Notas sugeridas:

*Não tente descrever a função de cada componente, mas descreva detalhadamente a função de cada componente invulgar ou crítico*

*As figuras e as tabelas devem ser legíveis, instrutivas,*

*legendadas e ter título.*

*Todas as ilustrações devem ser descritas e referidas no texto*

*Os exemplos devem ser suficientemente detalhados para*

*ilustrar o conceito.*

**Relatório Final   
  
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.

**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.

**Índice**

**Capítulo 1 – 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.

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 das 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.

**Capítulo 2 – Solução Funcional**

Terminologia

- Skill: Talento ou atributo mensurável de um determinado utilizador.  
 - Badge: Título em destaque atribuído a um utilizador quando preenche um determinado número de requisitos.  
 - Scouting: Procura ativa da prospeção de jogadores.Em síntese, a plataforma oferece a seguinte solução:  
  
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.  
- 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.

**Capítulo 3 – Porquê MEAN Stack?  
 - NodeJS versus outras linguagens**

**- Geral  
 - Event Driver  
 - Web Crawling  
 - Javascript/ Typed Script  
 - NoSQL (MongoDB) versus Modelos relacionais  
 - Angular 5, Benefícios**

**Capítulo 4 – Web Crawler   
- Conceito geral   
- Tecnologias  
- Pros e Contras  
- Problemas encontrados e respetivas soluções  
(Especificação técnica ao detalhe apenas na arquitetura e descrição de componentes)**

**Capítulo 5 – Gamification   
- Significado  
- Aplicação na Players Net  
 - Badges  
 - Skills  
 - Pesquisa filtrada (FM)   
- Benefícios**

**Capítulo 6 – Arquitetura   
- Geral  
- Definição e explicação dos componentes (sem grande detalhe técnico)**Back End Node Architecture overview

Front End Angular Architecture overview

Angular is a platform and framework for building client applications in HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.

The basic building blocks of an Angular application are NgModules, which provide a compilation context for components. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules. An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.

Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.

Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.

The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.

The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI).

An app's components typically define many views, arranged hierarchically. Angular provides the Router service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities.

Modules

Angular NgModules differ from and complement JavaScript (ES2015) modules. An NgModule declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities. An NgModule can associate its components with related code, such as services, to form functional units.

Every Angular app has a root module, conventionally named AppModule, which provides the bootstrap mechanism that launches the application. An app typically contains many functional modules.

Like JavaScript modules, NgModules can import functionality from other NgModules, and allow their own functionality to be exported and used by other NgModules. For example, to use the router service in your app, you import the Router NgModule.

Organizing your code into distinct functional modules helps in managing development of complex applications, and in designing for reusability. In addition, this technique lets you take advantage of lazy-loading—that is, loading modules on demand—to minimize the amount of code that needs to be loaded at startup.

For a more detailed discussion, see Introduction to modules.

Components

Every Angular application has at least one component, the root component that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.

The @Component() decorator identifies the class immediately below it as a component, and provides the template and related component-specific metadata.

Decorators are functions that modify JavaScript classes. Angular defines a number of decorators that attach specific kinds of metadata to classes, so that the system knows what those classes mean and how they should work.

Learn more about decorators on the web.

Templates, directives, and data binding

A template combines HTML with Angular markup that can modify HTML elements before they are displayed. Template directives provide program logic, and binding markup connects your application data and the DOM. There are two types of data binding:

Event binding lets your app respond to user input in the target environment by updating your application data.

Property binding lets you interpolate values that are computed from your application data into the HTML.

Before a view is displayed, Angular evaluates the directives and resolves the binding syntax in the template to modify the HTML elements and the DOM, according to your program data and logic. Angular supports two-way data binding, meaning that changes in the DOM, such as user choices, are also reflected in your program data.

Your templates can use pipes to improve the user experience by transforming values for display. For example, use pipes to display dates and currency values that are appropriate for a user's locale. Angular provides predefined pipes for common transformations, and you can also define your own pipes.

For a more detailed discussion of these concepts, see Introduction to components.

Services and dependency injection

For data or logic that isn't associated with a specific view, and that you want to share across components, you create a service class. A service class definition is immediately preceded by the @Injectable() decorator. The decorator provides the metadata that allows your service to be injected into client components as a dependency.

Dependency injection (DI) lets you keep your component classes lean and efficient. They don't fetch data from the server, validate user input, or log directly to the console; they delegate such tasks to services.

For a more detailed discussion, see Introduction to services and DI.

Routing

The Angular Router NgModule provides a service that lets you define a navigation path among the different application states and view hierarchies in your app. It is modeled on the familiar browser navigation conventions:

Enter a URL in the address bar and the browser navigates to a corresponding page.

Click links on the page and the browser navigates to a new page.

Click the browser's back and forward buttons and the browser navigates backward and forward through the history of pages you've seen.

The router maps URL-like paths to views instead of pages. When a user performs an action, such as clicking a link, that would load a new page in the browser, the router intercepts the browser's behavior, and shows or hides view hierarchies.

If the router determines that the current application state requires particular functionality, and the module that defines it hasn't been loaded, the router can lazy-load the module on demand.

The router interprets a link URL according to your app's view navigation rules and data state. You can navigate to new views when the user clicks a button or selects from a drop box, or in response to some other stimulus from any source. The router logs activity in the browser's history, so the back and forward buttons work as well.

To define navigation rules, you associate navigation paths with your components. A path uses a URL-like syntax that integrates your program data, in much the same way that template syntax integrates your views with your program data. You can then apply program logic to choose which views to show or to hide, in response to user input and your own access rules.

****

**Capítulo 7 – Descrição técnica  
- Metodologia de trabalho  
- Detalhe técnico explícito de todas as componentes de Back-End   
- Detalhe técnico explícito de todas as componentes de Front-End**Models

Guards  
 - Administrator (Explicar Processo JWT através de um Post para o servidor)  
 - Authorization (Explicar Processo de validação da Token – envio da token corrente em todos os pedidos que assim o exigem)  
  
Helpers  
  
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*.

Services  
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.  
  
 - 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.

Validators  
  
Components

A component controls a patch of screen called a view. For example, individual components define and control each of the following views from the Tutorial:  
  
You define a component's application logic—what it does to support the view—inside a class. The class interacts with the view through an API of properties and methods.

For example, HeroListComponent has a heroes property that holds an array of heroes. Its selectHero() method sets a selectedHero property when the user clicks to choose a hero from that list. The component acquires the heroes from a service, which is a TypeScript parameter property on the constructor. The service is provided to the component through the dependency injection system.  
  
Angular creates, updates, and destroys components as the user moves through the application. Your app can take action at each moment in this lifecycle through optional lifecycle hooks, like ngOnInit().

- App

- Competition

- Create Account

- Edit User Info

- Filter User Info

- Header

- Home

- Login

- Match

- Team

- Team Media

- Team Player

- Team Profile

- Team Roster

- Team Stats

- Tryout Modal

- User Info Profile

- User Info

- User Info Media

- User Info Stats  
 **Capítulo 9 – Descrição funcional  
- Manual de utilizador com exemplos ilustrativos da plataforma.**

**Capítulo 10 – Conclusões  
- Lições aprendidas com o desenvolvimento da plataforma.  
- O que é que poderia/deveria ter sido diferente.  
- Em que medida foram atingidos os objetivos.  
- Trabalho futuro.**

**Referências**

**Apêndices  
- Listagem de programas.  
- Definição tecnológica da plataforma.**