dados armazenados na sua estrutura. Ou seja, idealmente, por página irá apenas ser preciso um documento.

Com este conceito em mente, começa-se por atacar as principais características da plataforma:

▪ Devido à informação relativa às restantes entidades ser, na sua grande maioria, dificilmente mutável (não se muda de nome todos os dias) e em pouca quantidade, a sua replicação não é, de todo, critica.

▪ As características compostas, como somatórios e agregações, estão autocontidas nos documentos, sendo a sua procura mais rápida, embora mais custosa em termos de atualização de base de dados.

▪ O ponto mais preocupante desta escolha, no ponto de vista teórico, é o tamanho máximo dos documentos. Isto porque um documento pode ter o tamanho máximo de 14MB. Sob o ponto de vista prático não aparenta ser um fator realmente alarmante (em perspetiva, por exemplo: o aclamado livro “A Guerra dos mundos “por H. G. Wells de 287 páginas, tem apenas 400KB). Ainda que seja difícil o tamanho dos documentos ser alarmantemente grande, se isso acontecer há métodos de divisão de documentos disponíveis.

▪ Quanto menos pedidos forem feitos á base de dados, mais responsiva será uma SPA, por existir apenas um ou dois “requests” de documentos à base de dados em detrimento de vários pedidos por informação relacional como em base de dados SQL, a aplicação apresenta “load times” muito mais curtos.

▪ Otimização de queries e escalabilidade da base de dados: O conteúdo relacional da plataforma é pouco “clusterizado”: Ex: os golos, assistências e cartões de um jogador, durante uma época, não ficam armazenados em disco relativamente perto uns dos outros, num contexto de base de dados SQL.

Em MongoDB, todo esse conteúdo fica num único documento, evitando Fetchs a várias secções do disco. Para além disso, tem um sistema de “sharding” que permite, com facilidade, uma escalabilidade horizontal da base de dados.

10

Web Crawler

**• Conceito geral**

Um web crawler, é um programa de computador que navega pela internet de uma forma metódica e automatizada. Vulgarmente também denominados por bots, web spiders, web robot ou web scutter.

O processo que um web crawler executa é chamado de rastreamento da rede ou indexação. Muitos sites, em particular os motores de busca, usam web crawler para manter uma base de dados atualizada. Os web crawler são principalmente utilizados para criar uma cópia de todas as páginas visitadas para um pós- processamento por um motor de pesquisa que irá indexar as páginas baixadas para prover pesquisas mais rápidas.

**• Processo**

Foi desenvolvido em NodeJS uma API que percorre o site www.zerozero.pt de forma a obter o máximo de dados de todos os jogadores, treinadores, clubes de futebol, resultados e classificações das primeiras divisões dos escalões sénior e últimos de formação do país.

A API utiliza a biblioteca “Crawler-js” para ler os valores das páginas HTML do site acima referido, os valores são em seguida mapeados para os documentos (modelos) existentes na base de dados em Mongo, através da biblioteca “Mongoose”.

Para além de permitir assim uma sólida quantidade de informação dos utilizadores principais na plataforma independente de insumos dos mesmos, juntamente com o dispatcher - que determina os períodos de “fetch” de informação - é possível obter dados cuja necessidade de que estejam o mais atuais possível sejam constantemente atualizados em curtos períodos de tempo (ex.: resultados de jogos e estatísticas).

**• Escolha de Biblioteca**

O processo de crawling está intimamente ligado á logica de negócio, há centenas de equipas e milhares de jogadores no mercado que se pretende atingir, com potencial de escalar vários níveis de grandeza no futuro, logo a biblioteca teria que ser robusta e versátil.

A biblioteca escolhida foi a “Crawler-js” que funciona a partir de uma fila de pedidos com vários workers, e que é altamente configurável, podendo-se personalizar “Proxies”, tempos máximos entre pedidos, entre outro tipo de configurações. Pode-se ainda usar a interface de JQuery para navegar pelo DOM que é uma Framework open Source e sem custos.

**• Hierarquia de Entidades e Paralelismo**

Devido à quantidade de páginas a ser “crawled”, optou-se por um processo que seja possível paralelizar. O processo escolhido seria ideal, embora implique abordar-se outros vários problemas:

o Atomaticidade das operações. Em MongoDB as operações são atómicas ao nível do documento, inclusive o “lock” do mesmo, ou seja, qualquer operação que seja atómica e autocontida (ausência de “SETs” - tudo à base de incrementações e outras operações sem conhecer o estado corrente).

o Dependências entre os modelos e Concorrência

Devido à Modularização dos modelos em si (nenhum destes precisa obrigatoriamente de outro para ter alguma identidade), a hierarquia é feita pelo quão contidas estão as entidades com a subida do nível hierárquico.

Competição:

11

Contém poucas dezenas de equipas. Contém centenas de jogadores. Contém centenas de jogos.

Equipa: Contem dezenas de jogadores. Contem centenas de jogos, mas de competições diferentes.

Jogo: Contem 2 equipas. Dezenas de jogadores. Pertence apenas a 1 competição.

Os jogos serão crawled com base nas competições (todos os jogos de uma competição, ao invés de todos os jogos de uma equipa) para garantir que quando os mesmos estão a ser processados não exista um constante “lookup” da existência dos jogadores ou equipas do mesmo, pois estes já vão estar devidamente processados. Para isto acontecer modificou-se a estrutura da pilha da Framework para uma “priority queue” em que nenhum pedido de uma prioridade inferior pode começar sem serem completos todos os pedidos de hierarquias anteriores.

Como se está a fazer crawling de modelos numa base de dados externa que funciona por REST, pode usar-se os IDs de todas as entidades como chaves da procura da plataforma, não sendo preciso reconhecer o ID na base de dados, isto torna o processo de atualização dos modelos muito mais simples.

**• Problemas encontrados e respetivas soluções**

Pré mundial

Na primeira abordagem ao site sob o qual foi feita a leitura, o crawl era feito de uma maneira transparente e sem barreiras, tendo como único “obstáculo” a necessidade ter um segundo de intervalo entre pedidos, pois se fossem feitos dez pedidos num segundo, o website apresenta um “captcha” insolvível (a versão do captcha estava desatualizada, e era impossível fazer bypass do mesmo de uma forma diferente do intervalo entre pedidos), mas, de uma maneira geral, era de um crawl acessível.

Pós mundial

Com a entrada do mundial o site teve atualizações notáveis, especialmente na proteção contra crawling:

o Captchas atualizados e de última geração – embora com uma pesquisa mais exaustiva, fosse possível ultrapassar este obstáculo fazendo os captchas (manualmente) de maneira local, guardar o cookie de resolução, e usá-lo em várias proxies.

o Conteudo importante protegido com necessidade de registo - para aceder a calendários ou resumos de jogos passou a ser necessário estar registado, e este registo obrigava a resolução de um captcha obrigatório, que não olhava a cookies, o que implica um registo manual. Ou seja, surgiu a necessidade de, no crawler, automatizar o login e gerir que contas iriam estar associadas aos diferentes proxies.

o Limite de visualização de conteúdo importante - Cada conta passou a estar limitada a um número fixo de conteúdos importantes por dia, isto significa que, em média, cada conta que criada passou a poder fazer apenas crawl de meia época antes de atingir esse limite. Logo, o processo de obter várias competições e ainda fazer o “re-run” do crawler, devido a inserção de atualizações do mesmo, passou a implicar criação de várias contas, sendo este um processo bastante demorado.

12

Gamification

**• Significado**

Gamification em desenvolvimento de software, também conhecido pelo termo “ludificação”, é o conceito de aplicar lógicas de modelo de negócio e técnicas de design encontrados em jogos para enriquecer contextos diversos não relacionados com jogos. Tipicamente aplica-se o conceito de Gamification a processos e aplicações com o objetivo de incentivar as pessoas a adotá-lo ou influenciar a maneira como são usados.

*“Dentre os objetivos da ludificação podemos destacar: tornar a tecnologia mais atraente, estimular os usuários a se engajarem com comportamentos desejados, mostrar um caminho para o domínio e autonomia, ajudar a resolver problemas sem ser uma distração, e tirar vantagem da predisposição psicológica humana de se engajar em jogos.*

*Esta técnica pode encorajar as pessoas a realizar tarefas que elas normalmente considerariam chatas, como completar questionários, fazer compras, completar formulários de impostos ou leitura de sites. Dados disponíveis de sites, aplicações e processos gamificados indicam potenciais melhoras em áreas como envolvimento dos usuários, retorno sobre investimento, qualidade de dados, prazos ou aprendizagem.” – Wikipedia[2]*

**• Gamification na Players Net**

A PlayersNet é uma plataforma onde o desporto é a sua razão de existência, e em todas as modalidades desportivas existentes há algo que as une integralmente – os seus participantes jogam. Portanto, o processo de conceção e desenvolvimento da plataforma foi sempre com a noção de Gamification em mente.

o Badges (crachás)

Figura 1 – Crachás da PlayersNet

A informação corrente e atualizada que a plataforma consegue ter, faz com que se torne possível uma atribuição de crachás consoante um determinado tipo de resultados obtidos. Por exemplo: um jogador que marque golo consecutivamente em 10 jogos, é despoletado um evento através do web crawler que lhe atribui um “badge” que represente essa conquista.

Este conceito é fortemente aplicado em videojogos, porque fornece uma forma de competição entre os jogadores. Aqui na PlayersNet o propósito é o mesmo. Ao entrar diariamente na plataforma, o jogador pode ter acesso aos crachás conseguidos pelos seus colegas e sentir-se aliciado em atingi-los igualmente – é uma forma de manter o utilizador entretido no uso da plataforma a longo prazo.

13

o Skills (talentos)

Figura 2 – Gráfico de Skills da PlayersNet

Desta vez, a informação é gerada pelos utilizadores. Com o conceito que permite a um grupo de utilizadores votar numa determinada “skill” de um jogador, é gerado um terminal de feedback fidedigno acerca dos talentos reais do votado - uma forma interessante de dar mais valor à plataforma devido a um novo setor de informação acerca dos utilizadores que só a PlayersNet passa a ter acesso.

Este tipo de abordagem também é algo que se encontra em jogos de consola, não o aspeto funcional que envolve o voto dos utilizadores, mas o aspeto de design, onde se observa um gráfico extremamente intuitivo e que caracteriza em poucos segundos de análise as verdadeiras qualidades do atleta. A título de exemplo, em jogos como o “FIFA” ou o “Pro Evolution Soccer”, encontra-se um chamado “Radar Graph” para rapidamente se perceber o valor e skills de um jogador – figura 4.

Figura 3 – Chart JS radar Figura 4 – Fifa radar

Este tipo de gráfico também se encontra disponível na biblioteca usada para a apresentação do gráfico de barras usado na plataforma, no entanto, as labels não se apresentavam devidamente visíveis para um utilizador comum (Figura 3).

14

o Pesquisa filtrada

A pesquisa filtrada de jogadores – uma das mais importantes funcionalidades da plataforma que revoluciona toda a experiência de Scouting de jogadores em Portugal, por acreditar-se que será uma funcionalidade muito utilizada, tinha de ter uma apresentação e usabilidade o mais “User friendly” possível. Após uma pesquisa algo trabalhosa, encontrou-se esta exata “feature” num dos jogos mais emblemáticos da história dos videojogos de futebol – Football Manager.

Figura 5 – Pesquisa de Jogadores (Football Manager)

Figura 6 – Pesquisa de Jogadores (PlayersNet)

Como se pode observar comparando ambas as figuras 5 e 6, a feature de pesquisa de jogadores da plataforma PlayersNet foi integralmente inspirada na que já existe no videojogo referido.

15

Arquitetura

**• Arquitetura Geral do Projeto**

figura 7- Arquitetura Geral da plataforma

Tal como a maioria dos projetos aplicacionais, a PlayersNet divide-te em 3 camadas – Base de dados em MongoDB, Servidor em NodeJS, Front-end em Angular 5. Para além dos dados inseridos por parte dos utilizadores, o servidor está preparado para obter também informação de outras fontes (outras bases de dados ou outros sites).

**• Funcionamento geral do front-end**

O Angular é uma Framework concebida para a criação de aplicações client-side em HTML e TypeScript. Implementa funcionalidades core e opcionais como sets de bibliotecas de TypeScript que podem ser importados para o projeto.

Os blocos básicos de uma aplicação em Angular são os “NgModules”, que providenciam um contexto de compilação para os componentes. Os NgModules aglomeram código relacional em conjuntos funcionais – uma aplicação Angular é definida pelo seu grupo de NgModules. Um projeto angular tem sempre, pelo menos, um módulo raiz (root module) que permite “bootstrapping”, mas tipicamente tem muitos mais módulos com outras funcionalidades.

Os componentes definem views, que, por sua vez, são grupos de elementos de écran que o Angular pode escolher dentre os existentes e modificar de acordo com o que for definido em termos de lógica e dados. E os componentes servem-se de serviços, que providenciam funcionalidades específicas que não estão diretamente relacionadas com as views. Os serviços – “Service providers” podem ser injetados nos componentes como dependências, tornando o Código modular, reutilizável e eficiente.

Tanto os componentes como os serviços são classes normais, com “Decorators” que marcam o seu tipo e fornecem metadata que diz ao Angular como devem ser usados. A metadata para uma classe de um componente é associada com um template que define a view. Um template que combina HTML normal com diretivas Angular ligadas ao Código de lógica através de “mark-ups” que permitem que o Angular modifique o HTML antes da rendarização final. Enquanto que a metadata para uma classe de serviço fornece ao Angular a informação que o mesmo necessita para torna-lo disponível a todos os componentes através de “Dependency injection”.

Tipicamente um componente define várias views organizadas de forma hierárquica. O Angular fornece um router – “Router service” de forma a ajudar na definição dos URL’s entre as views. Na figura 8 pode observar- se o funcionamento geral da aplicação Angular integrada neste projeto.

Figura 8 – Funcionamento geral de uma aplicação em Angular

**• Síntese**

Na camada de apresentação, apesar de se tratar de desenvolvimento front-end, como está a ser desenvolvida em Angular 5 existe um padrão de desenho bem explícito na sua estrutura. Sendo que esta camada se pode dividir em “controladores”, “modelos” e “vistas”, a estrutura Angular é muitas vezes confundida com um padrão de desenho MVC ou MVVM. No entanto, não há por defeito controladores ou ViewModels no Angular realmente, há sim componentes que derivam de templates, serviços e modelos.

Posto isto, na camada de front-end existe uma subcamada de modelos que representam os que se encontram na base de dados, outra de serviços onde se encontram os pedidos ao servidor que mapeiam os modelos e também outros serviços como os de lógica de autenticação e autorização, e, por fim, a subcamada visual que, com o Angular, nos permite dividir por cada componente diferente (i.e. player-component, team- component, header-component, footer-component, chat-component, skills-component) especificando um ficheiro .html, .css e .ts (este último funciona como a classe da componente) por cada um deles tornando possível ter características bastante específicas por componente.

Olhando para a estrutura completa desta camada, pode-se notar uma orientação a objetos subtil na forma como a mesma é implementada.

16

17

Descrição técnica

**• Abordagem pretendida**

Figura 9 – Arquitetura do projeto (Pretendida)

o Web - Componente Front-End da aplicação que, não só acarreta a user interface do projeto (client), mas também páginas que permitem a configuração específica de alguns eventos (admin).

Desta forma, alterações de configuração tais como: - O intervalo de tempo em que corre o crawler (comunicação com o dispatcher), - Critérios de prémios dos jogadores (comunicação com o regex), - Critérios de atribuição de badgers a jogadores ou treinadores (comunicação com o regex) passam a poder ser feitas por um utilizador administrador.

o Dispatcher - Componente responsável por lançar eventos automáticos que o crawler irá consumir. Para além disso, é no dispatcher que se encontram as ações que determinam as configurações definidas pela componente admin – referidas na componente anterior e é nesta componente que é são escritos na base de dados os valores obtidos pelo crawler. O dispatcher está também encarregue por lançar eventos detectados na informação obtida pelos crawlers (como por exemplo, um jogador marcou um hat-trick, ou é a décima vitória consecutiva do clube XPTO). A detecção da maioria desses eventos é definida pela componente regex.

o Regex - Nesta componente encontra-se toda a lógica que determina as condições necessárias para que um evento seja despoletado. De forma a reduzir a quantidade de condições necessária para que se capture os inúmeros eventos pretendidos (tais como 3 golos marcados num jogo, 5o jogo consecutivo sem marcar golos, 34o jogo consecutivo sem perder, etc.) optou-se por atribuir a cada entidade (clube, jogador ou treinador) um texto codificado que identifica diferentes eventos, por exemplo, “G” representa um golo, “W” representa uma vitória e “L” representa uma derrota. Assim, através de um sistema de regular expressions que vai percorrer os enormes textos dos utilizadores em questão, obtêm-se as respostas pretendidas e consegue-se detectar a ocorrência dos eventos definidos.

o Crawler - Componente responsável por recolher dados automaticamente de outros sites e retorná-los ao dispatcher. A única informação que o crawler recebe são dados de configuração e estes vêm por parte do dispatcher.

18

o DB Api - Componente que trata de toda a comunicação com a base de dados. Esta componente recebe os valores que o dispatcher recebe do crawler, os valores que o dispatcher recebe dos administradores, envia (quer sejam os valores automáticos, quer os que vêm da componente web).

o DB - Base de Dados MongoDB.

o Server - Componente servidora principal. Contém todos os endpoints da aplicação (Web Api) e é responsável por todo o processamento de Back-End da aplicação, tais como: autenticação, autorização, mensagens, envio automático de e-mails e comunicação com a componente de armazenamento de dados.

**• Abordagem final**

Figura 10 – Arquitetura do projeto (final)

Devido ao problema relatado no capítulo 4, o tempo reduzido fez com que a estrutura de back-end não fosse terminada, apesar de estar maioritariamente operacional.

O componente servidor passou a estar unicamente organizada com a estrutura apresentada na figura 10 onde o componente separado da DB API deixa de existir (a comunicação com a base de dados passa a ser da responsabilidade do componente Server) e os componentes Regex, Dispatcher e Crawler encontram-se no mesmo escopo onde os dados fluem bilateralmente.

O componente de front-end também deixa de ter o modo “Admin”, pelo que as alterações às lógicas de crawl estão, neste momento, fixas.

Não obstante, todo o código está construído de forma modular, ou seja, apesar de não estarem implementados da forma pretendida, a estrutura admite essa alteração sem grandes reformulações na lógica do projeto.

**• Sistema de Autorização e Autenticação**

Numa abordagem comum de autenticação web, após o login, o servidor cria um hash exclusivo para o cliente e insere-o na base de dados, onde, em seguida, verifica se o mesmo está presente no cookie, cabeçalho ou corpo de todos os pedidos HTTP - o que significa que seria necessário consultas á base de dados todas as vezes que o cliente visitava uma pagina protegida. Essa abordagem não é, de todo, eficiente, e foi decidido optar por um protocolo que usa JWT, que resolve este problema.

19

O JSON Web Token (JWT) é um open standard (RFC 7519) que define uma maneira compacta e autocontida de transmitir com segurança informações entre entidades como um objeto JSON. Essas informações podem ser verificadas e confiáveis porque são assinadas digitalmente. Os JWTs podem ser assinados usando um segredo (com o algoritmo HMAC) ou um par de chaves pública / privada usando RSA ou ECDSA.

Na sua forma compacta, o JWT consiste em três partes separadas por pontos (.), Que são:

o Cabeçalho - Normalmente consiste em duas partes: o tipo do token, que é JWT, e o algoritmo de

hashing usado, como HMAC SHA256 ou RSA.

o Payload - Esta segunda parte do token, contém as declarações sobre uma entidade (normalmente, o

usuário) e dados adicionais.

o Assinatura - É usada para verificar se a mensagem não foi alterada ao longo da sua transferência e, no caso de tokens assinados com uma chave privada, esta também pode verificar se o remetente do JWT é o correspondente.

**• Sistema de Pesquisa Filtrada de Jogadores**

Os sistemas de filtragem que contêm valores estáticos como valores pré-definidos (pé preferencial, posição, etc...) são mapeados por um ID. Como os campos inteiros são mais facilmente indexados a nível da base de dados, esta abordagem torna a procura mais rápida e eficiente.

**• Componentes da Interface do Utilizador**

o Models

Na pasta \_models encontram-se todos os modelos da plataforma. Os mesmos que são definidos na componente servidora.

Ao contrário de linguagens mais “verbosas”, o TypeScript permite-nos trabalhar sem definir a estrutura de um objeto “por antecipação”, “acordo” ou “contrato”, ou seja, jogar com o que recebemos nos serviços esperando que a estrutura esteja correta - isto pode ser algo proveitoso, no sentido em que se evita eventuais erros quando o responsável pelo servidor faz alterações aos seus serviços, mas numa plataforma onde ambas as componentes (servidora e cliente) são desenvolvidas em paralelo e mantidas pela mesma pessoa, há mais a ganhar quando os modelos são todos definidos.

Desta forma, aqui encontram-se os seguintes modelos:

▪ Os Modelos deste projeto, que num sistema de base de dados relacional seriam as entidades:

UserInfoViewModel; CompetitionViewModel; MatchViewModel; TeamViewModel; Recommendation; SkillSet; User.

▪ Os Modelos de Vista (view models)que, apesar de terem a mesma termologia que alguns dos modelos base, não se inserem na categoria anterior porque estes são puramente

20

modelos de view (ViewModels), ou seja, modelos que foram criados com o simples e único propósito de alimentar uma vista:

FilterSearch; Media; PlayerMatch; SearchEntityViewmodel; TryOut.

▪ Os Modelos Embutidos (embedded models) que são subclasses de auxílio a modelos que agregam muita informação.

TeamMatch; TeamPlayer; CompetitionSeason; Country; ExternalIds; FilterSearch; TeamAdditionalInfo; TeamSeason; UserInfoSearch; UserInfoSeason;

o Modals

Esta pasta inclui os “Modals” principais da plataforma. Os “Modals” são páginas que se sobrepõem a outras páginas de HTML principais com um conteúdo reduzido. Normalmente contém uma seleção de elementos, adição de informação ou ficheiros ou uma visualização especial de algum elemento. Funciona exatamente como um componente (Component) e possui uma estrutura igual.

▪ Recommendation Modal - Modal responsável pela criação ou edição de uma recomendação. (Os componentes Team e User Info Profile utilizam este Modal)

▪ Team Player Evaluation modal - Modal responsável pela criação ou edição de uma avaliação avançada de um jogador (Shadow Player). (Este Modal é apenas utilizado pelo componente Team)

▪ Tryout Modal - Modal responsável pela criação ou edição de um treino de captação. (Este Modal é apenas utilizado pelo componente Team)

o Services

O serviço é uma categoria abrangente que inclui qualquer valor, função ou recurso de que uma aplicação precisa. Um serviço é tipicamente uma classe com um propósito bem definido. Deve fazer algo específico e fazê-lo bem.

O Angular distingue os componentes dos serviços para aumentar a modularidade e a reutilização. Ao separar a funcionalidade relacionada à visualização de um componente de outros tipos de processamento, é possível criar classes de componentes simples e eficientes.

Idealmente, o trabalho de um componente é permitir a experiência do utilizador e nada mais. Um componente deve apresentar propriedades e métodos para vinculação de dados, a fim de mediar entre a visualização (renderizada pelo modelo) e a lógica do projeto (que geralmente inclui alguma noção de um modelo).

Um componente pode delegar determinadas tarefas a serviços como buscar dados do servidor, validar a entrada do usuário ou registrar-se na plataforma. Ao definir essas tarefas de

21

processamento numa classe de serviço injetável, essas tarefas ficam disponíveis para qualquer componente.

Portanto, nesta pasta encontram-se todos os serviços utilizados pela plataforma que comunicam diretamente com a componente servidora em NodeJS. Aqui poder-se-iam encontrar web services REST ou SOAP também, mas a plataforma delega quaisquer comunicações com o “exterior” à componente servidora. Apesar dos web-services serem internos, as chamadas são feitas em client-side, portanto é sempre feito um pedido http para aceder a um serviço na componente servidora. Toda a comunicação com os serviços do servidor é feita através de promessas (Promises).

▪ Serviços da aplicação:

Authentication Service - Nesta classe encontram-se todos as chamadas aos web services responsáveis pela lógica de autenticação em cada página da plataforma. Incluindo o Login e Logout.

Competition Service - Classe responsável por comunicar com os web-services que despoletam ações sobre uma determinada competição.

Generic User Service - Classe responsável por comunicar com os web-services que efetuam pesquisas simples ou avançadas sobre todos os utilizadores existentes.

Match Service - Classe responsável por comunicar com os web-services que despoletam ações sobre um determinado Jogo.

Team Service - Classe responsável por comunicar com os web-services que despoletam ações sobre uma determinada Equipa.

User Service - Esta classe é única e exclusivamente responsável por comunicar com os web- services alusivos ao utilizador corrente.

User Info Service - Classe responsável por comunicar com os web-services que despoletam ações sobre um determinado Jogador.

o Components

Um componente contém uma “view” em html (e respetiva classe de estilos .css) e o seu controlador.

É definida a lógica do componente — o que faz para suportar a view—numa classe. A classe interage com a “view” através de uma API com propriedades e métodos.

No caso desta plataforma, todos os componentes têm um objeto “viewModel” que é alimentado pelo serviço presente no construtor que é fornecido através de “Dependency injection”.

O Angular cria, atualiza e destrói componentes à medida que o utilizador se move na aplicação. São despoletadas ações durante o período ativo do componente através das funções que são chamadas em diferentes alturas do ciclo de vida do mesmo, como por exemplo o ngOnInit(), o ngAfterInit() ou o ngOnDestroy().

22

Todos os componentes são definidos no corpo da sua classe .ts através de uma anotação igual ou semelhante à seguinte:

Figura 11 – Definição de componente Angular 5

As configurações da figura 11 são as do componente App, aqui seguem exemplos das opções de configuração de componentes mais úteis e comuns:

selector: Um seletor CSS seletor que diz ao Angular para criar e inserir uma instância do componente em questão sempre que encontrar a “Tag” correspondente num template HTML.

templateUrl: O endereço “module-relative” do template HTML deste componente. Este template define a view do componente.

providers: Um array de “providers” para os serviços que o componente requer. Desta forma, este array diz ao Angular como providenciar instâncias de serviços ao componente cujo construtor usará para obter os dados que necessita e apresentá-los através da sua “view”.

**• Os Componentes Angular da aplicação**

o App

Este é o componente core da aplicação. Aqui se insere a “View” presente em todas as páginas do projeto, constituída pelo “Header” e “Footer”, sendo que o “Header”, devido ao seu nível de complexidade, tem um outro componente individual que o representa.

Figura 12 – Estrutura HTML das views do projeto

Na figura 12 está representada a classe app.component.html. A “Tag” <app-header> representa o componente “Header”, portanto, este será invocado na construção da view da componente “app” e a sua classe .html inserida, pela ordem apresentada, nesta classe. O mesmo se passa com a “Tag” seguinte - <router-outlet> que, neste caso, representa a “view” do componente invocado pelo “router”.

o Header

No componente de cabeçalho (Header), insere-se a lógica de Login (através de outro componente independente), de pesquisa de utilizadores, visualização e navegação das notificações do utilizador e visualização e edição dos dados de perfil do utilizador.

23

O Login aparece apenas quando não é verificado nenhuma “token” autorizada a estar na página requisitada, ou seja, quando ainda não foi feito nenhum login ou que a “token” tenha expirado.

o Home

Apesar de ser no componente Create Account onde se cria uma conta, é no componente Home que se captura o primeiro grupo de informação do novo utilizador.

o Login

Componente responsável por efetuar o Login na aplicação através da obtenção das credenciais do utilizador.

o Create Account

Este componente é responsável pela lógica e apresentação da “User interface” da criação de um novo perfil de utilizador.

Figura 13 – Construtor da classe do componente Create-Account.component.ts

Tal como o seu construtor, representado na figura 13 indica, indica, este componente consome os serviços Team, Competition e Generic User para criar a sua lógica aplicacional e alimentar a sua “view”.

o Competition

Este componente representa a entidade “Competição” da plataforma. A sua “view” é alimentada pelo seu controlador “Competition.component.ts” que, por sua vez, consome os serviços de da classe “Competition.Service.ts”.

Os restantes componentes da aplicação têm exatamente o mesmo tipo de estrutura, cada um com o propósito do modelo que representa. Sendo eles: Edit User Info, Filter User Info, Match, Team, Team Media, Team Player, Team Profile, Team Roster, Team Stats, Tryout Modal, User Info Profile, User Info, User Info Media e User Info Stats.

24

Interface da plataforma

**• Home Page**

Nesta página encontra-se a possibilidade de registo na plataforma ou início de sessão. Se for criada uma conta nova, o utilizador é encaminhado para a página de criação de conta associada a um jogador, onde o mesmo tem a hipótese de se encontrar na lista dos jogadores já obtidos pela PlayersNet.

Caso o utilizador já tenha conta e inicie sessão, é encaminhado para a sua página de perfil

Figura 14 – Página Home

**• Criação de jogador**

Após introdução dos dados para criação de uma nova conta, o utilizador é encaminhado para a página ilustrada na figura 15 onde, através de uma seleção que se adeque ao seu perfil, o utilizador pode encontrar o seu nome, escolhe-lo e entrar no seu perfil.

Escolha de género, escalão, competição e equipa de modo a encontrar o seu perfil na lista existente na plataforma – figura 15, 16 e 17.

Figura 15 – Página de criação de jogador

**• Escolha de jogador**

Figura 16 – Página de criação de jogador (Escolha do jogador)

25

Figura 17 – Página de criação de jogador (Iniciar)

**• Edição de jogador**

Ao escolher o seu jogador (ou seja, a si próprio), o utilizador é encaminhado automaticamente para a página de edição de jogador. Nesta página o jogador pode editar todos os seus dados pessoais, bem como aceder à lista de todos os jogos que efetuou, confirmar os dados dos jogos e alterar a sua imagem de perfil.

Figura 18 – Página de edição de jogador

26

27

Ao escolher a época é apresentada a lista de todos os jogos nos quais o jogador participou. Aqui aparecem todas as épocas que a PlayersNet conseguiu obter que constituam informação do jogador. É importante realçar que isso pode implicar aparecer jogos do jogador num clube diferente.

Figura 19 – Página de edição de jogador (Escolha de época)

Como se pode observar na figura 20, é apresentada uma listagem dos jogos do jogador com informação acerca dos golos, assistências, minutos jogados e cartões recebidos. A informação obtida pela PlayersNet não é 100% fidedigna, porque os erros acontecem sempre e em todo o lado, portanto, nada melhor do que os próprios utilizadores a validar toda a informação da plataforma.

Um aspecto transversal a todo o site é o feedback. Nas recomendações, nos votos de talento, nos crachás e, aqui, na confirmação dos dados obtidos pela PlayersNet. Por haver uma vulnerabilidade no que toca a dados válidos, apenas a informação confirmada por todos os intervenientes do evento (ou através de outro

algoritmo definido pela PlayersNet) se encontra assinalada como válida .

Figura 20 – Página de edição de jogador (Listagem de jogos)

Na figura 21 é mostrado um exemplo de uma edição de jogo. Como se pode observar, os dados foram alterados por parte do utilizador, portanto, o símbolo de validação encontra-se cinzento (por validar).

Figura 21 – Página de edição de jogador (Edição de jogos)

**• Página de Jogador - Perfil**

Após salvar a edição dos dados de perfil de jogador do utilizador, o mesmo é redirecionado para a sua página de perfil. Como o exemplo anterior não tem todos os dados possíveis (porque nenhum utilizador está registado na plataforma), o exemplo seguinte é um jogador fictício existente na plataforma para efeitos demonstrativos.

Figura 22 – Página de jogador (perfil)

28

Na figura 22 está representada a página de perfil de um jogador. Podemos observar informação importante como o número de seguidores e recomendações obtidas, bem como outros dados pessoais.

Para além da informação já referida, encontra-se na página de perfil um gráfico de barras horizontal que representa os “votos” de talento (skills) que outros utilizadores atribuíram no perfil do jogador apresentado. Os votos são adicionados clicando num dos talentos escolhidos pelo utilizador, representados pelos botões azuis acima do gráfico, cada um com um ícone representativo (pode ler-se o nome do talento passando o rato em cima do botão). O gráfico é imediatamente atualizado.

Na coluna central da página, abaixo do gráfico de talentos, encontram-se os dois últimos ou mais populares posts de “media” e a lista de recomendações.

Para que um utilizador possa criar uma nova recomendação, o mesmo terá de aceder ao modal de criação de recomendação através de um clique no botão que indica isso mesmo:

Figura 22 – Criação de recomendação

Aqui o “recomendador” tem a liberdade de escrever um texto recomendativo ao jogador do perfil. Para além do texto, o “recomendador” deve explicar qual a sua relação com o atleta em questão para que, novamente, o funcionamento de validação e feedback presente em toda a plataforma seja mantido. A figura 23 mostra a lista de recomendações na página de perfil de um jogador.

29

Figura 23 – Página de jogador (Lista de recomendações)

Na coluna direita da página estão os crachás do jogador.

Figura 24 – Crachás PlayersNet

É atribuído ao jogador um determinado crachá quando um acontecimento específico e previsto pela administração da plataforma é dado e capturado no processo de aquisição de dados (neste caso, aquando execução do crawler).

A figura 24 mostra três crachás (demonstrativos) obtidos pelo atleta em questão. Os crachás podem representar eventos tais como: “Goleador 10!” que iria representar o acontecimento: 10 jogos consecutivos a marcar pelo menos 1 golo”, ou, por exemplo, “Mestre das Assistências” que seria eventualmente algo como: 20 jogos numa época com pelo menos 1 assistência. Os crachás podem ser uma panóplia enorme de eventos e é uma forma interessante de inserir uma competição de conquistas na plataforma que, neste caso, é feita através de “Gamification”.

**• Página de jogador - Media**

30

A página de um jogador contem três diferentes separadores. O primeiro já foi abordado – o perfil de jogador, os restantes dois são o separador Media e o separador de Estatísticas.

O separador de media apresenta todas as publicações de “media” que o jogador insere no seu perfil. Os vídeos ou imagens podem ser ordenados pelo número de gostos, data de inserção ou por popularidade, que iria representar o número de partilhas, vistas ou menções do vídeo ou imagem.

Figura 25 – Página de jogador (Media)

**• Página de jogador - Estatísticas**

O separador de estatísticas do jogador apresenta os dados estatísticos simplificados do jogador para cada época em que tenha jogado. No exemplo abaixo (na figura 25) o valor de “Época” encontra-se vazio porque a PlayersNet apenas processou os dados da época 2017/2018, portanto, na verdade, os dados ali presentes são para efeitos demonstrativos de como será a User Interface numa versão final.

Na figura 26 pode observar-se um exemplo do separador estatístico da página do jogador Iker Casillas.

31

Figura 25 – Página de jogador (Estatísticas – Jogador fictício)

Figura 26 – Página de jogador (Estatísticas – Jogador real)

**• Página de Clube - Perfil**

Na figura 27 encontra-se a página de perfil de um Clube. Os aspetos novos em deterimento da página de perfil de jogador são o plantel existente na coluna direita da página, listagem dos treinos de captações e o separador Plantel que, numa outra versão da plataforma, apresentaria com algum detalhe o(s) plantel(eis) do clube.

32

Figura 27 – Página de clube (Perfil)

A PlayersNet foi construída para dar apoio aos atletas com menor visibilidade, desta forma, há também uma barreira de comunicação imediata entre os clubes e esses atletas. A funcionalidade apresentada na figura 28 e 29 – criação e listagem de treinos de captação, dá aos clubes a hipótese de criar uma publicação que irá ficar disponível no seu perfil para que os atletas se possam candidatar.

Umas das funcionalidades idealizadas inicialmente foi a possibilidade de escolher quais os utilizadores poderiam ver os treinos de captação presentes na página de perfil do clube. Uma adição de treino de captação seria um dos eventos a ser despoletados nas notificações dos utilizadores que seguem o clube.

Figura 28 – Criação de captação Figura 29 – Listagem de captações

33

34

Como se pode observar na figura 30, para além do plantel de atletas, encontra-se também na página de perfil do clube o seu plantel técnico.

Figura 30 – Pagina de clube (Corpo Técnico)

**• Página de Clube - Estatísticas**

No separador de dados estatísticos do clube é apresentada uma lista das competições correntes do clube (a última época assumida pela plataforma) e os últimos jogos do clube.

Figura 31 – Pagina de clube (Estatística)

Na figura 27 pode observar-se o botão que permite . Clicando nesse botão, o utilizador é direcionado para a página representada na figura 32.

**• Jogador Privado (Shadow Player)**

O jogador criado trata-se de um atleta que apenas o clube poderá ver – Shadow Player. Poderão haver jogadores que a PlayersNet não detectou ou, por ainda não estarem registados na plataforma, ter as suas páginas pouco atualizadas.

Nesses cenários os clubes têm aqui a hipótese de criar um atleta privado, adicionar informação relevante, incluindo vídeos e imagens, e ainda atribuir uma ou várias avaliações de prestação do jogador quando o vêm jogar pessoalmente, permitindo a manutenção de um historial do jogador.

Figura 32 – Pagina de clube (Shadow Player)

Nas figuras 32 e 33 pode observar-se a página de um jogador ainda por criar.

35

Figura 33 – Pagina de clube (Shadow Player)

Na figura 34 e 35 encontra-se um jogador fictício, para exemplos demonstrativos, já com alguma informação adicionada.

Como se pode observar-se na figura 35, com o menu de Avaliações aberto, é oferecido ao utilizador a hipótese de avaliar em quatro categorias diferentes com um valor de texto aberto.

Figura 34 – Pagina de clube (Shadow Player – Jogador fictício)

36

Figura 35 – Pagina de clube (Shadow Player – Jogador fictício)

**• Pesquisa personalizada de jogadores.**

Esta funcionalidade permite a qualquer utilizador com “Autorização” para aceder a esta página (dependendo do modelo de negócio da plataforma, poderão ser apenas os clubes, apenas os utilizadores com contas Premium ou outros diferentes cenários) fazer uma pesquisa detalhada e filtrada com base nos atributos do jogador.

Figura 36 – Pesquisa Filtrada de jogadores

37

A pesquisa é de filtragem dinâmica, ou seja, é dado ao utilizador, agrupados em seis categorias diferentes, uma lista de filtros possíveis onde o mesmo pode escolher que atributos utilizar na sua pesquisa.

---------------------------------------------------------------------------------------------------------------------------------------------------

Figura 37 – Pesquisa Filtrada de jogadores

Após escolha dos filtros desejados, o utilizador deve optar pelo tipo de filtragem (igual a, menor/maior que, entre) e o valor do filtro (Figura 39).

Figura 39 – Pesquisa Filtrada de jogadores

38

Ao pressionar “Pesquisar” é carregada a lista dos jogadores que preenchem os requisitos da pesquisa. É uma ferramenta preciosa para o “Scouting” de atletas – um dos problemas que este projeto visa resolver.

Figura 40– Pesquisa Filtrada de jogadores

**• Jogo**

Na página de jogo encontram-se todos os dados que o representam:

- Os plantéis (inicial, suplentes e corpo técnico). - O árbitro da partida, bem como o local onde se deu o jogo. - Local do jogo e informação de competição (data, jornada, nome e logo da competição).

39

Figura 41 – Pesquisa Filtrada de jogadores

Para além dos dados mencionados, é apresentado também os crachás conseguidos por atletas no desfechar do jogo.

Figura 42 – Pesquisa Filtrada de jogadores

40

41

**• Pesquisa Simples**

Na barra de navegação, o utilizador pode pesquisar acerca de qualquer tipo de conta (clube ou jogador) e é apresentada uma lista dos resultados obtidos. (Figura 43)

Figura 43 – Pesquisa Simples de jogadores, clubes e treinadores

42

Conclusões

**• Problemas e Obstáculos**

o Alteração da arquitetura

Como consta no capítulo que explica a arquitetura do projeto, houve uma significativa alteração do que tinha sido inicialmente definido que iria ser a estrutura da aplicação.

A primeira abordagem encontrava-se pouco dividida, o que acrescia o número de dependências entre componentes, tornava a aplicação pouco suscetível a alterações futuras que, num projeto desta categoria, tende a acontecer à medida que se vai desenvolvendo e havia componentes (como o crawler) que eram responsáveis por vários tipos de processamento.

Na abordagem corrente já se pode encontrar uma arquitetura mais segmentada, bastante sensível a alterações e melhorias sem que interfira com a lógica funcional do projeto, com componentes com propósitos bem definidos e de melhor compreensão aos olhos de quem tenta interpretar a lógica do projeto.

No entanto, esta alteração, bem como a definição dos modelos da base de dados não relacional (MongoDB funciona apenas por referências entre entidades), foram causa de uma demora maior nessa fase de desenvolvimentos, o que trouxe consequências nos tempos estimados para outros desenvolvimentos.

o Dependência dos Sites para os Crawlers

Este foi sem dúvida o obstáculo com maior impacto no desenvolvimento deste projeto. Numa das versões iniciais deste relatório (o relatório de progresso) foi apresentado o seguinte texto acerca deste tema:

*“Tal como se verifica nas explicações do funcionamento do web cralwer desta aplicação, há uma enorme dependência das fontes do mesmo para que o funcionamento saia ileso e com sucesso. Ou seja, basta que uma das páginas lidas pelo crawler seja alterada para que o mesmo deixe de funcionar corretamente.*

*Para solucionar este problema, foi necessário fazer uma análise da probabilidade da alteração das páginas em questão junto com os administradores do site das mesmas da qual se concluiu que, apesar de muito remota, existe obviamente essa possibilidade. Portanto, terá de ser criado no dispatcher um sistema de Logging que notifique essas alterações para que rapidamente se possa atualizar o crawler em função das mesmas. Não sendo esta uma definitiva, o objetivo é que, futuramente, se consiga obter todos os dados diretamente da mesma fonte onde o site de onde, neste momento, se está a extrair informação a consegue obter.” – Web Crawlers [4]*

Como se pode observar no capítulo 4, a estrutura de servidora de back-end sofreu alterações consideráveis, nomeadamente a exclusão da separação da maioria dos componentes e a elaboração de um componente servidor único e interligado.

O tempo dedicado em resolver a situação surgida pela alteração do modelo de back-end no seu componente de crawling (explicado no capítulo 4) foi consideravelmente superior ao expectado. Sem este componente a plataforma não tinha vida, então a sua prioridade foi sempre a maior possível, no entanto, isso fez com que grande parte dos restantes componentes saíssem prejudicados.

o Tempo para o desenvolvimento Front-End

43

Seguindo a explicação no primeiro problema apontado neste capítulo, o tempo tornou-se escasso para alguns desenvolvimentos que acabaram por não acontecer.

Visto que todo o projeto carece de um workflow altamente elaborado, há desenvolvimentos de back-end que não poderiam, de forma alguma, sair prejudicados, pondo em causa o funcionamento geral da aplicação. Logo, se realmente houver necessidade de abdicar de alguns desenvolvimentos, será a camada de apresentação a sacrificada, e assim foi, especialmente na organização de classes CSS e reutilização de componentes Angular em páginas com uma estrutura idêntica.

**• Lições**

O desenvolvimento core de uma aplicação web com este nível de complexidade exige um nível igual de empenho, dedicação e recursos. É um projeto com potencial para se tornar numa verdadeira plataforma de rede social desportiva, mas o tempo para um projeto sólido e estruturado teria de ser partilhado por mais programadores.

Portanto, a primeira lição aprendida foi a estimativa de desenvolvimento, apesar de grande parte desse fator ser justificado por um dos contratempos encontrados durante o desenvolvimento do projeto (referido no subcapítulo “Problemas”) – a mudança inesperada do funcionamento do site onde a plataforma percorre o crawler. Nos relatórios Beta e de progresso foram apresentados cronogramas que se mostraram surreais à medida que o tempo passava.

Em adição à estimativa de tempos pouco precisa, este projeto foi útil para aplicar muitos conceitos aprendidos na licenciatura no qual o mesmo se insere. A linguagem utilizada não foi ao acaso, para além dos fatores principais mencionados no capítulo 3, NodeJS foi a linguagem aplicada na cadeira de Programação na Internet, onde, para além da aprendizagem do funcionamento geral, conceitos básicos e as linguagens Javascript e Typescript, explorou-se com o sólido nível de detalhe a programação assíncrona, paralelismo e processos automatizados – algo fundamental para a componente de crawling deste projeto. O mesmo se passa com a cadeira de Programação Concorrente, cujos conceitos estão nos princípios base da linguagem NodeJS e a sua aplicação em crawlers – explicado com maior detalhe no capítulo 4.

Desenvolver, na sua íntegra, um projeto de aplicação web, é certamente uma tarefa complexa, a elaboração deste projeto forneceu bases e conhecimentos muito específicos do que requer um trabalho a este nível.

**• O que é que poderia/deveria ter sido diferente.**

Indo ao encontro do que foi explicado nos dois subcapítulos anteriores, com uma melhor estimativa de tarefas e um objetivo mais realista, o resultado final da plataforma podia ter sido substancialmente melhor.

O projeto não se encontra como desejado, infelizmente, mas esse assunto é tratado no tópico abaixo. Utilizando uma abordagem mais segura, ou seja, focada na qualidade do desenvolvimento de software da plataforma e abordando apenas um ou dois “problemas” a serem atacados, analisados, resolvidos e demonstrados a nível da engenharia informática, faria com que a PlayersNet fosse uma solução com melhor estabilidade, apesar de apresentar um potencial significativamente inferior.

A plataforma no seu conceito geral aparentava tentar apenas resolver um “problema” – uma rede social desportiva. No entanto, é apenas um problema a nível de modelo de negócio, o que não é de todo o propósito da cadeira de Projeto e Seminário. Porque, a nível da engenharia informática, a PlayersNet tentava resolver inúmeros problemas com um

44

nível considerável de exigência. Tais como:

Desenvolvimento Back-end

- Web crawlers; - Proxys; - Regex; - Paralelismo; - Web services; - Segurança informática; - Sistemas servidores; - Base de Dados não relacional (NoSQL);

Desenvolvimento Front-end

- Desenvolvimento Web geral e em SPAs; - Angular 5; - Web design; - Classes de CSS; - Ficheiros HTML com um alto nível de complexidade;

Nas fases finais deste desenvolvimento houve a necessidade de se optar por fechar alguns componentes com uma qualidade de código superior e mais organizada ou apostar na demonstração do potencial do projeto, e na sua grande maioria, optou-se pela segunda abordagem. Esta decisão tem alguma razão de ser.

Enquanto houve uma necessidade de alocar alguns recursos de tempo em reuniões com algumas entidades que pouco ou nada se relacionam com desenvolvimento informático (federação portuguesa de futebol, associações distritais, treinadores, jogadores, diretores e representantes de clubes, etc.) surgiram vários interessados em financiar o lançamento oficial da plataforma.

Sendo que a ideia deste projeto partiu inteiramente por parte de quem o desenvolveu, a possibilidade de lançar a PlayersNet a um nível real fez com que o desenvolvimento passasse a ter uma vertente quase comercial. Ou seja, ao invés de seguir a linha inicialmente definida e, apercebendo-se que o tempo se tornou reduzido, optar-se por fechar componentes com qualidade, optou-se por continuar a implementar novos componentes mesmo que os anteriores não estivessem devidamente implementados. Esta decisão não foi a mais correta – o propósito da cadeira de Projeto e Seminário é puramente académico, e seguir uma abordagem diferente desta prejudicou o output final.

**• Em que medida foram atingidos os objetivos**

Os objetivos fundamentais primeiramente definidos aquando da definição do projeto não foram atingidos. O objetivo principal era desenvolver um projeto que fosse 100% funcional e com as funcionalidades explicadas na proposta de projeto, e isso não se verifica.

Não obstante, outros desafios foram surgindo no desenrolar do desenvolvimento do projeto e, felizmente, grande parte deles foi ultrapassada com sucesso.

Num espaço de tempo relativamente curto, tendo em conta a solução final, conseguiu-se apresentar um projeto com respostas técnicas muito interessantes já enunciadas e exploradas nos primeiros capítulos deste relatório. Num trabalho onde o foco principal é a Engenharia Informática, e tratando-se de um projeto académico, onde todos os trabalhos têm um único e simples propósito – aprender, considero que tenha sido bastante positiva a experiência global.

Com isto, como já foi referido, não foram atingidos os objetivos inicialmente propostos. No entanto, foram sem dúvida atingidos os novos objetivos emergidos com o nascimento de outros níveis de complexidade informática não antecipados na proposta deste projeto.

45

**• Trabalho futuro**

Como se pode perceber pela forma como a plataforma foi desenvolvida, o projeto passou a ser um protótipo que se foca cegamente num futuro próspero.

Os problemas que este projeto visa solucionar, é espectado que tenha uma aplicação breve e importante no “mundo real” – ajudando os clubes e os jogadores (agora de futebol, mas poderá ser escalado para todos os outros desportos) a comunicar melhor entre si e publicitarem-se de uma forma mais interativa, fidedigna e real.

Como foi referido no subcapítulo que aborda o que poderia ter sido diferente, grande parte do desenvolvimento deste projeto foi focada na prosperidade da PlayersNet. Para que se torne num projeto fechado é algo que exige muito tempo e dedicação, portanto não poderá ser desenvolvido com uma alocação de recursos parcial. Com a possibilidade de haver um investimento financeiro nesta plataforma, os objetivos tornam-se tão altos quanto o investimento for, mas é certamente algo que não irá cessar com o término deste projeto a nível académico, com ou sem investimento.

**• Notas**

Em auxílio à descrição deste projeto, no dossier de projeto, insere-se uma pasta com a descrição técnica detalhada de todo o projeto, com uma estrutura e visualização web bastante user friendly. Para quaisquer dúvidas acerca da descrição técnica, serve essa pasta para uma melhor explicação.

46

Referências

[1] Web Crawlers - https://pt.wikipedia.org/wiki/Rastreador\_web; [2] Gamification - https://pt.wikipedia.org/wiki/Ludificação; [3] Angular Advantages - https://www.zeolearn.com/magazine/7-advantages-of-developing-apps-with-mean- stack; [4] Web Crawlers - https://medium.com/of-all-things-tech-progress/introduction-to-webcrawling-with- javascript-and-node-js-f5a3798ee8ac;