

Universidade do Minho - Escola de Engenharia

Relatório do trabalho prático de Engenharia Web

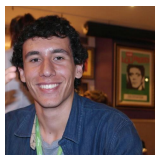
BetESS - PARTE II

Autores :

Diana Costa (A78985)



Marco Silva(A79607)



Resumo

O presente relatório apresenta a segunda parte do projeto da unidade curricular de Engenharia Web, onde se implementa uma aplicação *server-based* e um *client browser*, com base na especificação elaborada na primeira parte, através do *WebRatio*. O ideia do desenvolvimento de uma "*Betting House Web App*" provém da UC de Arquiteturas de Software, onde se elaborou um projeto - BetESS -, e todos os requisitos associados a uma plataforma de apostas.

Conteúdo

| | | |
|----------|---|-----------|
| 1 | Introdução | 3 |
| 2 | Problema | 4 |
| 3 | Base de Dados | 5 |
| 3.1 | Versão WebRatio | 5 |
| 3.2 | Versão MongoDB | 6 |
| 4 | Implementação | 7 |
| 4.1 | Análise de requisitos | 7 |
| 4.2 | Análise de tecnologias | 8 |
| 4.3 | Arquitetura da solução - back-end | 9 |
| 4.4 | Front-end | 11 |
| 4.4.1 | Login/Registo | 11 |
| 4.4.2 | Área de Utilizador | 12 |
| 4.4.3 | Área de Administração | 15 |
| 4.4.4 | Directrizes de Acessibilidade de Conteúdo da Web (WCAG) | 19 |
| 5 | Conclusões e Sugestões | 20 |

1 Introdução

Este projeto foi elaborado no âmbito da Unidade Curricular de Engenharia Web, do 4^a ano do Mestrado Integrado em Engenharia Informática. Numa primeira fase, definiu-se a interação entre *users* e uma aplicação, utilizando a ferramenta *WebRatio* e a linguagem IFML. Para a fase presente, era necessário que se desenvolvesse uma aplicação *server-based* escalável e com resposta em tempo real, e um *browser client* reativo, acessível, responsivo, fácil de usar e seguro, com base na especificação da primeira parte. O sistema a desenvolver segue a mesma linha de raciocínio do projeto "BetESS", já realizado pelo grupo, que consistia na implementação de uma plataforma de apostas. As funcionalidades a desenvolver mantêm-se, e apenas seria necessário adicionar, para esta unidade curricular, utilizadores *premium*, que teriam acesso a eventos exclusivos, desde que pagassem a quantia requerida para tal. Assim, esta aplicação deverá suportar o registo de apostadores, e será gerida por um administrador, que deverá criar eventos, associando as equipas existentes com as *odds* respetivas. Os utilizadores registados podem aceder a uma lista de eventos disponíveis e realizar uma ou mais apostas, sendo que apenas possível apostar num resultado (apostas sobre múltiplos resultados não são permitidas). Aquando da realização da aposta, é necessário que o jogador indique a quantia a apostar e o resultado pretendido, para que o sistema possa manter uma lista das apostas realizadas por cada apostador. Esta aposta, que possui um estado conforme a sua condição, quando passar de aberta para fechada, notifica os apostadores do resultado da mesma, e se for o caso, do valor ganho em *ESScoins* conforme a *odd*.

Assim, e como todos os requisitos, esquemas de dados e interações estão já definidos, neste relatório pode-se constatar o trabalho desenvolvido pelo grupo referente a duas fases: uma primeira, onde se planeou e desenvolveu o *back-end*, de forma a servir de base e lógica de toda a aplicação, e uma outra, onde se elaborou o *design* do *front-end*, de forma a dar vida a todas as operações de *backend* envolvidas. Todas estas fases implicaram a tomada de decisões pelo grupo, que são explicadas ao longo do relatório.

Deste modo, o relatório começa por introduzir o problema e declarar e explicar a base de dados. De seguida, apresenta-se toda a arquitetura e decisões envolventes do *back-end*. Nesta secção são discutidas as várias tecnologias e *frameworks* utilizadas, juntamente com a arquitetura proposta e as decisões do grupo de forma a alcançar os requisitos não-funcionais propostos pela equipa docente. Depois, noutra secção, elabora-se o mesmo raciocínio mas para o *front-end*, onde se pretendeu alcançar uma interface de, pelo menos, nível "A", segundo a WCAG (*Web Content Accessibility Guidelines*). O relatório termina com as conclusões, em que o grupo faz uma análise do trabalho desenvolvido ao longo do semestre, tendo em conta as dificuldades obtidas.

2 Problema

Pretende-se simular um sistema que possibilite o registo de utilizadores, que indicam uma quantia monetária (*ESScoins*) disponível para aposta. Ao verificar uma lista de eventos, o apostador (premium ou normal) escolhe um evento "Aberto" (também premium ou normal) com a respetiva *odd* (no formato equipa1 - equipa2(<*odds* respetivas>)), e decide apostar uma quantia, sendo que o sistema deverá manter uma lista de apostas por utilizador. Quando as apostas realizadas por um *user* passarem ao estado "Fechada" (assume-se que o resultado do evento é conhecido), o utilizador deverá ser notificado de tal ocorrência, juntamente com o seu ganho com a mesma. Para determinar o valor ganho numa aposta deverá ser definido para cada evento as *odds* para os possíveis resultados e sobre essas *odds* será calculado o valor ganho numa aposta. Exemplificando, no caso de vitória do SC Braga na aposta "FC Barcelona – SC Braga (1.38 , 4.40, 8.00):

1. os apostadores que tenham indicado a vitória do SC Braga recebem 8.00 *ESScoins* por cada *ESScoin* apostada;
2. os apostadores que tenham indicado a vitória do FC Barcelona ou o empate recebem 0.

É de realçar que **não existe o estado "A decorrer"** como em muitos sistemas reais, sendo que este, no sistema do grupo, é apenas ilusório. Não serão permitidas efetuar operações como atualizar *odds* ou *cashout* enquanto um evento decorre.

Definido o problema base a resolver, no que toca aos requisitos não-funcionais, é necessário que a aplicação tenha:

1. **Resposta em tempo real:** a aplicação lida com apostas em jogos e deve garantir que novas apostas não sejam possíveis após o final do jogo;
2. **Escalabilidade:** a plataforma deve ter uma quantidade enorme de pedidos em determinados jogos (por exemplo, *Super Bowl*, Liga dos Campeões da Europa) e deve responder sem problemas;
3. **Interface:** deve seguir, pelo menos, o nível "A" das Directrizes de Acessibilidade de Conteúdo da Web (WCAG). Esta deve ser reativa, acessível, responsiva, fácil de usar e segura.

3 Base de Dados

Para ser possível a persistência dos dados, o grupo viu-se, numa primeira fase, na obrigação da criação de uma base de dados. Esta tarefa foi facilitada pelo *WebRatio*, que permite a criação de um *domain model* que automatiza este processo. Ainda assim, é importante notar que **as cardinalidades nesta ferramentas são opostas às que seriam observáveis num modelo lógico de dados comum**.

Posto isto, e como também seria necessário uma BD para a segunda fase, o grupo vai detalhar o raciocínio utilizado para a elaboração do domain model, assim como a sua evolução para a nova base de dados, em MongoDB, que implicou o aparecimento de dois modelos um pouco diferentes dadas as suas naturezas relacionais e não-relacionais.

3.1 Versão WebRatio

Numa primeira fase, o grupo analisou a estrutura de dados proveniente da primeira fase do projeto da unidade curricular, que se apresenta abaixo. São visíveis as entidades:

- **Desporto**, que, ligado a eventos, envolve várias ligas e equipas;
- **Equipa**, já que um evento envolve sempre duas equipas;
- **Liga**, interligada a um evento, que possui várias equipas;
- **Evento**, envolvendo equipas, ligas e desportos;
- **Aposta**, que é efetuada sobre um evento;
- **Notificação**, que está relacionada com um utilizador e com um evento;
- **Utilizador**, onde são armazenados os dados de um *user*;
- **Group**, entidade típica do *WebRatio* para definir grupos de utilizadores;
- **Module**, que serve para atribuir *site views* aos diferentes tipos de utilizadores.

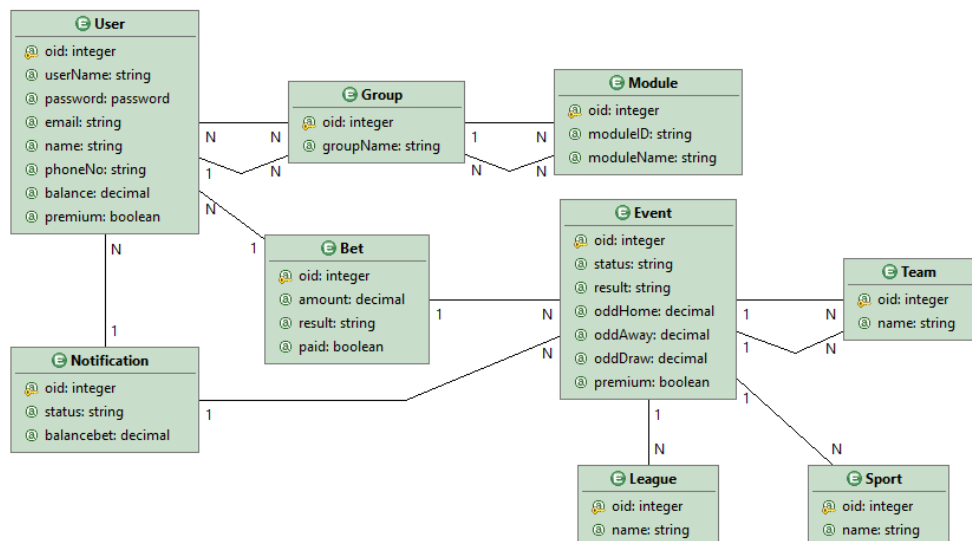


Figura 1: *Domain Model* elaborado no *WebRatio*.

Omitem-se as restantes explicações do modelo de dados, uma vez que o mesmo já fora explicado no relatório anterior tendo em conta a rapidez de *queries* e de forma a responder aos requisitos do projeto.

3.2 Versão MongoDB

Para esta fase, e dados os requisitos do projeto, optou-se por passar de um contexto relacional para um não-relacional. Esta alteração do paradigma permite alcançar:

- **Maior escalabilidade e flexibilidade:** mais adequado à arquitetura de micro-serviços, já que tem mais flexibilidade para alavancar enormes quantidades de dados estruturados, semi-estruturados ou desestruturados (associados ao conceito de distribuição);
- Grande volume de dados;
- **Alto desempenho:** oferece velocidade no acesso a dados e respostas a *queries* mais rápidas;

Assim, apresenta-se abaixo um esquema representativo das coleções. As duas base de dados obtidas provêm da divisão do *data model* do *WebRatio* em dois, que servirão dois microserviços (tal divisão será explicada na próxima secção). Assim, para o microserviço nº1, a sua base de dados refere eventos, ligas, desportos e equipas. Para o microserviço nº2, a base de dados é constituída por apostas, notificações e utilizadores.

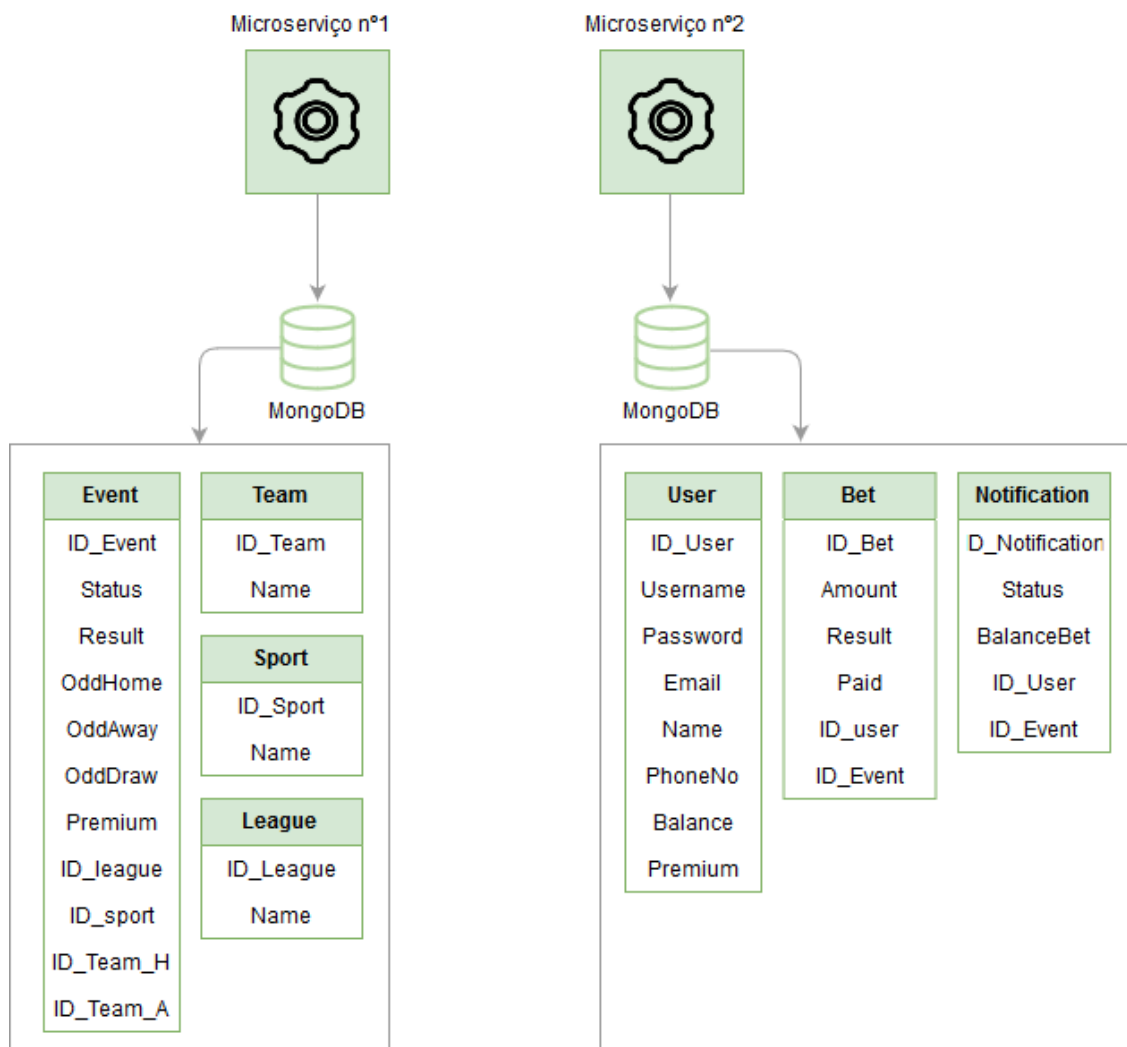


Figura 2: Esquema das novas bases de dados em MongoDB

4 Implementação

Nesta fase, a equipa iniciou o desenvolvimento do *back-end* e *front-end* requeridos, para além da implementação de toda a lógica relacionada com o *reverse proxy* e mecanismo de troca de mensagens entre microserviços, arquitetura selecionada. Para esta fase, teve de se decidir sobre quais as ferramentas a utilizar, de acordo com as necessidades correntes.

4.1 Análise de requisitos

Uma vez que todos os requisitos, funcionais e não-funcionais, provêm já do projeto desenvolvido na UC de Arquiteturas de *Software*, não há muito a acrescentar senão os pedidos especificamente para este projeto.

Desta forma, os requisitos não-funcionais "Escalabilidade" e "Resposta em tempo real" requeridos para a aplicação cumprem-se pela arquitetura de microserviços e pelo sistema implementada pelo grupo. Esta arquitetura flexível permite alcançar **escalabilidade**, especialmente se o esquema de dados também estiver dividido e desenhado para tal. Enquanto que, se for necessário escalar algo num sistema típico monolítico, todo ele tem que escalar, mesmo componentes que não necessitam de tal, numa arquitetura orientada a microserviços esta tarefa é facilitada, pois os sistemas podem ser dimensionados de acordo com as demandas dos componentes. Para isto, e para suportar a escalabilidade, note-se a esquematização dos dados da secção anterior (3.2). Optou-se por dividir as tabelas inicialmente existentes tendo em conta uma **orientação à funcionalidade**. Através desta abordagem foi possível perceber quais as dependências entre as diversas entidades de dados. Assim, uma vez que a tarefa de maior complexidade é o fecho de um evento desportivo, as entidades Bet, User e Notification serão alojadas no mesmo serviço uma vez que apresentam um elevado nível de interdependência entre elas. A entidade Event entra também no fecho de um evento mas, uma vez que as ações a realizar sobre este modelo de dados são bastante simples, optou-se pela utilização de um mecanismo de comunicação entre micro serviços (*RabbitMQ*) para a atualização do estado do evento para fechado. As restantes entidades Event, Team, Sport e League ficariam localizadas num outro microserviço, dado que são maioritariamente acedidas e monitorizadas pelo administrador, e dado o nível de dependência das mesmas: um evento está sempre associado a duas equipas, um desporto e uma liga.

Quanto à **resposta em tempo real**, tal é alcançada automaticamente pela natureza do sistema implementado que utiliza *Node.js*, criando assim aplicações de alta escalabilidade (como um servidor web), capazes de manipular milhares de conexões/requisições simultâneas em tempo real, numa única máquina física. Exemplificando com o fecho de um evento, no momento em que tal ação é efetuada, para além de todos os apostadores receberem os seus ganhos (caso acertem no resultado) e uma notificação, o estado do evento é alterado para "Fechado", o que impede o mesmo voltar a aparecer na página de eventos disponíveis do utilizadores que, consequentemente, não pode apostar mais a partir do final do jogo.

Quanto aos requisitos da interface, que precisava de seguir, no mínimo, o nível "A" das WCAG, tal será analisado com mais detalhe na secção 4.4.4.

4.2 Análise de tecnologias

No que toca à seleção de tecnologias para o sistema proposto, foi necessário analisar todas as componentes de uma aplicação *full-stack* orientada a microserviços. Assim, resumem-se as tecnologias e *frameworks* selecionadas, juntamente com uma breve explicação da razão da sua escolha.

- **Vue.js:** é uma *framework* de *JavaScript* para criar *user interfaces* e aplicações *single-page*. Escolhida pela sua simplicidade, rapidez, modularidade e "suavidade" a executar as mesmas tarefas de outras *frameworks*;
- **Bootstrap:** é uma *framework* web com código-fonte aberto para desenvolvimento de componentes de interface e front-end para sites e aplicações web usando HTML, CSS e *JavaScript*, melhorando a experiência do utilizador num ambiente amigável e responsivo. Escolhida pelo elevado número de recursos acessíveis, e pela diminuição do esforço de escrita de código, uma vez que oferece blocos de código *pre-built* ao invés de ter que ser estruturado de raiz;
- **jQuery:** é uma biblioteca de funções *JavaScript* que interage com o HTML, desenvolvida para simplificar os *scripts* interpretados no navegador do cliente. Selecionada pois faz com que seja mais fácil a escrita de código JS no *front-end*;
- **Node.js:** é um interpretador *open source* de código *JavaScript* de modo assíncrono e orientado a eventos, focado em migrar a programação do *Javascript* do lado do cliente para os servidores, criando assim aplicações de alta escalabilidade (como um servidor web), capazes de manipular milhares de conexões/requisições simultâneas em tempo real, numa única máquina física. Por estas razões, e pela sua popularidade, este interpretador foi selecionado para o *server-side*;
- **Express:** é uma estrutura de aplicações web para o *Node.js*, lançada como *software* livre e *open source*. Selecionada pela sua simplicidade e popularidade em lidar com a gestão de tudo, desde as rotas, até a lidar com pedidos e *views*;
- **Vagrant:** é um produto de *software open source* para criação e manutenção de ambientes de desenvolvimento de *software* virtual portáteis. Escolhido para lidar com os diferentes microserviços, entre os quais, os referentes à autenticação, mecanismo de mensagens, entre outros;
- **Nginx:** é um servidor HTTP e *reverse proxy*, útil para lidar com o redirecionamento de pedidos que chegam à API Layer. Ferramenta útil para, se se pretender continuar o projeto futuramente, poderem ser implementados balanceamento de carga, caching, compressão de respostas grandes, entre outros;
- **RabbitMQ:** é um *software* de mensagens de *open source*, utilizado pelo grupo para a implementação de mensagens entre os microserviços;
- **MongoDB:** base de dados escolhida, pelas razões já apresentadas na secção 3.2.

4.3 Arquitetura da solução - back-end

Como já referido, o grupo utilizou o *Vagrant* de forma a lidar com os diferentes serviços existentes, entre eles, serviço de autenticação, de API *layer*, serviço Users/Bets/Notifications e, em quarto, Events/Leagues/Sports/Teams. Toda esta infraestrutura encontra-se implementada em código no ficheiro *Vagrant* e *scripts* de instalação para cada instância.

Para estes serviços comunicarem surge o *RabbitMQ*, ferramenta orientada para a execução de tarefas de uma forma assíncrona. Este projeto necessita de respostas síncronas, o que, não sendo o foco do *RabbitMQ*, constituiu um desafio para implementação.

Para a implementação da API *layer*, e como já referido, utilizou-se o *Nginx*. O facto de ser o ponto único de entrada na infraestrutura, sendo responsável pelo reencaminhamento dos pedidos para os serviços respetivos e consequente devolução das respostas ao cliente, necessitaram de uma atenção extra, especialmente por ter sido o primeiro contacto do grupo com a implementação de um *reverse proxy* desta natureza.

Mostra-se, de seguida, a solução proposta pelo grupo quanto à arquitetura do sistema orientado a microserviços.

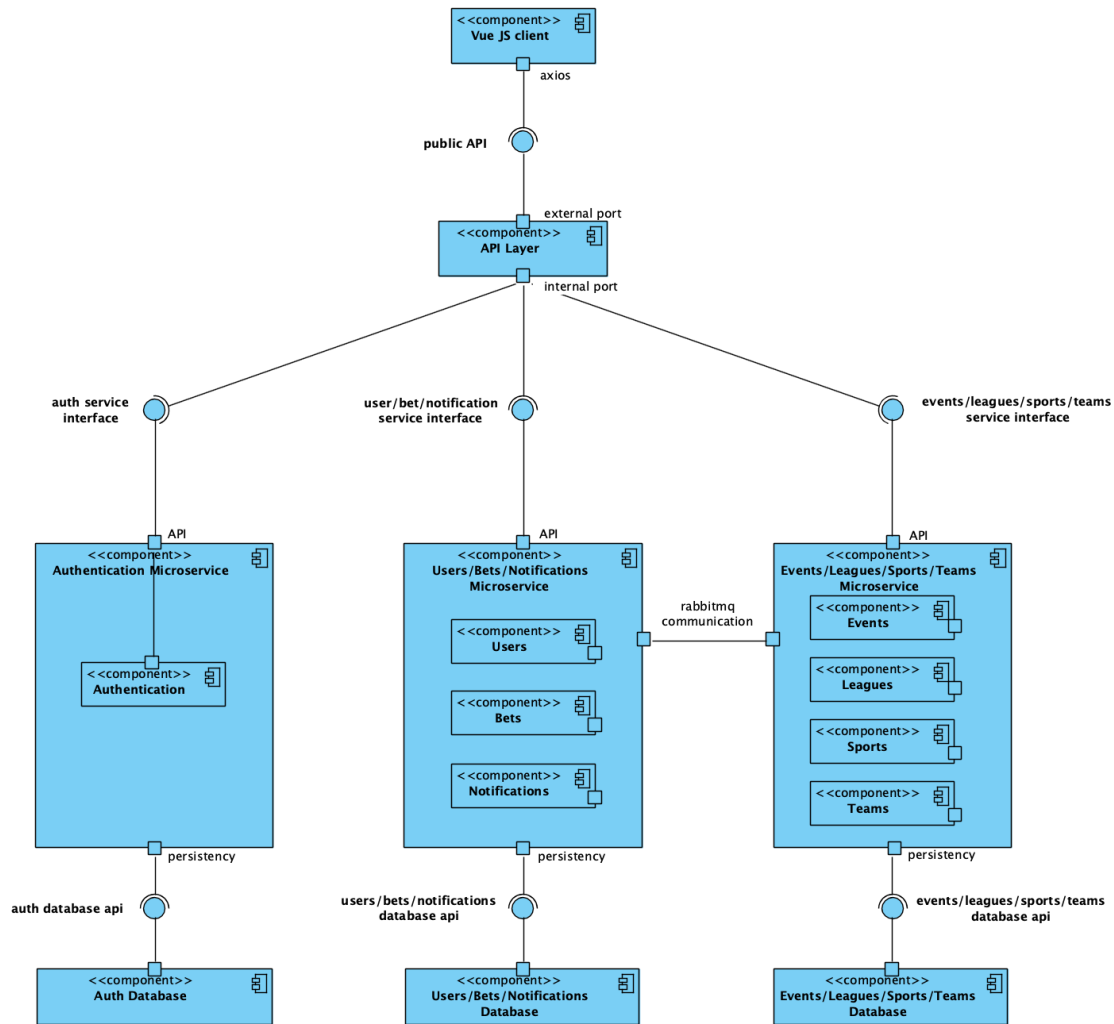


Figura 3: Arquitetura do sistema, orientada a microserviços

Finda a explicação de toda a base do projeto, e detalhado o propósito de cada microserviço, resta explicar o método de autenticação que constituiu um impasse inicial pelas diversas maneiras de resolver o problema. Assim, decidiu-se que um *token* seria o mais adequado à versatilidade método de autenticação e à simplicidade da autenticação de um pedido, bem como a identificação do utilizador autenticado. A imagem seguinte esquematiza a abordagem utilizada pelo grupo.

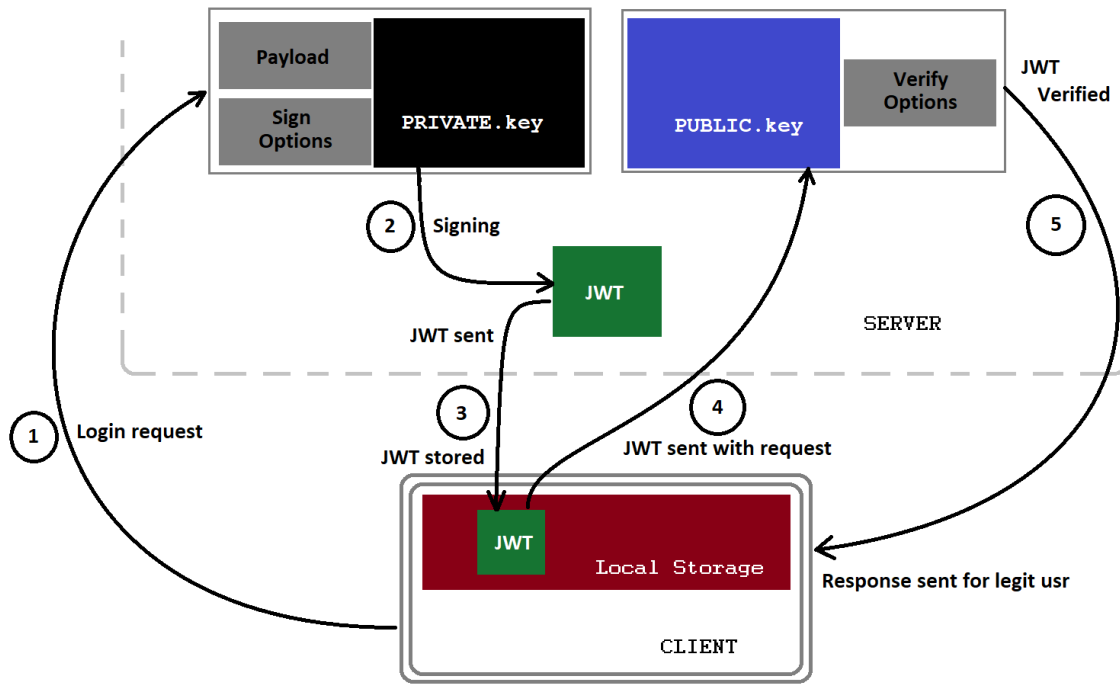


Figura 4: Esquema do mecanismo de autenticação através de *tokens*.

Foi, então, necessário recorrer a um par chave pública/chave privada para a criação, descodificação e verificação da veracidade do *token* (utilizando um gerador automático *online* desses pares de chaves - algoritmo RS256). Este mecanismo permite inserir toda a informação necessária para a identificação de um utilizador (o seu *username*, o seu *id*, se é um *user premium* ou não, ...). Neste momento, o *token* criado pelo grupo não tem uma *expiration date*, pelo que não é necessário renová-lo. Desta maneira, o *user* está sempre pronto para se autenticar, desde que possua o *token*. É de notar que o objetivo do grupo seria passar o *token* pelos *headers* dos pedidos. No entanto, um problema de configuração do *Nginx* e do *axios* não resolvido a tempo levou a que a solução passasse por este campo fluir através do *body*, facto que deverá ser corrigido numa fase futura de desenvolvimento deste projeto. No entanto, em fase de teste através do *Postman*, o *token* era corretamente encaminhado pelo *header*.

Assim, e como é típico, se um *user* tentar efetuar o *login* sem estar registado no sistema, o sistema apresentará uma mensagem de erro. O mesmo acontece se um utilizador registado tentar autenticar-se e as suas credenciais não coincidirem com as que se encontram na base de dados. Para acrescentar mais um nível de segurança, foi implementado um redirecionamento de links. Visto que a primeira página que um utilizador entra em contacto é a de *sign in/sign up*, este não deveria ter acesso a links "internos" se não tivesse o *token*. Desta forma, se um *user* não autenticado tentar entrar na página de apostas, é redirecionado para a página de *sign in/sign up*.

4.4 Front-end

Na primeira fase do projeto, e como resultado de todo o planeamento, elaboraram-se interações com o utilizador através da linguagem IFML no *WebRatio*, das quais resultaram as *mockups* do projeto. Estas são imagens ilustrativas de como a página *web* deve ser organizada, sendo que o seu aspeto é básico, mostrando todas as funcionalidades expectáveis. Permitem também ao cliente ter uma ideia do que vai obter, podendo apresentar críticas com vista à melhoria do sistema. Adicionalmente, são um bom suporte aquando do desenvolvimento. Como tal, dispensam-se quaisquer elaborações adicionais de *mockups*, uma vez que provêm já da fase anterior. Todas as páginas da interface serão expostas e detalhadas abaixo.

4.4.1 Login/Registo

A autenticação no sistema é feita através da *email* e da *password* do utilizador que se pretende autenticar. Esta funcionalidade foi desenvolvida através de *cookies*, pelo que, quando os dados estão devidamente inseridos, o servidor cria um *cookie* que é armazenado pelo *browser* do utilizador até este terminar sessão (é fornecido um token ao utilizador que é armazenado no *browser*). Desta forma, garante-se que a sessão é salvaguardada em todas as páginas do *browser*. A *password* inserida é comparada com aquela que está, à partida, guardada na base de dados. Aqui deveria ter sido usado um método de encriptação, que usaria um comparador de hash neste ponto, já que a *password* seria encriptada no momento do registo do utilizador. Isto seria mais seguro, e será implementado numa continuação futura deste projeto.

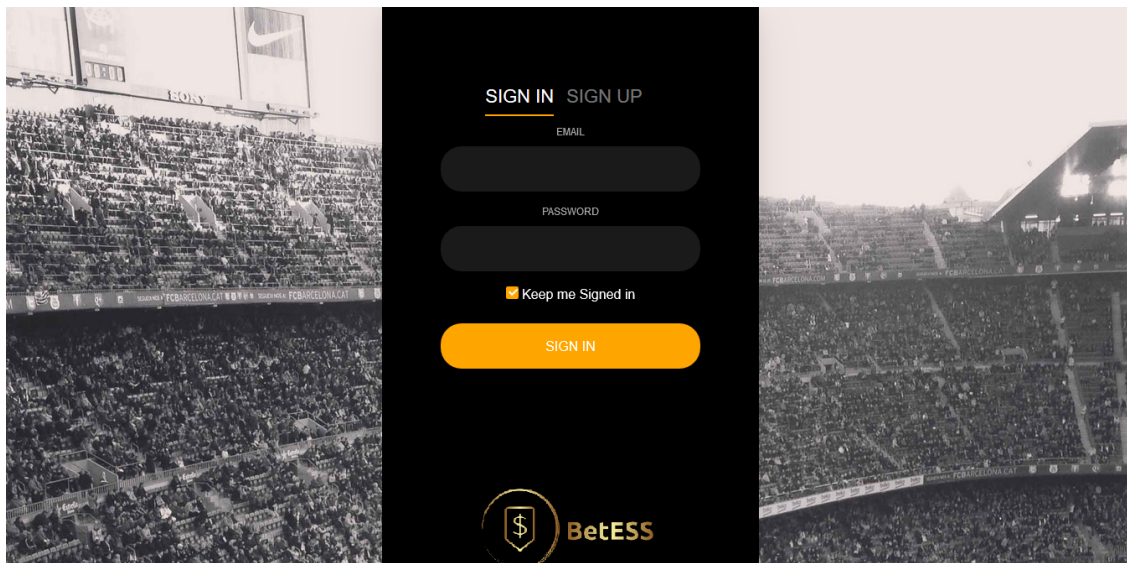


Figura 5: Página de *login*

O registo no sistema pressupõe a inserção de dados obrigatórios, tais como *username*, nome, *email*, contacto e *password*. A *password* deveria ser encriptada no momento do registo, por questões de segurança, como já mencionado.

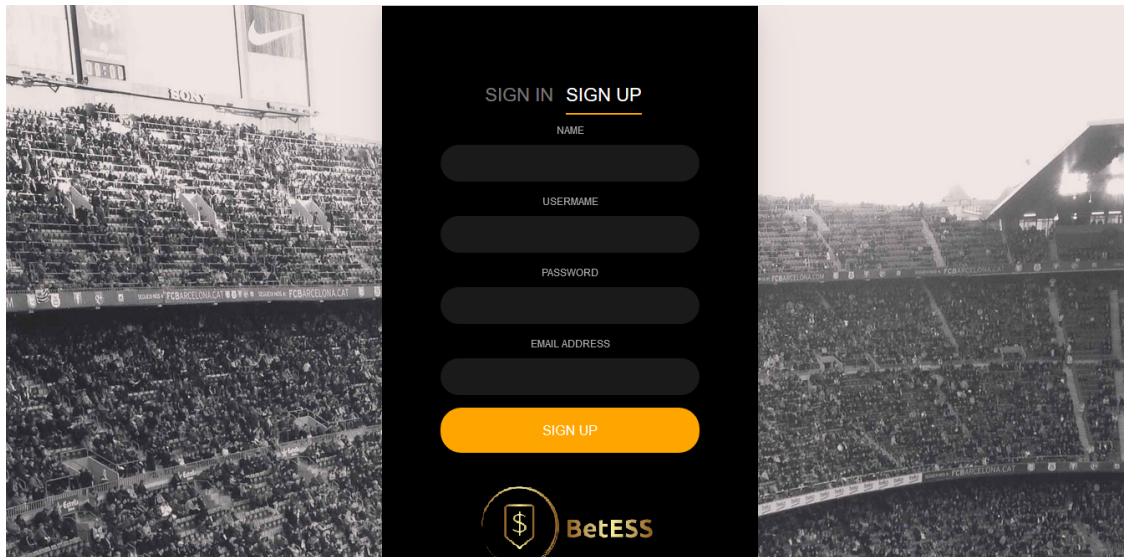


Figura 6: Página de registo

4.4.2 Área de Utilizador

Quando um utilizador se autentica na plataforma, depara-se com a sua *home*. Aqui, e através do menu superior da página, o *user* é redirecionado para as páginas onde poderá apostar, ver as suas apostas, gerir os seus créditos, ver e editar o seu perfil, verificar as suas notificações ou sair do sistema.

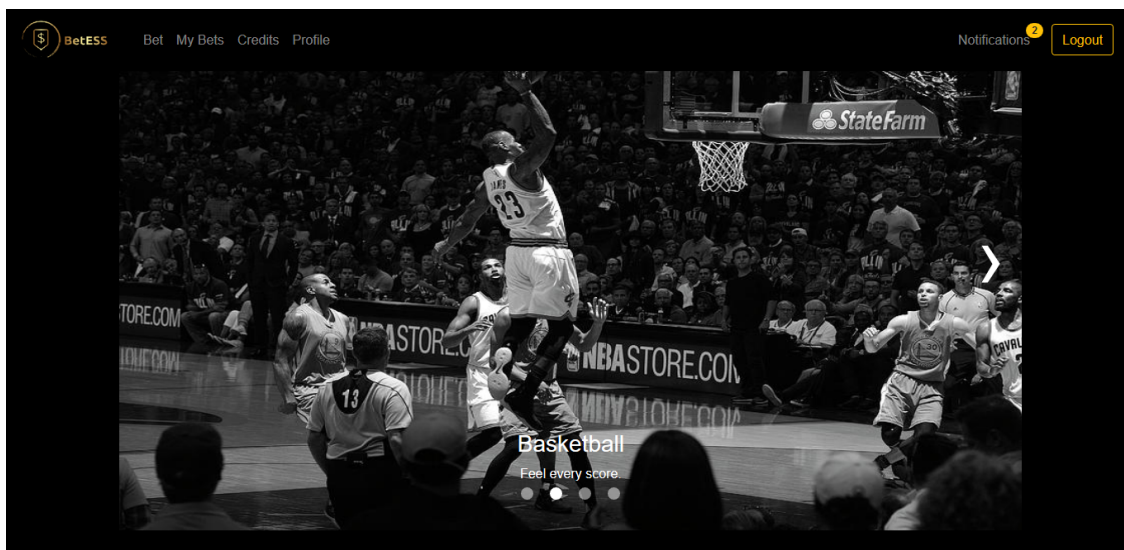


Figura 7: *Home* do utilizador

Na página de apostar, em primeiro lugar, o utilizador deverá inserir a quantia que pretende apostar (em ESScoins), e, de seguida, consultar os eventos disponíveis na altura. Depois de escolhido o evento, e seleccionada a *odd* que pretende apostar, basta clicar em submit para que a aposta seja efetuada e fique registada na página "*My Bets*".

BetESS Bet My Bets Credits Profile Notifications **Logout**

Available Events

1st Insert the amount to bet
10

2nd Choose the odd from the desired event

Submit

FCPorto - SLBenfica
 Soccer, Primeira Liga
 Possible Gains: 20 ESScoins
 1.5 3.1 1.3

GS Warriors - TOR Raptors
 Basketball, EUA - NBA
 Possible Gains: 10 ESScoins
 3.4 1.9 0.4

ChelseaFC - ArsenalFC
 Soccer, Premier League
 Possible Gains: 12 ESScoins
 1.3 2.2 3.0

EvertonFC - ManchesterUFC
 Soccer, Premier League
 Possible Gains: 222 ESScoins
 **** Premium ****
 4.5 1.2 2.1

Figura 8: Apostar (utilizador)

Na página das apostas do utilizador, encontram-se todas as apostas em eventos (ainda não terminados), com alguns detalhes, juntamente com a opção de *cashout*, que permite a um *user* desistir da aposta, sendo-lhe devolvido apenas 50% do saldo que investiu.

BetESS Bet My Bets Credits Profile Notifications **Logout**

Your Current Bets

FCPorto - SLBenfica
 Soccer, Primeira Liga
 Possible Gains: 20 ESScoins
 Cashout

GS Warriors - TOR Raptors
 Basketball, EUA - NBA
 Possible Gains: 10 ESScoins
 Cashout

ChelseaFC - ArsenalFC
 Soccer, Premier League
 Possible Gains: 12 ESScoins
 Cashout

EvertonFC - ManchesterUFC
 Soccer, Premier League
 Possible Gains: 222 ESScoins
 **** Premium ****
 Cashout

Figura 9: Lista de apostas (utilizador)

A página de gestão de créditos tem como fim carregar e levantar dinheiro da plataforma. Considera-se que o dinheiro é carregado/levantado diretamente de/para a conta bancária do *user*.

The screenshot shows the 'Credits Management' page. At the top, there is a navigation bar with 'Bet', 'My Bets', 'Credits', and 'Profile'. On the right, there are 'Notifications' and a 'Logout' button. The main content area is titled 'Credits Management' and contains two side-by-side forms: 'Draw' and 'Deposit'. The 'Draw' form has a text input field with the value '10' and a 'Submit' button. Below the input field, it says '* The amount will be returned to your bank account.' The 'Deposit' form has a text input field and a 'Submit' button. Below it, it says '* The amount will be withdrawn from your bank account.' At the bottom of the page, it displays 'Current Balance: 460 ESScoins'.

Figura 10: Gestão de créditos (utilizador)

Em relação ao perfil do cliente, este pode consultar os seus dados, para além de editar o seu contacto e a *password* apenas. Aqui poderá também efetuar o *upgrade* para *user premium*, se já não o for, o que lhe custará 50 ESScoins, e lhe dará acesso a apostar eventos *premium* não disponíveis para um jogador normal.

The screenshot shows the 'Your Profile' page. At the top, there is a navigation bar with 'Bet', 'My Bets', 'Credits', and 'Profile'. On the right, there are 'Notifications' and a 'Logout' button. The main content area is titled 'Your Profile' and contains a user profile card for 'João Oliveira', a 'BetESS Member - Not Premium'. The card displays the following information: Username: 'Loli_25', Email: 'Loli@uminho.pt', Phone number: '919191919', Password: '*****', Balance: '350 ESScoins', and a 'Premium?' section with an 'Upgrade' button. Below the 'Upgrade' button, there is a small note: '* Being a premium user gives you access to unique events and opportunities, to the simple amount of 50 ESScoins'.

Figura 11: Perfil (utilizador)

Por fim, na página de notificações, o utilizador recebe o estado de todos os eventos que fecharam, nos quais ele tinha apostado. Aqui ele percebe se perdeu ou ganhou ESScoins, e pode marcar as notificações como lidas.

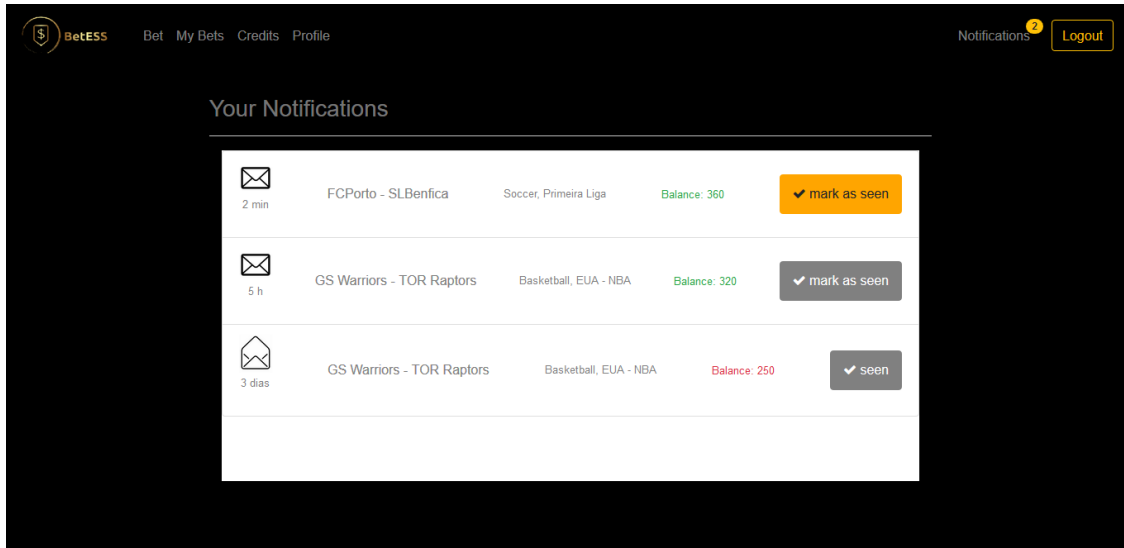


Figura 12: Notificações (utilizador)

4.4.3 Área de Administração

Quando um administrador se autentica na plataforma, depara-se com a sua *home*. Aqui, e através do menu superior da página, o *admin* é redirecionado para as páginas onde poderá adicionar, remover e verificar desportos, ligas e equipas. Pode, também, verificar as apostas dos utilizadores e ver, gerir e criar os eventos. Por fim, através do *logout*, poderá sair do sistema.

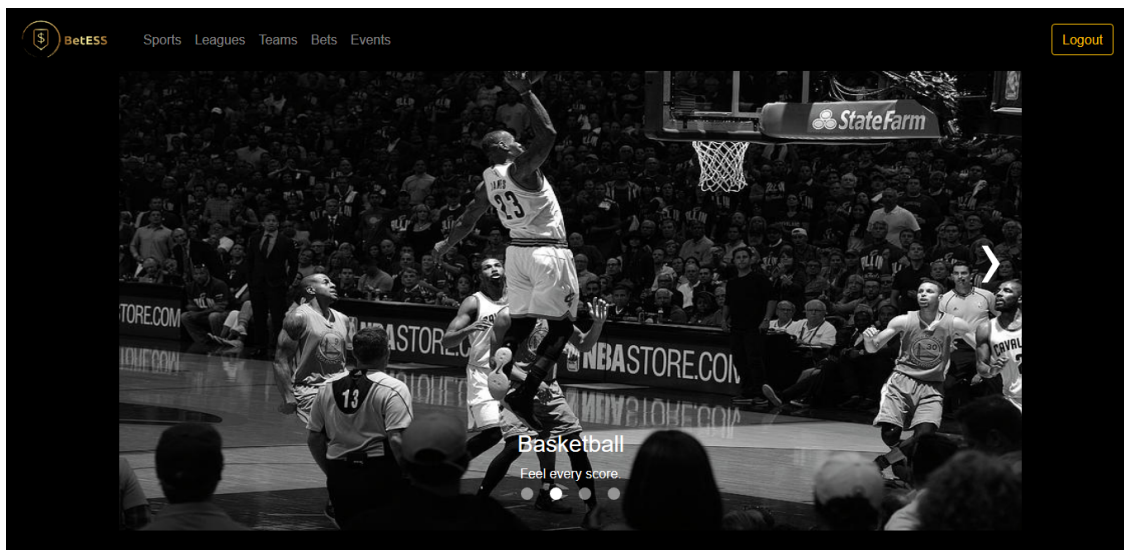


Figura 13: *Home* do administrador

Na página de desportos, o administrador do sistema poderá ver os desportos existentes e criar/remover um que deseje.

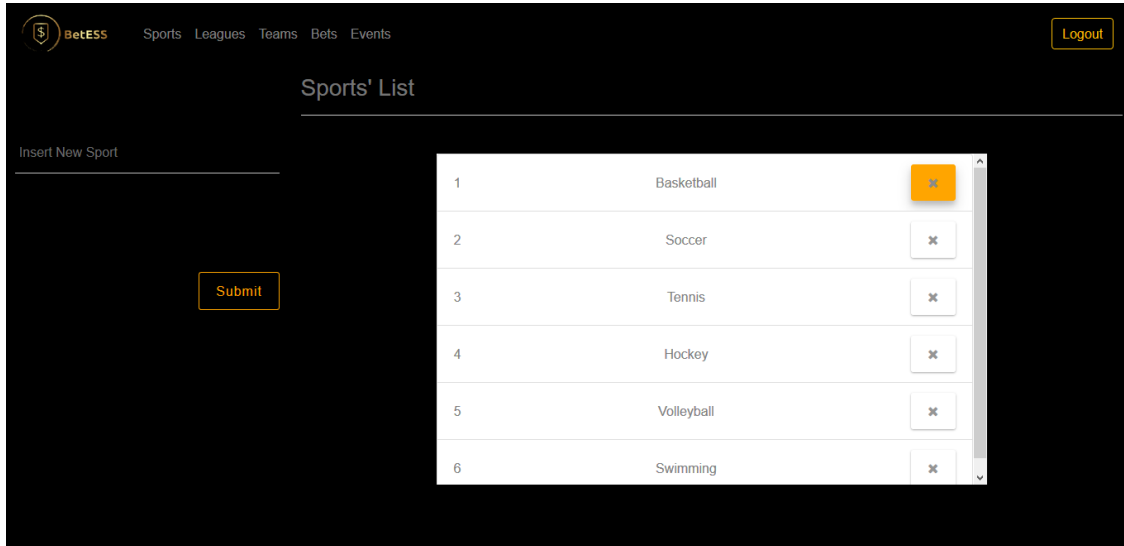


Figura 14: Criação, gestão e lista de desportos (Administrador)

Em relação às ligas, o *admin* poderá ver e adicionar alguma liga. A remoção das mesmas é, também, possível. Nesta página, seria preferível que desse para consultar as ligas por desporto, algo que poderia ser implementado numa próxima fase.

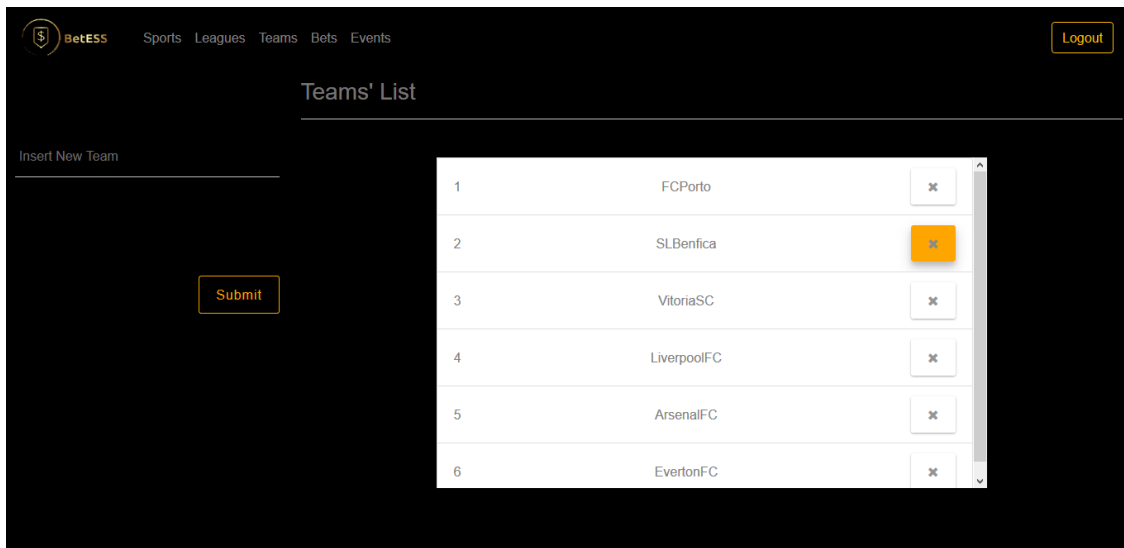


Figura 15: Criação, gestão e lista de ligas (Administrador)

Para a página de equipas, um administrador pode ver, criar e eliminar as mesmas. Também aqui seria desejável que se pudessem verificar equipas por desporto e por ligas, algo que poderia ser desenvolvido numa fase posterior.

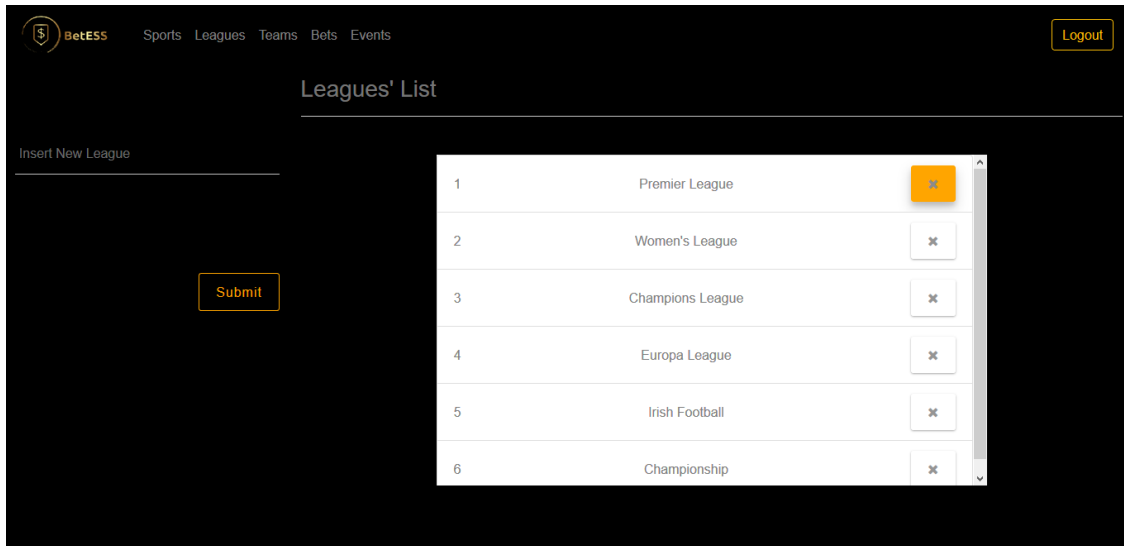


Figura 16: Criação, gestão e lista de equipas (Administrador)

Relativamente às apostas dos utilizador, o *admin* poderá, como forma de controlo, verificar todas as apostas atuais de todos os *users* do sistema. Não é possível eliminar apostas nem ver mais acerca do utilizador senão o seu *username*.

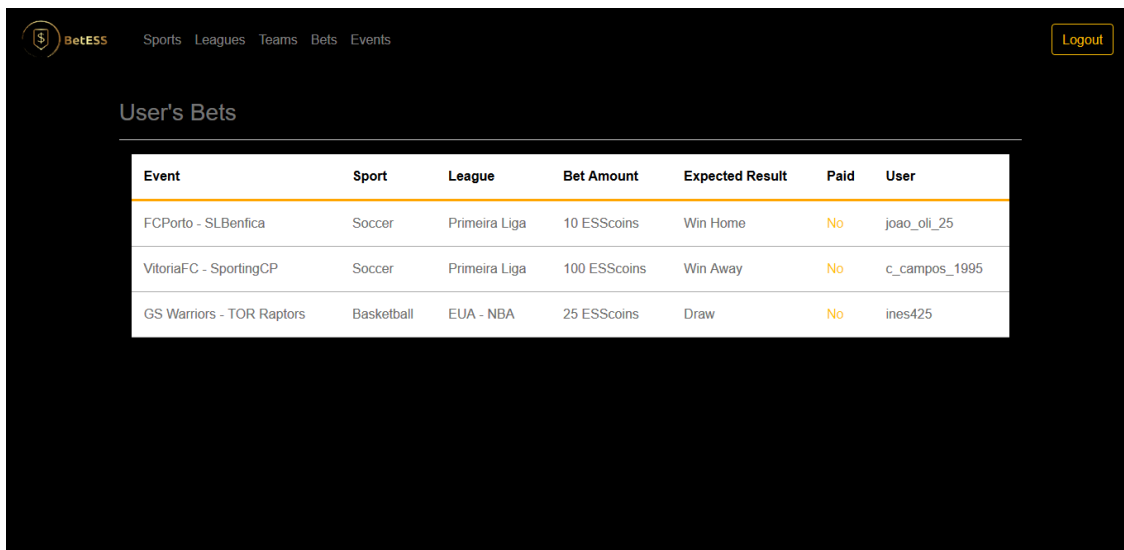


Figura 17: Lista de apostas de utilizadores (Administrador)

Quanto à página de eventos, provavelmente o maior foco da plataforma, um administrador poderá, em primeiro lugar, criar novos eventos, definindo os campos que se mostram na figura. Para além disso, poderá também gerir os eventos, isto é, fechar e eliminar. Quando fecha um evento, um *admin* define o resultado do evento, e todos os utilizadores recebem os seus ganhos, caso acertem no resultado do mesmo, juntamente com uma notificação. Caso o evento seja eliminado, todos os utilizadores com apostas no mesmo recebem o valor que apostaram.

BetESS Sports Leagues Teams Bets Events Logout

Create Event:

Select League Select Sport

Insert Odd Home

Insert Odd Away

Insert Odd Draw

Home Team Away Team

Premium?

Create Event

Events' Management

| | | | | | | | | | |
|---------------------------|-----------------------|-----|-----|-----|------|-----------|---------|--------|-------|
| GS Warriors - TOR Raptors | Basketball, EUA - NBA | 1.5 | 4.5 | 2.1 | Open | No Result | Premium | Delete | Close |
| FCPorto - Rio AveFC | Soccer, Primeira Liga | 0.5 | 0.3 | 4.1 | Open | No Result | Premium | Delete | Close |

Figura 18: Criação, verificação e gestão de eventos (Administrador)

4.4.4 Diretrizes de Acessibilidade de Conteúdo da Web (WCAG)

As Diretrizes de Acessibilidade para Conteúdo Web (WCAG) 2.0 abrangem um vasto conjunto de recomendações que têm como objetivo tornar o conteúdo Web mais acessível. O cumprimento destas diretrizes fará com que o conteúdo se torne acessível a um maior número de pessoas com incapacidades, incluindo cegueira e baixa visão, surdez e baixa audição, dificuldades de aprendizagem, limitações cognitivas, limitações de movimentos, incapacidade de fala, fotossensibilidade bem como as que tenham uma combinação destas limitações. Seguir estas diretrizes fará também com que o conteúdo Web se torne mais acessível e compreensível aos utilizadores em geral. Os critérios de sucesso das WCAG 2.0 são escritos sob a forma de declarações testáveis, que não dependem de uma tecnologia específica.

Para o presente projeto, era requerido, no mínimo, o **nível "A"** das diretrizes WCAG. Sendo o guia de diretrizes vasto, nem todas as normas se aplicarão dado que, por exemplo, o grupo não tem nenhum vídeo na interface. Por isso, as normas que não são aplicáveis ao projeto possuirão a sigla **"NA"** (quando avaliadas quanto a verificar ou não a diretriz), enquanto que, para as restantes, será feita uma avaliação conforme a condição de respeitar tal ponto.

É importante notar que a tabela resume os pontos por título de diretrizes e princípios, sendo que, na secção de referências, encontra-se o *website*[1] donde o grupo extraiu as normas referentes ao nível mínimo pedido, e onde se pode ler detalhadamente as mesmas.

| Princípio | Diretriz | Critério de sucesso nível A | Especificações do critério | Verifica? |
|------------------|---------------------------------------|--|---|------------|
| 1. Percetível | 1.1. Alternativas em Texto | 1.1.1. Conteúdo Não Textual | Controlos, Inserção de dados | SIM |
| | | | Conteúdo em Multimédia Dinâmica ou Temporal | NA |
| | | | Teste | NA |
| | 1.2. Média Dinâmica ou Contínua | 1.2.1. Conteúdo só de áudio e só de vídeo (pré-gravado) | Experiência Sensorial | NA |
| | | | CAPTCHA | NA |
| | | | Decoração, Formatação, Invisível | SIM |
| | | | Só áudio pré-gravado | NA |
| | | | Só vídeo pré-gravado: | NA |
| | 1.3. Adaptável | 1.2.2. Legendas (pré-gravadas) | - | NA |
| | | 1.2.3. Audiodescrição ou Alternativa em Multimédia (pré-gravada) | - | NA |
| | | 1.3.1. Informações e Relações | - | SIM |
| 2. Operável | 2.1. Acessível por Teclado | 2.1.1. Teclado | - | SIM |
| | | | - | SIM |
| | | | - | NA |
| | 2.2. Tempo Suficiente | 2.2.1. Tempo Ajustável | Desligar | NA |
| | | | Ajustar | NA |
| | | | Prolongar | NA |
| | | | Exceção por ser Tempo Real | NA |
| | | 2.2.2. Colocar em Pausa, Parar, Ocultar | Exceção por ser Essencial | NA |
| | | | Exceção de 20 Horas | NA |
| | | | Em movimento, em modo intermitente, em deslocamento | NA |
| | 2.3. Convulsões | 2.3.1. Três Flashes ou Abaixo do Limite | - | SIM |
| | | 2.4.1. Ignorar Blocos | - | NA |
| 3. Compreensível | 3.1. Legível | 3.1.1. Idioma da Página | - | SIM |
| | | | - | SIM |
| | | | - | SIM |
| | 3.2. Previsível | 3.2.1. Ao receber o Foco | - | SIM |
| | | 3.2.2. Ao entrar num campo de edição (input) | - | SIM |
| | | 3.3.1. Identificação de Erros | - | SIM |
| | 3.3. Assistência na Inserção de Dados | 3.3.2. Etiquetas ou Instruções | - | SIM |
| | | 4.1.1. Análise sintática (parsing) | - | SIM |
| | | 4.1.2. Nome, Função, Valor | - | SIM |
| | 4.1. Compatível | 4.1.1. Análise sintática (parsing) | - | SIM |
| | | 4.1.2. Nome, Função, Valor | - | SIM |

Tabela 1: Critérios de sucesso para o nível A das WCAG 2.0

Conclui-se, então, que a interface desenvolvida respeita as normas WCAG, no sentido em que cumpre todos os pontos que lhe são aplicáveis e tendo em conta o nível "A". No entanto, e antes do grupo ter contacto com as diretrizes, faltavam verificar vários pontos, sendo que foram necessárias algumas modificações à interface inicial desenvolvida de forma a encontrar-se em linha com as WCAG.

5 Conclusões e Sugestões

A resolução deste trabalho prático foi bastante importante e enriquecedora, pois permitiu aos membros do grupo perceber e interiorizar melhor os conceitos abordados nas aulas da Unidade Curricular de Engenharia Web.

Numa fase inicial, com a elaboração do projeto no *WebRatio* e apesar da dificuldade inicial, é notável a importância da abstração e modelação de um problema, por forma a criar soluções simples e melhorar as interações com o utilizador. A agilidade, rapidez e generalização da ferramenta e da linguagem IFML permitiram ao grupo avançar com muito mais rapidez para a segunda fase do projeto, uma vez que já estava definido, à partida, tudo o que seria necessário implementar, para além de que as *mockups* já estariam prontas.

No que toca a esta fase em concreto, é importante notar a elevada curva de aprendizagem. Foi o primeiro contacto do grupo com as tecnologias utilizadas, quer de *front-end*, *back-end*, servidor e do processo de autenticação. Para além deste impasse, o facto da equipa nunca ter implementado um sistema baseado numa arquitetura de microserviços atrasou ainda mais o desenvolvimento do projeto, e a união das tecnologias à existência de um reverse proxy não foi, de facto, tarefa simples. Esta arquitetura altera conceitos como autenticação e troca de mensagens entre componentes, algo que não constitui problema em arquiteturas monolíticas. Ainda assim, mesmo com a elevada quantidade de tempo despendido pelo grupo com questões do *reverse proxy* (fonte de bastantes problemas), e tendo encontrado alguns impasses nas *promises* do *JavaScript*, o grupo considera que o trabalho desenvolvido no *back-end* e *front-end* fluiu sem qualquer problema.

Em suma, é feita uma apreciação positiva relativamente ao trabalho realizado, visto que a implementação de todas as funcionalidades propostas foram conseguidas com sucesso. O grupo conseguiu tirar partido dos conhecimentos adquiridos neste projeto, sentido-se capaz de, num contexto futuro, aplicar os conceitos subjacentes de forma eficaz. É evidente que, num outro contexto (como um projeto de grandes dimensões), seria benéfico que fossem implementadas um maior conjunto de funcionalidades adicionais para uma melhor concretização do sistema.

Referências

- [1] Diretrizes WCAG, <https://www.w3.org/Translations/WCAG20-pt-PT/>
- [2] Betclic, <https://www.betclic.pt/>
- [3] Bet.pt, <https://www.bet.pt/apostas-desportivas/>