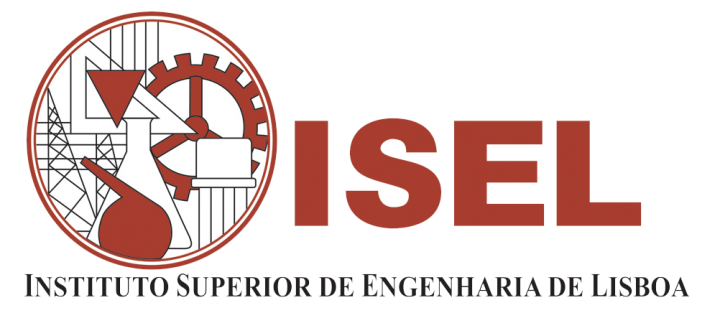
Departamento de Engenharia Electrónica e Comunicações e de Computadores

Licenciatura em Engenharia Informática e de Computadores



Relatório do Trabalho Final

Programação na Internet

Docente: Eng.º Luís Falcão

**Grupo 8:**

36893 – Gonçalo Domingues

36920 – Diogo Afonso

40916 – Nuno Pereira

# *Indice*

[*Indice* 1](#_Toc445068686)

[*Introdução* 1](#_Toc445068687)

[*Terminologia* 3](#_Toc445068688)

[*Instalação* 4](#_Toc445068689)

[*Packages* 5](#_Toc445068690)

[*controllers* 5](#_Toc445068691)

[*middlewares* 9](#_Toc445068692)

[*models* 11](#_Toc445068693)

[*node\_modules* 12](#_Toc445068694)

[*public* 13](#_Toc445068695)

[*views* 14](#_Toc445068696)

[*root & bin* 15](#_Toc445068697)

[*Autenticação* 17](#_Toc445068698)

[*Conclusões* 18](#_Toc445068699)

[*Conclusões* 18](#_Toc445068700)

# *Introdução*

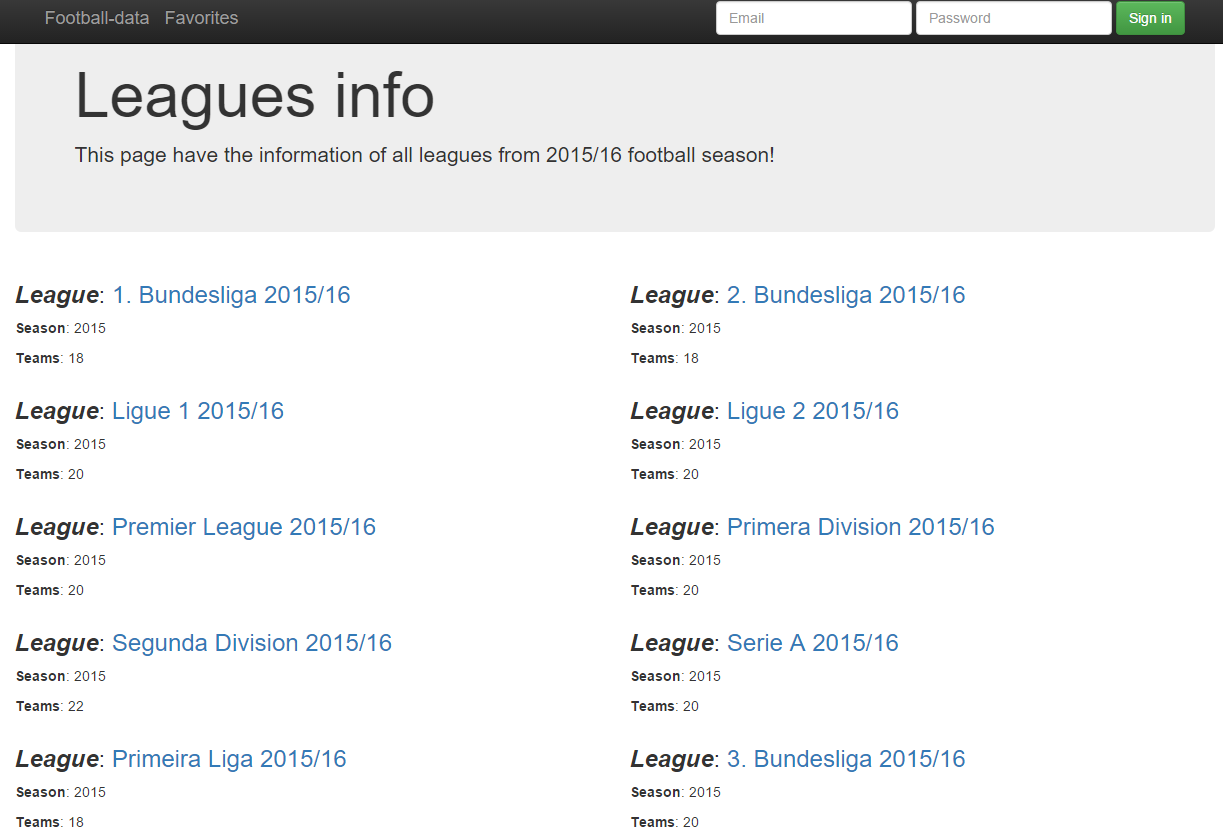
Neste documento pretendemos demonstrar o objectivo da aplicação *Web* “football-data”, assim como as soluções apresentadas pelo grupo no trabalho final da unidade curricular **Programação na Internet**.

No presente trabalho o grupo assimilou as diversas temáticas leccionadas durante o semestre na unidade curricular, sendo elas, o protocolo *http*, as linguagens *javascript*, *css* e *html* assim como as *frameworks* *Express,* *Handlebars* e *Twitter* *Bootstrap*. (Houve um cuidado em utilizar as versões recentes do *HTML* 5, *CSS* 3 e *javascript* (*ECMA* *script* 6)).

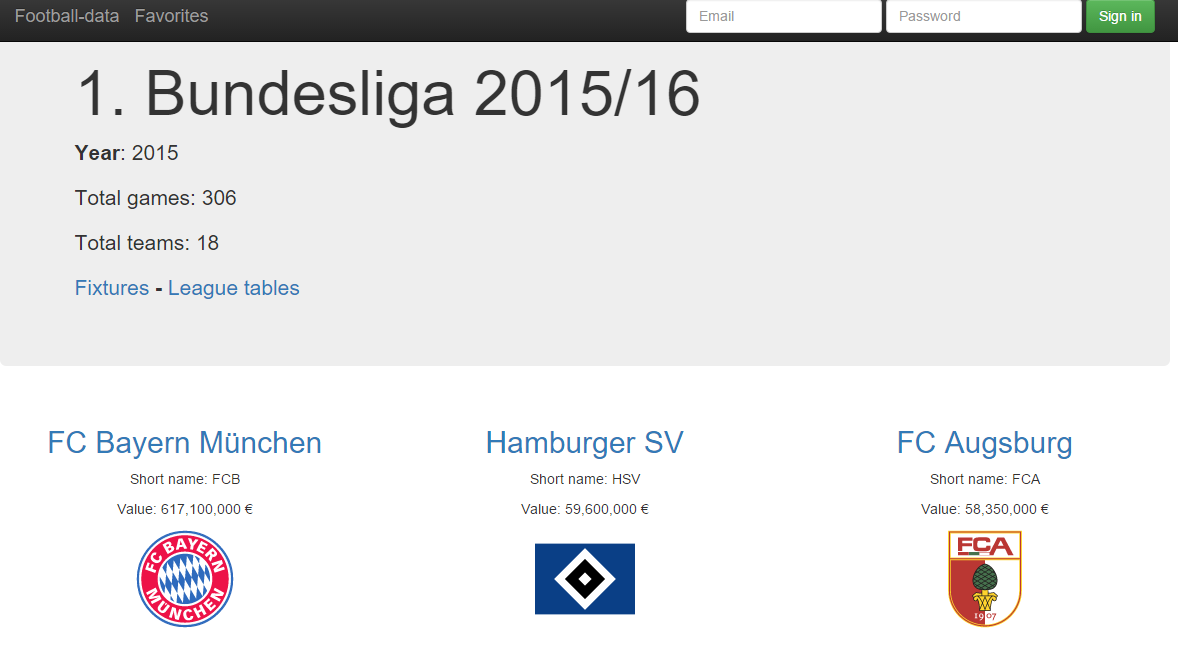
A implementação desta aplicação *Web* foi baseada na *framework Express*, que por sua vez esta é “construída” em *node.js*, tecnologia baseada em *javascript* para a criação de aplicações servidoras. A *framework Twitter Boostrap* foi também utilizada para definir o *layout* das páginas *html* apresentadas. Foi também utilizada a técnica *ajax* para a realização de algumas operações e *dom* para tornar a construção das páginas *html* mais dinâmicas.

Como foi dito anteriormente, o objectivo desta aplicação *Web* “*football-data*”, permite ao utilizador visualizar as ligas disponíveis (para a época 2015/2016), com hiperligações para as equipas das respectivas ligas, que por sua vez para as devidas equipas. É dada também a possibilidade ao utilizador de se autenticar na aplicação Web e ter direito a criar grupos de equipas favoritas, assim como ver os próximos N jogos ou últimos N jogos.

Um utilizador comum, ou autenticado, é presenteado pela seguinte página:



Que por sua vez ao clicar numa liga:



Podendo visualizar os jogos da liga, ou a tabela classificativa. Bem como os jogos de uma equipa em especifico.

Ao utilizador autenticado é possível consultar os seus grupos de equipas favoritas, bem como adicionar novas equipas.

Graças às ferramentas fornecidas pela *framework Express*, a aplicação faz uso de *middlewares* que permitem as rotas de pedido-resposta, comuns no protocolo *http*. Estas rotas contêm três obejctos, o *request* que contem toda a informação do pedido *http* feito ao servidor; o *response* que contem toda a informação da resposta *http* por parte do servidor; e o *next* que permite a chamada do próximo *middleware* na cadeia de *middlewares*. Para armazenar de forma persistente as equipas favoritas, foi utilizada uma base de dados não relacional, *couchdb*.

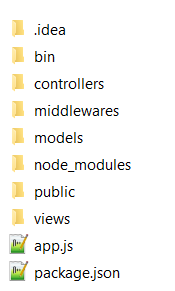
Todos estes temas abordados na introdução vão ser desenvolvidos nos próximos capítulos.

# *Terminologia*

|  |  |
| --- | --- |
| *HTTP* | Hyper Text Transport Protocol. |
| *HTML* | Hyper Text Markup Language. |
| *CSS* | Cascade Style Sheet. |
| *JSON* | Javascript Object Notation. |
| *DOM* | Document Object Model |
| *Javascript* | Linguagem de script e interpretada. |
| *AJAX* | Asynchronous javascript. |
| *Node.js* | Interpretador de javascript em componente servidora. |
| *Express* | Framework baseada em node.js que fornece muitas recursos a aplicações Web. |
| *Handlebars* | Framework que funciona em conjunto com Express, que gere as páginas html. |
| *Twitter Bootstrap* | Framework que permite a customização das páginas html. |
| *Middleware* | Função que gere o pedido-resposta ao servidor. |
| *Request* | Objecto que representa o pedido http feito ao servidor. |
| *Response* | Objecto que representa a resposta http feita pelo servidor. |
| *Next* | Objecto que gere a cadeia de chamadas dos middlewares. |
| *Get* | Metodo http que permite obter recursos específicos. |
| *Post* | Metodo http que permite submeter dados ao servidor. |
| *Put* | Metodo http que permite submeter/actualizar dados |
| *Couchdb* | Repositório de dados que armazena dados em formato json, sendo o acesso feito pelo protocolo http. |
| *MVC* | Model View Controller. |
| *NPM* | Node Package Module. |

# *Instalação*

Para instalar a aplicação *Web* “*football-data*”, o utilizador deve abrir uma linha de comandos na raiz do projecto, que contém o ficheiro *package.json* (que contém informação básica da aplicação, como o nome, versão, ponto de entrada e dependências), depois de aberta a linha de comandos o utilizador deve colocar o comando “npm install”.



O comando acima mencionado irá instalar os seguintes módulos (necessários para o funcionamento da aplicação):

"dependencies": {

"body-parser": "^1.15.0",

"cookie-parser": "^1.4.1",

"debug": "^2.2.0",

"express": "^4.13.4",

"express-handlebars": "^3.0.0",

"express-session": "^1.13.0",

"hbs": "^4.0.0",

"http": "0.0.0",

"node-couchdb": "^0.4.0",

"passport": "^0.3.2",

"passport-local": "^1.0.0",

"path": "^0.12.7",

"request": "^2.69.0"

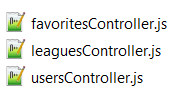
}

Por fim, para correr a aplicação *Web* o utilizador deve mudar para a directoria “*<raiz>/bin/”* que contem o ficheiro *www*, deve ser aberta uma linha de comandos nesta directoria e correr o comando *node www* e no browser colocar o *url* ***localhost:3000***.

(Nota: é necesário o *node.js* instalado, assim como o *couchdb*, para este último é necessário correr aplicação “*Start CouchDB*”).

# *Packages*

## *controllers*



A aplicação *Web*, “*football-data*” faz uso do padrão ***MVC***, ou seja a interacção com o utilizador, acesso a dados e regras de negócio são geridas a partir destes *controllers*, comunicando com os modelos e apresentando os resultados nas *views*. Sendo estas “*transacções*” “*transparentes*” para o utilizador final.



O *favoritesController*, é o controlador que permite gerir todos os *end-points* que sejam relativos à gestão dos grupos de favoritos. Sendo esta gestão, uma criação de um grupo; associar uma equipa a um grupo; ou até mesmo remover essa associação; podendo remover um grupo (por consequência, remove as equipas associadas); podendo consultar os próximos/últimos N jogos de uma equipa.

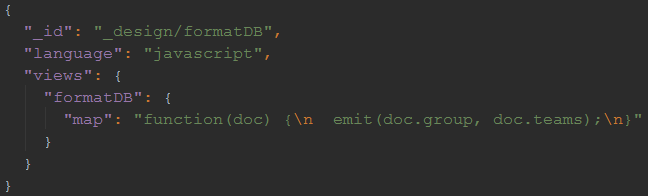
|  |  |  |
| --- | --- | --- |
| ***End-point*** | ***Método http*** | ***Ajax*** |
| */pingdb* | ***GET*** | Sim |
| */insertGroup* | ***POST*** | Não |
| */insertTeam* | ***POST*** | Não |
| */deleteGroup* | ***POST*** | Não |
| */all* | ***GET*** | Não |
| */changeT/:idL* | ***POST*** | Sim |
| */addT/:groupN/:idL/:idT* | ***POST*** | Sim |
| */:idGroup/teams* | ***GET*** | Não |
| */nextFixtures* | ***GET*** | Não |
| */fixtures/:idL/:idT/:name/:groupName* | ***GET*** | Não |
| */deleteTeam* | ***POST*** | Não |

Todos estes *end-points* são prefixados por *“/favorites”*, sendo que são responsáveis pela manutenção dos grupos de favoritos.

O *end-point “/favorites/pingdb”* é responsável pela verificação do estado da base de dados. Se esta está “acessível” ou criada, pois se não tiver os mecanismos necessários à sua criação e utilização são despoletados. Quer isto dizer, para este *end-point* ser executado existe uma cadeia de *middlewares* que prepara, a resposta do servidor.



Os *middlewares* “*checkDatabase*”, “*createDatabase*” e “*createView*” são chamados em “cadeia”, o primeiro *middleware* é responsável por fazer um pedido *http* *GET* à base de dados e com base no *status* *code* (*200* – **OK**, *404* – **NOT FOUND**), dado pela resposta é criada ou não a base de dados (assim como sua *view*). Para criar a base de dados, é feito um pedido *http* *PUT* para o *uri* “*http://localhost:5984/footballdata*”, em que o nome “*footballdata”* vai ser o nome do documento que vai armazenar a informação sobre os grupos de equipas favoritas. Quando a base de dados é criada a sua “*view*” também é gerada.



Para tal é enviado um documento à base de dados com as seguintes propriedas, *“\_id*”, “*language*” e “*views*”. Sendo que esta ultima propriedade contém um objecto como valor, esse objecto é o nome que a *view* vai ter, e tem uma propriedade *map*, cujo valor é uma *String* que contém função que vai mapear todos os documentos da base de dados, uma função que recebe uma parâmetro (*doc*), que é uma entrada da tabela e chama a função *emit* com dois parâmetros para serem mapeados no nome e no valor.

(Nota: o *end-point* “*/user/pingdb*”, funciona na mesma maneira mas para outro *end-point* diferente na base de dados, o *userlogin*).

Os *end-point* “*/insertGroup*”, “*/insertTeam*”, “*/deleteGroup*”, “*/deleteTeam*”, “*/:idGroup/teams”,*  “*/all*” são muito semelhantes, sendo que são se tratam de operações *CRUD* (*Create*, *Read*, *Update* e *Delete*) na base de dados. Os *end-point* “*/insertGroup*” são a criação de um novo documento na base de dados; o “*/insertTeam*” é a associação de uma equipa a um grupo; o “*/deleteTeam*” é a remoção dessa associação; o “*/deleteGroup*” é a remoção total de um grupo sendo o “*/all*” a leitura de todos os grupo, por fim o ultimo *end-point* é a leitura de um grupo em especifico.

Os *end-points* utilizados pelo ajax, “*/changeT/:idL”* e *“ /addT/:groupN/:idL/:idT”*, são utilizados para obter, no caso do primeiro *end-point*, as equipas de uma liga e o segundo *end-point* associar uma equipa em um grupo.

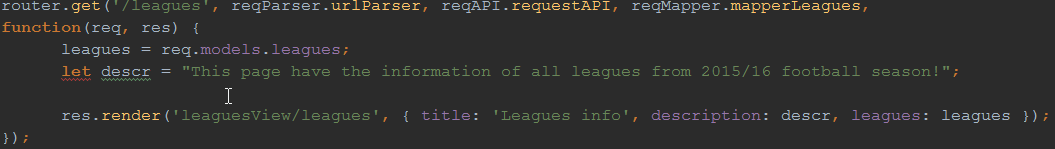
Por fim, os *end-points* *“/nextFixtures” e “/fixtures/:idL/:idT/:name/:groupName”* são para saber os jogos efectuados (ou os que faltam) de uma equipa.

O *leagueController*, é o controlador que faz a gestão de aceder a uma *Web* *API* externa, obter dados, mapear esses dados e mostrar ao utilizador final. Neste controlador o utilizador não autenticado pode apenas consultar as ligas; as equipas de uma liga; os jogos e a tabela classificativa de uma liga. Podendo apenas, o utilizador autenticado associar uma equipa a um grupo de favoritos que tenha criado.

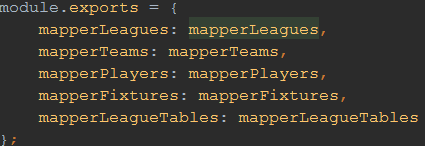
|  |  |  |
| --- | --- | --- |
| ***End-point*** | ***Método http*** | ***Ajax*** |
| */leagues* | ***GET*** | Não |
| */leagues/:idL/teams* | ***GET*** | Não |
| */leagues/:idL/teams/:idT/players* | ***GET*** | Não |
| */leagues/:idL/teams/:idT/teamFixtures* | ***GET*** | Não |
| */leagues/:idL/fixtures* | ***GET*** | Não |
| */leagues/:idL/leagueTables* | ***GET*** | Não |

Todos estes *end-points* são prefixados por *“/football-data”*, sendo que são responsáveis pela interacção entre a *API* e o *browser* (utilizador).

Os middlewares deste controlador seguem todos o padrão apresentado:



Há um *middleware*; o *urlParser*, que funciona como o *url* *resolver* que sabe com base no *end-point*, qual o *uri* que deve de ser obtido na *WebB* *API*; o *requestAPI* apenas faz o pedido ao recurso decidido no *middleware* anterior; e o *middleware* *mapperLeagues* faz parte de um conjunto de *middlewares* que fazem o mapeamento dos dados obtidos pela *WwbAPI* para objectos conhecidos à aplicação.



Este *controlador*, *leagueController*, é bastante importante no funcionamento da aplicação *Web*, pois é ele que define as rotas que aplicação servidora pode responder; quais as páginas *html* a serem apresentadas (com o conteúdo pedido pelo utilizador), bem como acesso a base de dados ou à *Web* *API*.

O *usersController*, é responsável pela permissão de utilizadores se conseguirem autenticar ou registar na aplicação *Web*.

|  |  |  |
| --- | --- | --- |
| ***End-point*** | ***Método http*** | ***Ajax*** |
| */user/login* | ***GET*** | Não |
| */user/login* | ***POST*** | Não |
| */user/logout* | ***GET*** | Não |
| */user/register* | ***GET*** | Não |
| */user/ register* | ***POST*** | Não |
| */user/pingdb* | ***GET*** | Não |

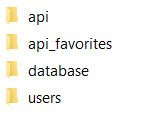
O *end-point “/pingdb”* já foi explicado o seu propósito anteriormente. Nos *end-points* “*/login”* e “*/logout”* (*GET* ou *POST*) são geridos pelo uso de um módulo externo, o módulo *passport* do *npm*. O *end-point “/register”*, recebe as credenciais de um utilizador que se quer registar na aplicação *Web*, o nome de utilizador, password e email. Essa informação é armazenada na base de dados *couchdb*, no documento *userlogin*.

## *middlewares*

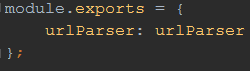
O package *middlewares* é o responsável pela gestão da lógica da aplicação. Funcionam um pouco como a “cola” da aplicação *Web*. Porque são eles que permitem, no caso do *package* *api*, gerir quais os recursos a serem obtidos assim como mapeá-los para aplicação.

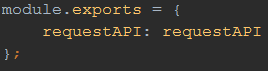
No caso do *package* *api\_favorites* e *database*, são muito semelhantes porque ambos fazem acesso à base de dados. Mas no caso do *api\_favorites* são feitos acesso mais específicos à *Web* *API*, este *package* foi criado para uma melhor gestão de obtenção de informação consoante as equipas favoritas.

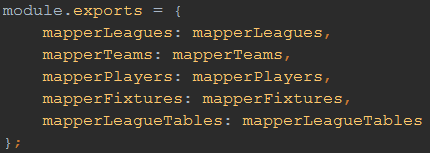
São estes *middlewares* que vão fazer parte da cadeia de *middlewares*, presentes nos pedidos *http* que aplicação *Web* está à “escuta”. Estes middlewares contêm os objectos, *request*, *response*, *next* que são transversais a essa cadeia de pedidos. No caso do *package* *api*, existe um objecto, conhecido pela aplicação o *req.models* que permite passar informação entre os *middlewares*, até ao *end-point*.



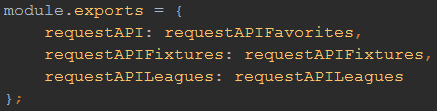
Do *package api*:



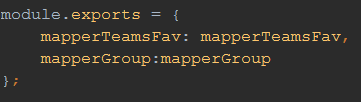


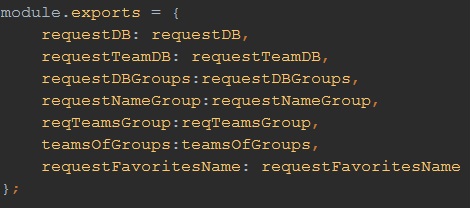


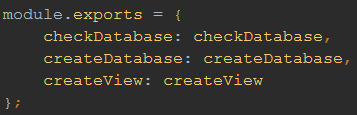
Do package api\_favorites:



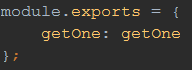
Do package database:







Do *package users*:



## *models*

Contêm ficheiros em formato .*json* com as propriedades que são conhecidas à aplicação Web, definem como devem de ser objectos conhecidos pela aplicação Web, “*football-data*”.



No caso do *package apiObj*, dizem respeitos aos objectos *fixtures*, *league*, *leagueTable*, *players* e *teams* provenientes da *Web API*.



Os objectos presentes em *couchdbView* são os objectos que vão fazer o mapeamento no *couchdb*.



## *node\_modules*

São os módulos presentes nesta aplicação *Web*, que permite o seu bom funcionamento.

O módulo *Express*, é que permite ter uma aplicação servidora, sendo os módulos *Express-handlebars* e o *hbs* responsáveis por definir as *views* (páginas *html*). O módulo *passport*, *Express-session* e *passport-local* são responsáveis por gerir utilizadores autenticados na aplicação. O módulo *request* tem um grande papel nesta aplicação *Web*, pois permite a comunicação entre a *Web* *API* e a base de dados, em conjunto com o módulo *node-couchdb*.

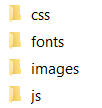
|  |  |
| --- | --- |
| ***body-parser*** | ^1.15.0 |
| ***cookie-parser*** | ^1.4.1 |
| ***debug*** | ^2.2.0 |
| ***express*** | ^4.13.4 |
| ***express-handlebars*** | ^3.0.0 |
| ***express-session*** | ^1.13.0 |
| ***hbs*** | ^4.0.0 |
| ***http*** | 0.0.0 |
| ***node-couchdb*** | ^0.4.0 |
| ***passport*** | ^0.3.2 |
| ***passport-local*** | ^1.0.0 |
| ***path*** | ^0.12.7 |
| ***request*** | ^2.69.0 |

## *public*

É o *package* responsável pela “embelezar” o *layout* das páginas *html* apresentadas, por via das *css* e dos *scripts* provenientes da *Framework* *Twitter* *Bootstrap*. Bem como os *scripts* utilizados para fazer pedidos *ajax* ao servidor e modificar a página dinamicamente, fazendo uso de *dom*.

Que são eles o *addTeamFavorites.js*, que adiciona uma equipa a um grupo e afecta página *html* via *dom* (sem fazer o *reload* da página). O *handleTeams.js* que permite obter as equipas de uma liga em especifico. Estas funções estão no valor do atributo *onchange* do elemento *html select*.

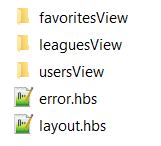
E por fim o *couchdbPing.js* e o *couchdbUserPing.js*, em que ambos são responsáveis por fazer pedidos assíncronos à base de dados, para saberem o seu estado.



## *views*

Construídas com base na *Framework* *Handlebars* são elas que são o componente gráfico que permite ao utilizador navegar pela aplicação *Web*. Definindo os *end-points* em elementos *html* <*a*>, com o atributo *href* ou o atributo *action* do *form*, que permite realizar os devidos pedidos *http* para os diversos *end-points*. Aqui neste *package* encontram-se todos os recursos que podem ser visualizados desta aplicação servidora.

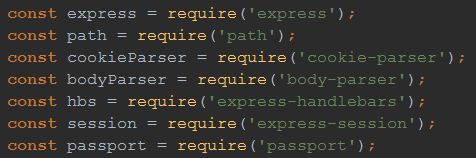
Em *layouts.hbs* são anexados os resultados das *views* geradas em *favoritesView*, *leaguesView* e *usersView*. Sendo a *error.hbs* quando o *middleware* “*return next(err);*” é chamado.



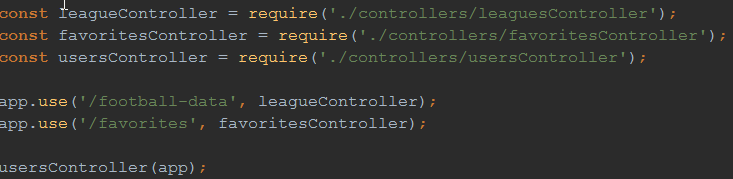
## *root & bin*



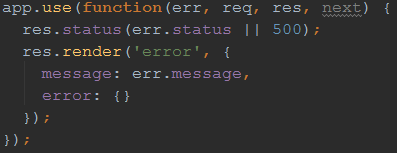
É em *app.js* que se encontram os módulos apresentados anteriormente, a serem registados na aplicação. Registo do módulo *passport* ou do módulo *Express-handlebars*, são exemplos de alguns dos registo de *middlewares* feitos neste ficheiro.



São também registados os *controllers* para aplicação servidora ficar “à escuta” dos *end-points* em que se registou. Em *package.json* estão as informações básicas da aplicação *Web* já mencionadas anteriormente.



Por sua vez, são também registados os *middlewares* que tratam os erros.



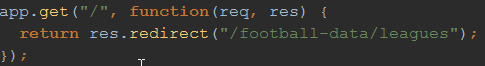
É no módulo *app.js* que são definidas as convenções para os atributos *src* (do elemento *html*, *script*) e *href* (do elemento *html*, *link*).





Apenas sendo necessário dar o caminho relativo, desde a directoria, ao invés do colocar o caminho total dos “anexos” às páginas *html*.

Por fim existe uma definição da *homepage* da aplicação *Web*, “*football-data*”:

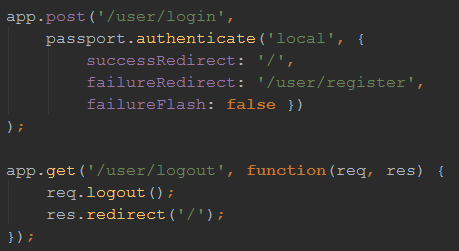


Existe um ficheiro que se encontra dentro da directoria *bin*, que recebe o *middleware* gerado em *app.js* que vai iniciar o servidor, *http* no porto 3000 (por defeito), com todos os *middlewares* registados. É o *entry-point* (ou a *main function*) da aplicação *Web*, “*football-data*”



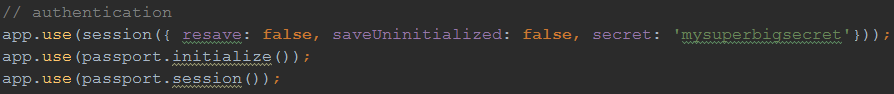
# *Autenticação*

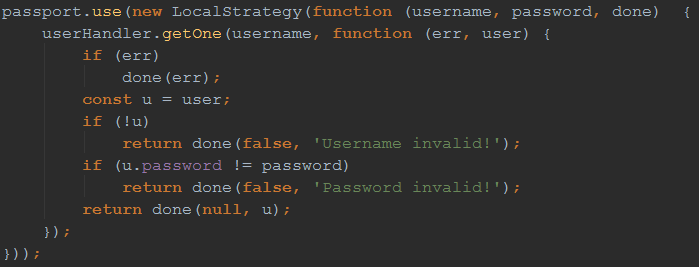
Um dos requisitos da aplicação *Web*, “*football-data*”, é gestão e criação de utilizadores autenticados. É utilizado um *middleware* que faz o acesso ao documento *userlogin*, onde estão armazenadas as credenciais de utilizadores, para obter um utilizador válido.



Isto é possível através da função *getOne(username, callback*), esta função obtém o utilizador e passa-o à *callback*. Para ser utilizado em dois *middlewares*, necessários para serem registados pelo módulo *passport*. O registo de uma estratégia, que é um conceito utilizado pelo módulo para delegar autenticação; para a *desserialização*/*serialização* de um utilizador, necessária para remover/criar a propriedade *req.user*. Sendo esta propriedade importante para saber se existe um utilizador autenticado na aplicação.

Em *app.js* é necessário registar alguns *middlewares*, que vão definir uma autenticação persistente na aplicação, exemplo: sair do *browser* sem fazer *logout* e voltar à pagina sem se autenticar.





# *Conclusões*

Durante o projecto definido como trabalho final da unidade curricular, Programação na Internet. O grupo conseguiu realizar uma aplicação semelhante a uma real e actual com muitas tecnologias a fazerem parelha. O grupo definiu algumas estratégias que considerou importantes, para a realização desta aplicação Web.

A **divisão**, é uma palavra-chave deste projecto, “dividir para reinar” é um paradigma que foi partilhado com a utilização de uma linguagem interpretada para a realização da componente servidora. Isto trás muita liberdade, mas tem os seus contras, por ser uma tecnologia em constante expansão, tem o contra de no futuro certas versões não funcionaram como o desejado.

O grupo tentou aproveitar ao máximo a “*beleza*” do *node.js*, os *middlewares*. Pois permitem realizar importante no atendimento a um pedido *http* feito à aplicação. Sendo todo o fio de execução uma *thread* que atende o pedido, é importante não existir código “*bloquante*” e o uso do objecto *next*, para haver progressão na cadeia de middlewares até ao *end-point*. Não esquecer o objecto *req.models* que é partilhado, por cada pedido entre cada *middleware*.

Houve um reaproveitamento daquilo que as *frameworks* utilizadas nesta aplicação, têm para oferecer. No caso das css e alguns scripts, a utilização da *framework Twitter* *Bootstrap* veio permitir ter páginas *html* com um visual simplista e intuitivo. A junção desta *framework* com a *framework* *Handlebars*, permitiu a construção de páginas *html* com informação relevante ao utilizador podendo ter uma navegação/experiência positiva na aplicação *Web*.