



Universidade do Minho

UMinho

Mestrado Engenharia Informática
Requisitos e Arquiteturas de Software
(2022/23)

RASBET

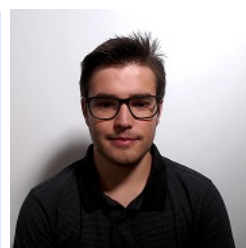
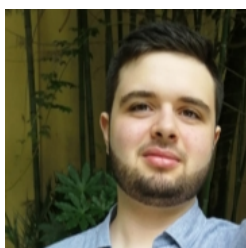
Grupo 7, PL5 - Entrega 2

PG50404 - Gonçalo Ferreira

PG50369 - Gonçalo Santos

PG50003 - Hugo Nogueira

PG50560 - Luis Faria



Braga, 4 de dezembro de 2022

Prefácio

Reflexões sobre o projeto desenvolvido:

Para esta segunda fase do desenvolvimento do projeto RASBet, a equipa retira um balanço positivo em relação ao trabalho apresentado: todos os requisitos funcionais e não funcionais (com prioridade igual ou superior a "Could") foram respeitados na solução final, todas as restrições impostas, cumpridas e uma grande variedade de diagramas foi apresentada, de forma a dar mais transparência à arquitetura e decisões que moldaram a aplicação RASBet. Ainda assim, o grupo sofreu algumas dificuldades durante o desenvolvimento, nomeadamente: no "refactoring" necessário da backend para acomodar uma arquitetura mais modular e de fácil manutenção (recorrendo a Typescript e padrões de desenho comuns) e na utilização e adoção de novas ferramentas para a frontend (anterior a este projeto, nenhum dos elementos da equipa tinha experiência no desenho e implementação de uma aplicação web com interface gráfica)

Contributo de cada elemento:

Distribuídos de forma semelhante à primeira fase, a equipa de implementação voltou a dividir-se pelas duas aplicações criadas, com a produção deste documento, dividida de forma igualitária pelos membros:

- **A equipa do backend**, constituída por Gonçalo Ferreira e Hugo Nogueira, que colaboraram nas reformas realizadas ao código, necessárias para atingir as metas de performance e manutenção definidas na primeira fase
- **A equipa do frontend**, constituída por Gonçalo Santos e Luis Faria, que continuaram no desenvolvimento da estrutura da interface gráfica da aplicação e da interação com os diferentes tipos de utilizadores

O grupo concordou em atribuir uma avaliação de 0 (contributo próximo da média) a todos os seus elementos.

Nota#1: As instruções de como executar a aplicação do projeto encontram-se no README que acompanha a implementação do projeto.

Nota#2: A primeira versão do relatório, manteve-se inalterada, dado que nenhum requisito não funcionais foi alterado, ou adicionado.

Resumo

Serve o presente relatório como forma de documentação da 2ª fase do projeto RASBet. Nesta fase de desenvolvimento a equipa concretizou a implementação dos requisitos definidos na fase anterior, especificando neste documento a arquitetura da solução criada.

Desta forma, o relatório começa com uma breve introdução que relembra os objetivos definidos na fase anterior, e estabelece os principais objetivos para esta fase. Segue-se um capítulo, que explicita as restrições que terão de ser respeitadas durante esta fase do desenvolvimento. No capítulo seguinte, são descritas as fronteiras deste projeto, identificando todos os atores ou sistemas que interagem de alguma forma com a RASBet. Na secção "Solução Arquitetural" a equipa apresenta os padrões arquiteturais seguidos, e as tecnologias utilizadas na solução (cada uma acompanhada de uma justificação para o seu uso). No capítulo seguinte, são demonstradas as vistas sobre o projeto, com recurso a diversos modelos UML, todos concisamente descritos e após isto são discutidos alguns dos padrões de desenho implementados na solução. Por fim, o último capítulo deste documento apresenta um breve "Manual do Utilizador" que demonstra a interface gráfica do sistema e as suas funcionalidades.

Conteúdo

| | |
|---|-----------|
| Prefácio | 1 |
| 1 Introdução e Objectivos | 3 |
| 1.1 Introdução | 3 |
| 1.2 Objectivos do desenvolvimento | 3 |
| 2 Restrições ao desenvolvimento | 4 |
| 2.1 Restrições Obrigatórias da Solução | 4 |
| 2.2 Restrições Prazo/Agendamento | 4 |
| 2.3 Restrições Orçamento | 4 |
| 3 Contexto e Escopo do projeto | 5 |
| 3.1 Desportos e Ligas disponíveis | 5 |
| 4 Solução Arquitetural | 7 |
| 4.1 Escolha da "stack"tecnológica | 7 |
| 5 Vistas sobre o projeto | 9 |
| 5.1 Building Block View | 9 |
| 5.1.1 Diagrama de Componentes | 9 |
| 5.1.2 Diagrama de Classes | 11 |
| 5.1.3 Diagrama de Entidade-Relacionamento | 12 |
| 5.2 Runtime View | 13 |
| 5.2.1 Registo de um Apostador | 13 |
| 5.2.2 Adicionar odds iniciais a um evento | 14 |
| 5.2.3 Fechar evento desportivo | 14 |
| 5.2.4 Realizar aposta simples | 15 |
| 5.2.5 Atualizar listagem de eventos desportivos | 15 |
| 5.3 Deployment View | 17 |
| 6 Exemplos de Padrões de Desenho explorados | 19 |
| 6.1 Padrão comportamental: <i>Observer</i> | 19 |
| 6.2 Padrão estrutural: <i>Facade</i> | 19 |
| 6.3 Padrão creacional: <i>Singleton</i> | 19 |
| 6.4 Padrão estrutural: <i>Adapter</i> | 20 |
| 7 Manual do Utilizador | 21 |

Lista de Figuras

| | | |
|------|--|----|
| 3.1 | Diagrama de contexto | 5 |
| 4.1 | <i>Tech stack</i> | 7 |
| 5.1 | Diagrama de componentes do sistema RASBet | 9 |
| 5.2 | Diagrama de classes do backend da RASBet | 11 |
| 5.3 | Diagrama de entidade-relacionamento da Base de Dados | 12 |
| 5.4 | Diagrama de sequência do Use Case Registo de Apostador | 14 |
| 5.5 | Diagrama de sequência do Use Case Adicionar Odds iniciais | 14 |
| 5.6 | Diagrama de sequência do Use Case Fechar Apostas sobre evento | 15 |
| 5.7 | Diagrama de sequência do Use Case Fechar Apostas sobre evento | 15 |
| 5.8 | Diagrama de sequência da atualização da listagem | 16 |
| 5.9 | Diagrama de sequência da atualização dos resultados dos eventos | 16 |
| 5.10 | Diagrama de sequência da atualização dos resultados dos eventos desportivos da 1ª Liga portuguesa | 17 |
| 5.11 | Estado atual do deployment da aplicação RASBet | 17 |
| 5.12 | Possível desenho futuro do deployment aplicação RASBet | 18 |
| 6.1 | Singleton design pattern | 20 |
| 7.1 | Página de Login | 21 |
| 7.2 | Página de Registo | 21 |
| 7.3 | Listagem de eventos de Futebol Portugueses para um apostador | 22 |
| 7.4 | Listagem de eventos de Futebol para um Administrador | 23 |
| 7.5 | Listagem de eventos de Basquetebol para um Especialista | 23 |
| 7.6 | Página de Histórico de Apostas | 24 |
| 7.7 | Página da Carteira | 24 |

1. Introdução e Objectivos

1.1 Introdução

Terminada a fase de levantamento e tratamento de requisitos, e perante uma primeira implementação funcional da aplicação RASBet, este projeto poderá agora avançar para a fase do desenvolvimento arquitetural da solução.

De relembrar que, na primeira fase, o sistema RASBet foi definido como uma aplicação web, de uso fácil e intuitivo, que permite aos seus utilizadores submeter apostas sobre um variado catálogo de eventos desportivos, de diferentes desportos, com um grande ênfase na sua expansibilidade para novos desportos, novos tipos de aposta e mais apostadores. Ficou também definido que este sistema será automaticamente atualizado (com recurso a APIs externas), necessitando apenas de dois perfis de utilizadores para moderação do sistema: os especialistas (responsáveis pela atribuição de odds iniciais a cada evento) e administradores (responsáveis pelo controlo de eventos e promoções).

A partir da implementação funcional base, a equipa de desenvolvimento terá agora de completar o produto especificado, implementando todos os requisitos definidos na solução arquitetural criada, ajustando-a se necessário.

1.2 Objectivos do desenvolvimento

Para esta fase surgem assim 5 objetivos para o desenvolvimento:

1. Completar a implementação de todos os requisitos funcionais, definidos com prioridade "Should".
2. Completar a implementação dos requisitos de usabilidade e aparência, que tornarão a aplicação mais acessível e apelativa para os utilizadores
3. Melhorar o desempenho e escalabilidade da aplicação permitindo, uma melhor experiência de utilização para ainda mais utilizadores
4. Cumprir a grande maioria dos requisitos não funcionais (ainda não explorados) relacionados com a segurança, manutenção, culturais, e de natureza legal.
5. Implementar os requisitos funcionais de menor prioridade

O cumprimento destes objetivos representará um sucesso para o desenvolvimento aplicação, que desta forma se tornará mais apelativa para os seus utilizadores, retendo mais apostadores, o que por sua vez, representará um aumento dos lucros para os instigadores do projeto.

2. Restrições ao desenvolvimento

Para esta fase as restrições ao desenvolvimento, são semelhantes às definidas na primeira fase do projeto, com a adição de algumas restrições, que o grupo considerou pertinentes.

2.1 Restrições Obrigatórias da Solução

- **Descrição:** O apostador poderá apostar em qualquer um dos eventos desportivos disponíveis para aposta
Justificação: Esta restrição é a funcionalidade base do sistema, e deve por isso ser considerada uma restrição obrigatória ao desenvolvimento
- **Descrição:** O apostador tem acesso a uma listagem de desportos, e após a seleção dessa modalidade, a uma listagem de eventos desportivos desse mesmo desporto.
Justificação: O apostador deverá ter acesso a todas as listagens de eventos desportivos de forma a escolher quais as apostas, que considera mais apelativas.
- **Descrição:** A aplicação tem de ser um aplicação *web*.
Justificação: A aplicação deverá poder ser usada em qualquer lugar, a partir do seu computador ou *smartphone*.
- **Descrição:** A aplicação deverá ser capaz de atualizar a listagem de eventos e o resultados dos mesmos, com base na informação de APIs externas ao sistema
Justificação: A introdução desta funcionalidade tornará o sistema mais dinâmico, reduzindo a necessidade de controlo humano.

2.2 Restrições Prazo/Agendamento

- **Descrição:** O presente documento e o código fonte da aplicação terão de se ser entregues até dia 05 de dezembro de 2022.
Justificação: Definição da data de entrega do projeto para avaliação do estado do projeto quanto à sua solução arquitetural, e aos avanços na implementação.

2.3 Restrições Orçamento

- **Descrição:** O orçamento total para o desenvolvimento do projeto é de 20 000€ (vinte mil euros), durante um período de 4 meses.
Justificação: A equipa responsável pelo desenvolvimento do projeto é constituída por quatro engenheiros de *software*. Para além de ter em conta os salários dos elementos, é preciso também a compra de um domínio, bem como a de servidores para alojar todos os dados da aplicação.

3. Contexto e Escopo do projeto

O Escopo do projeto encontra-se então na criação e desenvolvimento de uma plataforma de registo de apostas, que receberá os eventos desportivos e seus resultados de APIs externas, e apenas estará responsável pela lógica da distribuição de prémios e pela apresentação da interface gráfica que permitirá aos diferentes tipos de utilizadores interagir com o sistema. No diagrama de contexto da figura 3.1, é possível identificar os sistemas e utilizadores que interagem com o sistema:

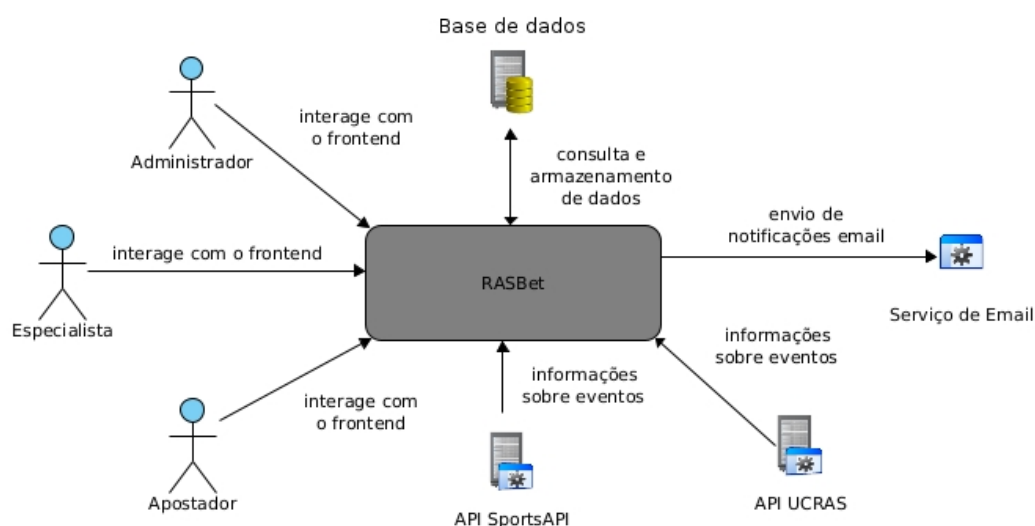


Figura 3.1: Diagrama de contexto

- **Apostadores, Especialistas e Administradores** - estes utilizadores representam a componente humana da interação com o sistema
- **Serviço de Email** - serviço implementado pelo módulo NodeMailer, é responsável pelo envio de emails de notificação aos utilizadores no final de apostas
- **Base de dados** - a ligação a um sistema de base de dados, é essencial para garantir a persistência de dados do sistema
- **API UCRAS** - servidor externo à aplicação que é responsável pela distribuição dos eventos desportivos da 1ª Liga Portuguesa de Futebol
- **API SportsAPI** - servidor externo à aplicação que é responsável pela distribuição de diversos eventos desportivos de vários desportos e de várias competições

3.1 Desportos e Ligas disponíveis

A aplicação será desenvolvida com a expansibilidade de desportos e competições em mente, mas na versão da aplicação que será disponibilizada em conjunção com este documento, terá

disponíveis os seguintes desportos e competições:

- **Futebol:**
 - Liga Portugal (Campeonato Português de Futebol masculino)
 - Premier League (Campeonato Inglês de Futebol masculino)
 - Ligue 1 (Campeonato Francês de Futebol masculino)
 - La Liga (Campeonato Espanhol de Futebol masculino)
 - UEFA Champions League
 - Liga Europa da UEFA
 - Copa do Mundo da FIFA
- **Basquetebol:** NBA (Liga de basquetebol dos EUA)
- **Fórmula Um:** Campeonato Mundial de Fórmula 1 da FIA

Todos os eventos desportivos com exceção da Liga Portugal, são retirados da API SportsAPI.

4. Solução Arquitetural

Dado que a aplicação RASBet se identifica como uma aplicação web, o seu desenho arquitetural será baseado na arquitetura mais comum deste tipo de aplicações: **Single Page Application**. Esta arquitetura é constituída por duas aplicações, o frontend (que lida com as interações com o utilizador e é responsável pela atualização dinâmica da página) e o backend (onde está alojada a grande maioria da lógica de negócio da aplicação e armazenamento de dados). A arquitetura SPA oferece melhor performance e escalabilidade, do que optar por apenas uma aplicação que envia páginas completamente novas a cada interação do utilizador. Para além disto, é também possível identificar o padrão arquitetural **Layered** (por camadas) onde podemos distinguir as camadas: *Presentation layer* (a frontend da aplicação), a *Business logic layer* (localizada principalmente no backend) e a *Data access layer* (implementada pela Base de dados e classe DBCommunication). Esta arquitetura promove a independência destas camadas, facilitando manutenções (a alteração de uma camada não implica a alteração das restantes) e a colaboração entre os participantes do projeto (que poderão trabalhar em paralelo nas diferentes camadas).

4.1 Escolha da "stack" tecnológica

Na figura 4.1, está representada a *stack* tecnológica definida para a solução arquitetural.

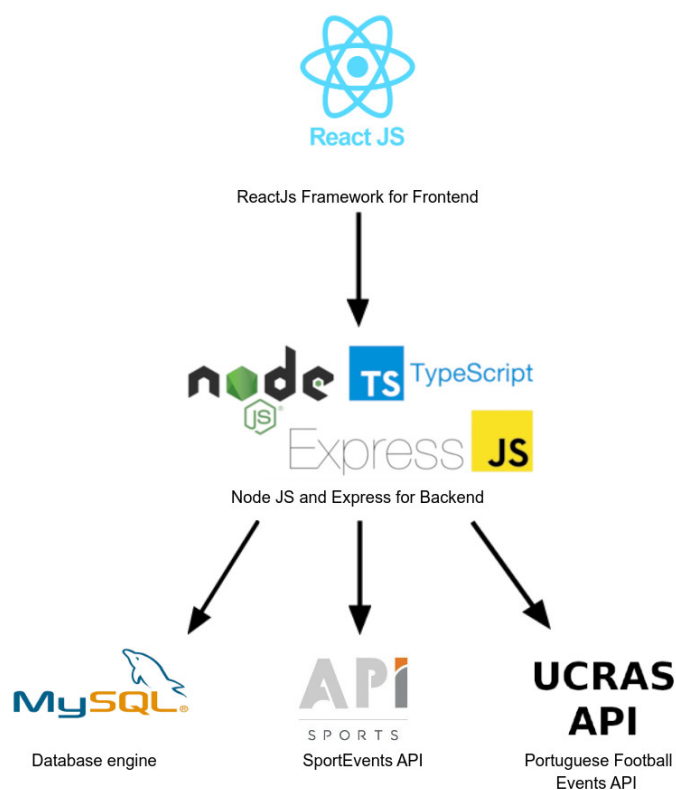


Figura 4.1: *Tech stack*

- Para o frontend da aplicação o grupo recorreu à framework **ReactJs**: construída sobre JavaScript, a framework, facilita o desenvolvimento de "single-page web applications", muito interativas e com compatibilidade para dispositivos móveis.
- Para o backend, a equipa escolheu como tecnologias a utilizar o conjunto de NodeJS, Typescript e ExpressJS, pelas seguintes razões:
 - **NodeJS** permite a execução local de código JavaScript, sem ser necessário um browser. Esta escolha permitiu à equipa manter a base de código toda em JavaScript, o que foi importante para permitir a fácil transição entre trabalho do backend e frontend. Em termos de performance, a natureza assíncrona desta plataforma permite uma resposta rápida e eficiente a pedidos concorrentes.
 - A biblioteca **ExpressJS** serve de middleware da aplicação, o que facilitou a implementação da comunicação entre o backend e o frontend. Esta plataforma permitiu também o desenvolvimento de "Server-side events", que dinamizaram o sistema de notificações.
 - **Typescript** é um "superset" de JavaScript, que adiciona tipos estáticos à linguagem. Por si só, JavaScript (uma linguagem desenhada para criar scripts) não oferece as funcionalidades de organização e estruturação de código, comuns entre linguagens orientadas a objetos, o que para projetos desta dimensão, tornariam a sua manutenção e suporte, substancialmente mais complexa. A transição para Typescript, permitiu à equipa redesenhar a arquitetura da aplicação, para um produto final, que considera mais modular e melhor estruturado
- A utilização do motor de base de dados relacional **MySQL** garantiu flexibilidade ao grupo para criar a sua estrutura de entidades personalizada para o problema (o que poderia não ser possível com a utilização de um ORM). A equipa considerou também a utilização de uma base de dados não-relacional, mas dado que esta aplicação apresenta tamanhos fixos de dados, e uma grande dependência das queries executadas sobre a mesma, a equipa decidiu manter a decisão de criar uma base de dados relacional.

5. Vistas sobre o projeto

Neste capítulo serão utilizados diagramas UML para demonstrar o funcionamento da aplicação três níveis diferentes: **Building Block View**, **Runtime View** e por fim **Deployment View**.

5.1 Building Block View

Aqui serão apresentados os diagramas UML relacionados com a decomposição estática do sistema e os diferentes níveis de abstração da estrutura do código fonte.

5.1.1 Diagrama de Componentes

Um diagrama de componentes é um diagrama UML de desenho estrutural de uma solução de software, que se foca nos componentes do sistema (componentes são pedaços de software modulares, bem encapsulados e facilmente substituíveis, cujo comportamento está bem definido). Este diagrama antecede a criação do diagrama de classes, e permite identificar o que será necessário construir no sistema. Na figura 5.1, foi desenhado o diagrama de componentes do sistema RASBet.

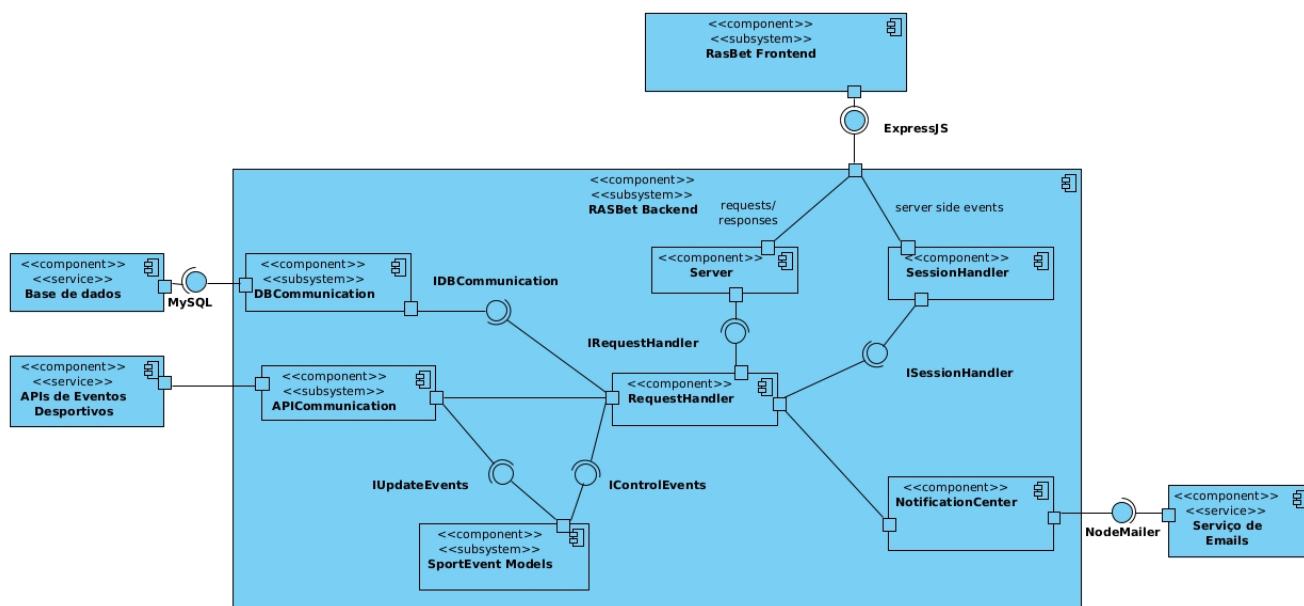


Figura 5.1: Diagrama de componentes do sistema RASBet

Neste diagrama é possível identificar os componentes:

- **RASBet Frontend**, que representa o frontend da aplicação; Este comunica com o backend recorrendo a pedidos HTTP interpretados pelo módulo ExpressJS, que serão depois tratados pelo Server e RequestHandler.

- **RASBet Backend**, responsável por englobar toda a estrutura interna do Backend
- **Server**, servindo como um dos extremos de comunicação com o frontend, este módulo será responsável pela distribuição dos diferentes pedidos, pelos diferentes "handlers" de pedidos contidos no módulo RequestHandler
- O **RequestHandler**, o componente central do sistema, sendo responsável pela resposta aos diferentes pedidos recebidos na backend. Relaciona as funcionalidades dos componentes DBCommunication, APICommunication, SportEvent Models, NotificationCenter e SessionHandler.
- **DBCommunication**, que se apresenta como o módulo do sistema, responsável pela comunicação com o sistema de base de dados, implementando os métodos da sua interface, que permitem a consulta, atualização e criação de dados na BD.
- O componente **APICommunication**, o ponto de contacto com as APIs externas que fornecem ao sistema, informações atualizadas acerca de eventos desportivos, que mais tarde serão utilizados para criar e resolver apostas.
- O subsistema **SportEvent Models**, o responsável por guardar a informação que provém das APIs (através da interface IUpdateEvents) e, alterar e consultar estes dados, para a implementação das funcionalidades da backend do sistema RASBet (através da interface IControlEvents)
- **NotificationCenter**, o módulo do sistema que trata do envio de notificações Email para os apostadores do sistema, quando existirem alterações sobre as suas apostas
- Por fim, o componente **SessionHandler** que armazena as sessões de cada utilizador, atribuindo a cada um, um token, que será utilizado para validar as suas operações. Implementa também o sistema de envio de notificações diretamente para o browser dos apostadores, através de Server-Side Events, que tornam o sistema mais dinâmico e responsivo a alterações importantes de estado.

5.1.2 Diagrama de Classes

Num nível de abstração mais próximo da implementação, torna-se importante o desenho de um diagrama de classes, que demonstra todas as classes e principais métodos que vão atuar no funcionamento da aplicação. Na figura 5.2 encontra-se o diagrama de classes do backend da solução (já que o frontend da aplicação não segue o paradigma orientado a objetos, e não possui uma organização inerente).

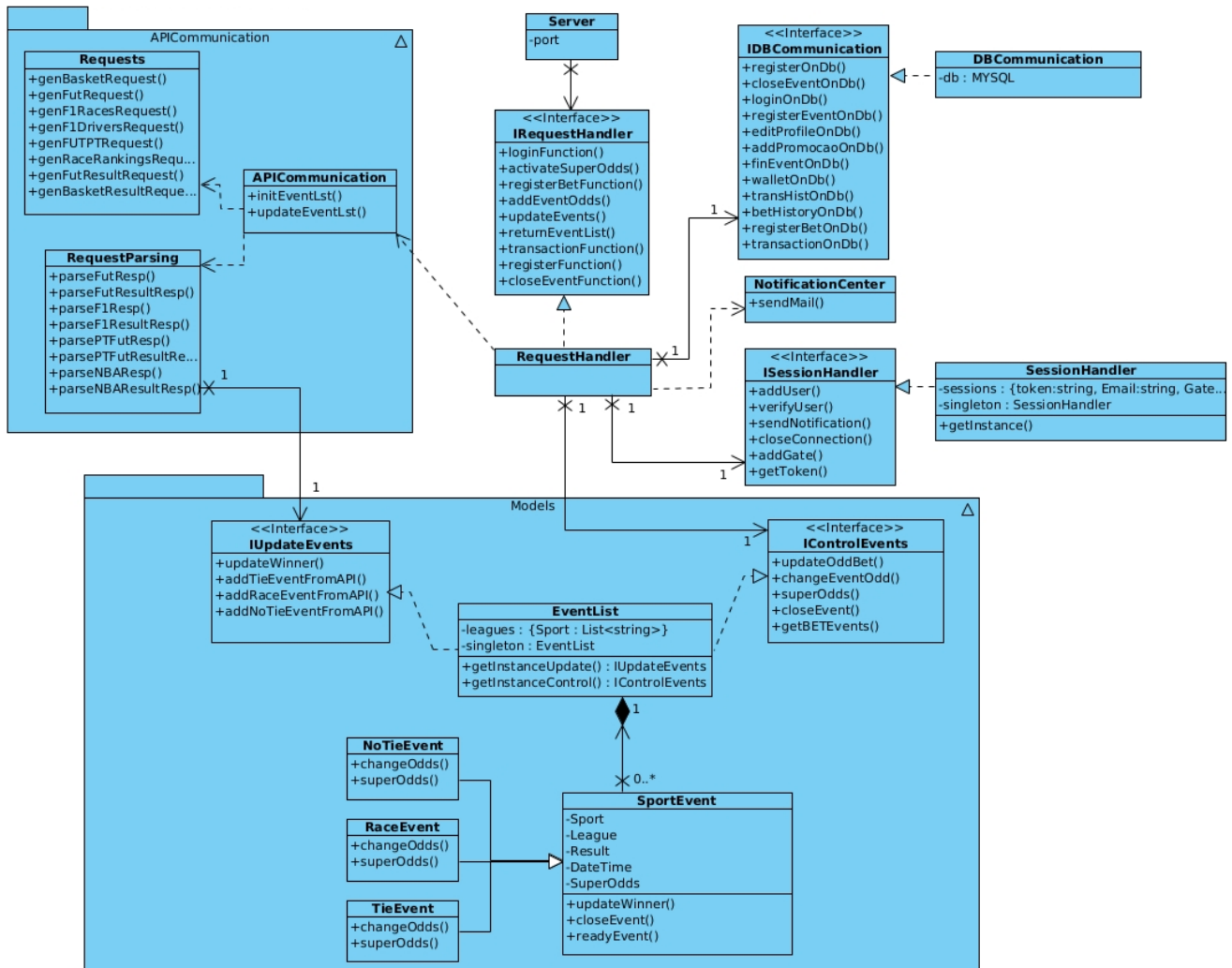


Figura 5.2: Diagrama de classes do backend da RASBet

Dado que grande parte da descrição do funcionamento destas classes já foi desenvolvido na descrição do diagrama de componentes (e de forma a evitar redundância), apenas serão descritos alguns dos aspetos mais pertinentes deste desenho:

- Como referido anteriormente a classe RequestHandler é o ponto central da aplicação; aqui são reunidas as funcionalidades de todos os módulos presentes, de forma a dar resposta aos diferentes pedidos provenientes do frontend.
- O módulo API Communication, implementa 3 classes diferentes: o APICommunication, responsável pelo envio dos pedidos à API, a classe Requests, que contém a definição dos pedidos para cada API, e por fim o RequestParsing, que é responsável por interpretar todas as respostas provenientes da API, e com estas informações, atualizar os dados dos eventos.

- O package Models, implementa todas as classes que são responsáveis pelo armazenamento das informações acerca dos eventos desportivos. A classe EventList, guarda todos os eventos e é responsável pela atualização e consulta de informações sobre os mesmos. Esta classe implementa também duas interfaces: a interface IUpdateEvents, que implementa os métodos que criam eventos e atualizam os seus resultados e a interface IControlEvents, que implementa os métodos que serão utilizados pelo RequestHandler para o desenvolvimento das funcionalidades do backend
- Existem 3 tipos de eventos possíveis: NoTieEvent, eventos cujo o resultado entre duas equipa não pode acabar em empate; TieEvent, eventos cujo resultado entre duas equipa poderá acabar em empate; e por fim RaceEvent, onde existe mais do que um participante e apenas um poderá ser vencedor. Todas estas classes, são subclasses da classe SportEvent que define os atributos e métodos comuns.

5.1.3 Diagrama de Entidade-Relacionamento

Ainda no contexto da Vista sobre os componentes estruturais da solução, devemos também abordar a arquitetura do nosso esquema de Base de dados. Assim na figura 5.3 podemos encontrar o diagrama de entidade-relacionamento (ERD) e, por baixo, uma descrição concisa de cada tabela, que constitui a base de dados do sistema RASBet.

Nota: Devido à alta volatilidade e escala dos dados dos eventos e seus participantes, estas informações foram reduzidas na base de dados, ao seu estado mais básico, representado na tabela Evento; todos os dados relativos a estes eventos são retirados da API e guardados em memória durante a execução da aplicação, sem que exista persistência destes dados em caso de falha, para além das informações básicas apresentadas.

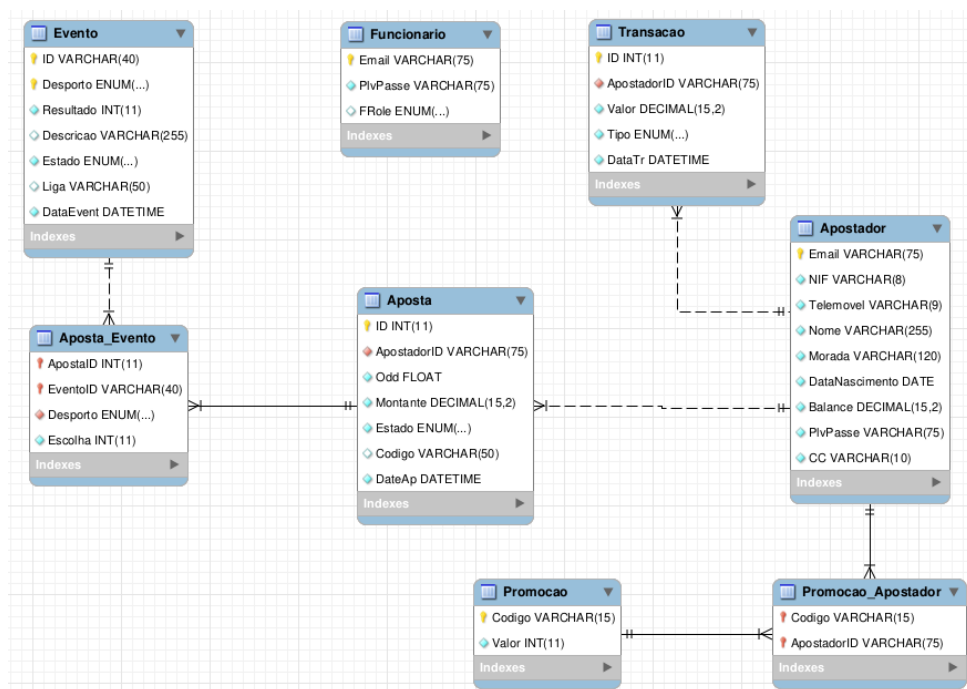


Figura 5.3: Diagrama de entidade-relacionamento da Base de Dados

- **Evento** - tabela que contém a informação base de um evento desportivo sobre o qual foram realizadas apostas. Estes eventos são registados, quando a primeira aposta sobre o mesmo é registada e serão atualizados no final do evento com o resultado do mesmo.
- **Aposta_Evento** - tabela que expressa a associação N:M (muitos para muitos) entre as

tabelas Evento e Aposta. Esta ligação exhibe a possibilidade da criação de apostas sobre diferentes eventos, e de uma única aposta sobre vários eventos (apostas múltiplas).

- **Aposta** - tabela que contém as informações necessárias para o registo de uma aposta. Relaciona as Apostas, com o(s) Evento(s) onde foi realizada, o Apostador que a criou, e o código utilizado
- **Transacao** - tabela que regista as transações nas carteiras dos apostadores. Cada transação tem definido um tipo que pode ser: "Aposta_Ganha", "Aposta", "Levantamento_Conta", "Deposito_Conta" e "Refund".
- **Apostador** - tabela que guarda as informações de contacto e de identificação dos apostadores da aplicação
- **Promocao** - tabela que armazena os códigos promocionais definidos pelos administradores, que poderão ser utilizados uma vez por cada apostador
- **Promocao_Apostador** - tabela que expressa a associação N:M (muitos para muitos) entre as tabelas Promoção e Apostador, que servirá como forma de regular o uso de cada promoção, uma vez por apostador.
- **Funcionario** - a tabela Funcionário guarda a informação de login e o papel de cada funcionário. Dado que o sistema não necessita de tantos dados acerca destes atores, a equipa decidiu separá-los dos apostadores. Esta decisão obriga o processo de login e registo a consultar ambas as tabelas.

5.2 Runtime View

A vista sobre o "Runtime" permite demonstrar o comportamento do sistema durante a execução de use-cases importantes, interações com utilizadores e sistemas externos e os erros e exceções que poderão surgir durante funcionamento da aplicação. Os diagramas mais apropriados para demonstrar este tipo de vista sobre o projeto são os diagramas de sequência, que permitem visualizar as diferentes mensagens que terão de ser passadas entre as interfaces de um sistema até que a interação seja dada como concluída. Nas próximas secções serão apresentados e descritos, os diagramas de sequência de alguns dos use-cases, que a equipa considerou como sendo de maior relevância.

5.2.1 Registo de um Apostador

De forma a registar um Apostador, este terá de preencher a página de registo, com as suas informações, que serão posteriormente enviadas pela Frontend para o servidor, que por sua vez comunica com a base de dados de forma a saber se os dados fornecidos podem ser registados. Caso não haja inconformidades a conta do apostador é registada. Por fim o apostador é informado se o registo teve sucesso, ou em caso contrário, qual o problema dos dados submetidos.

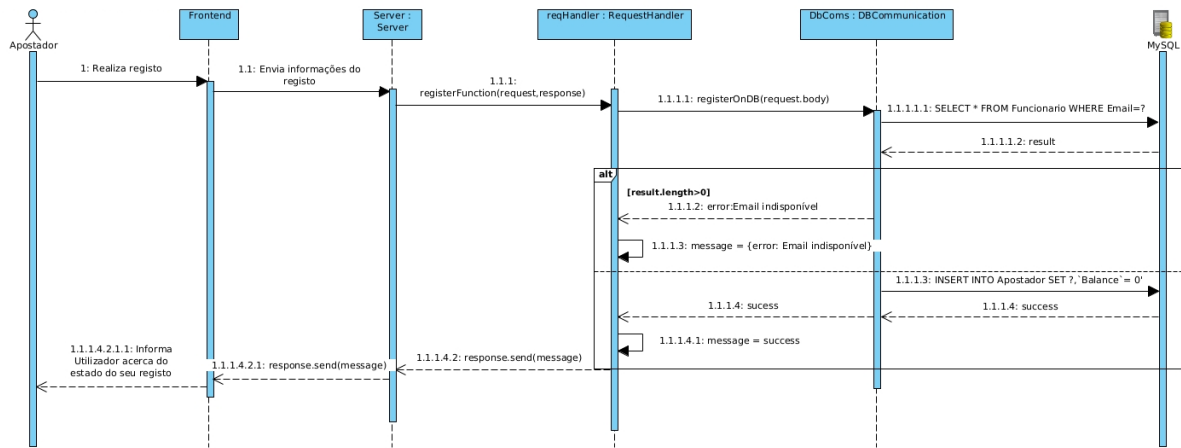


Figura 5.4: Diagrama de sequência do Use Case Registo de Apostador

5.2.2 Adicionar odds iniciais a um evento

A fim de adicionar odds iniciais a um evento, um especialista deve seleccionar o evento desportivo que pretende, e especificar as odds dos possíveis resultados. De seguida, estas são enviadas ao servidor, no qual é feita a verificação das permissões do utilizador que enviou este pedido. Depois deste passo, as odds do dito evento desportivo são aplicadas, e por fim este é indicado como pronto para apostas.

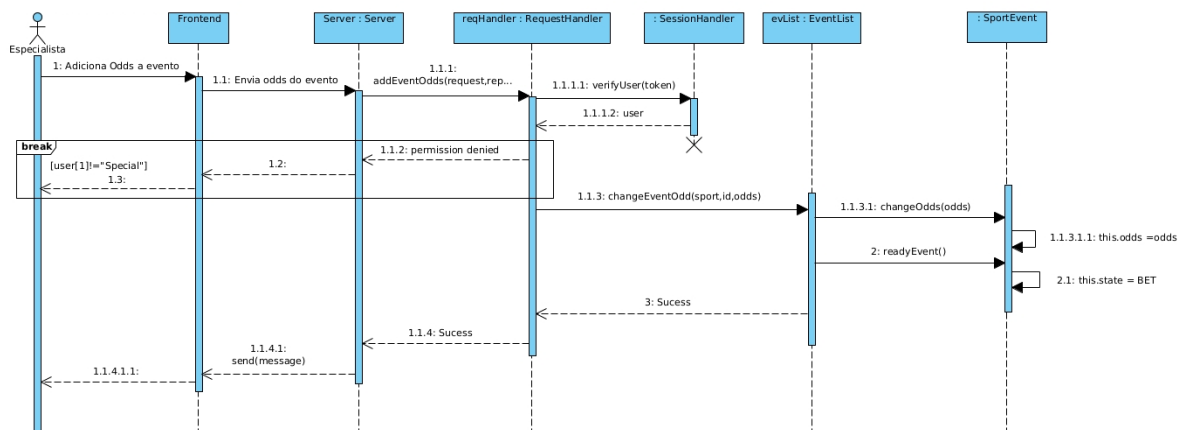


Figura 5.5: Diagrama de sequência do Use Case Adicionar Odds iniciais

5.2.3 Fechar evento desportivo

Para fechar um evento desportivo, um administrador começa por seleccionar a opção Fechar Evento num determinado evento desportivo. De seguida a frontend informa o servidor da intenção de fecho do dito evento. Este por sua vez verifica se o pedido foi feito por um administrador e só depois é que começa as operações. Estas consistem em alterar o estado do evento, e de todas as apostas no qual este estava presente. Feito isto, todos os apostador afetados são reembolsados pelo valor total da aposta, alterando assim o estado das suas carteiras (evento do lado do servidor, comunicado com os apostadores com sessão iniciada). Estes apostadores são também avisados por email do ocorrido.

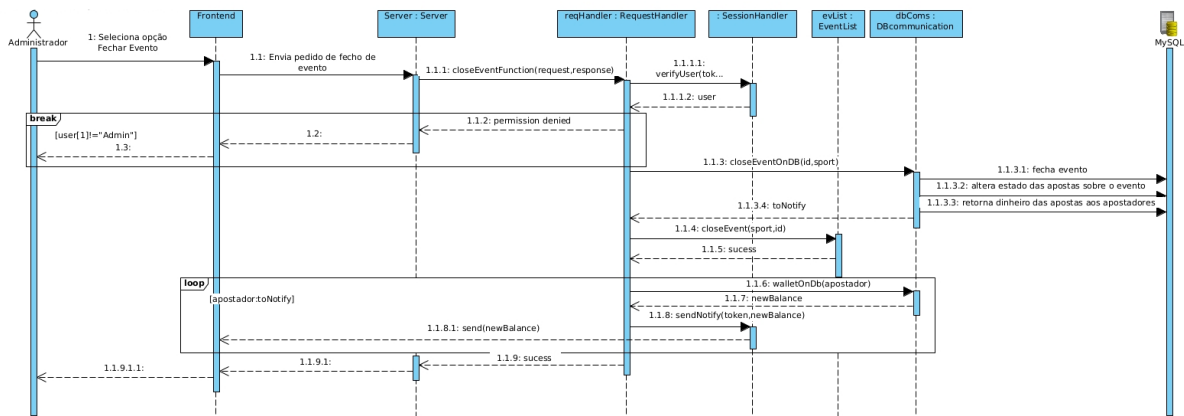


Figura 5.6: Diagrama de sequência do Use Case Fechar Apostas sobre evento

5.2.4 Realizar aposta simples

O processo de realizar uma aposta simples inicia quando um apostador preenche um boletim de apostas e tenta submetê-lo. As informações submetidas são enviadas para o servidor, no qual são realizados vários passos para o registo da aposta: primeiro é confirmado se quem submeteu o boletim tem autorização para o fazer; de seguida, caso um código promocional esteja a ser usado, é verificado se este código foi usado previamente (caso tenha sido a aposta é recusada); depois é introduzido o evento desportivo correspondente na base de dados (caso ainda não esteja lá presente e o seu estado seja válido) . Após isto a transação do montante apostado é realizada, o código, caso tenha sido usado, é associado à conta do apostador e por fim a aposta é adicionada.

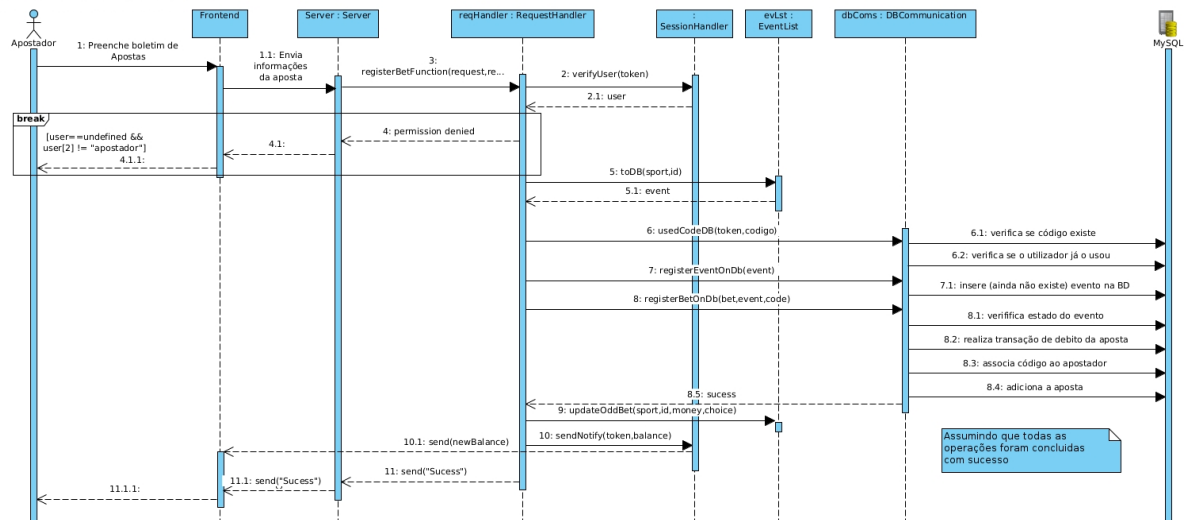


Figura 5.7: Diagrama de sequência do Use Case Fechar Apostas sobre evento

5.2.5 Atualizar listagem de eventos desportivos

Um administrador, a qualquer momento tem a opção de atualizar a listagem de eventos desportivos. Quando este a ativa, um pedido de atualização, é enviado para o servidor, onde é feita a confirmação de que quem o enviou tem permissões para tal. Caso isto se confirme, a função para atualizar os resultados (updateResults() explicada posteriormente) entra em ação. Depois disto, são retirados da API, novos eventos e por fim é devolvida uma mensagem de sucesso.

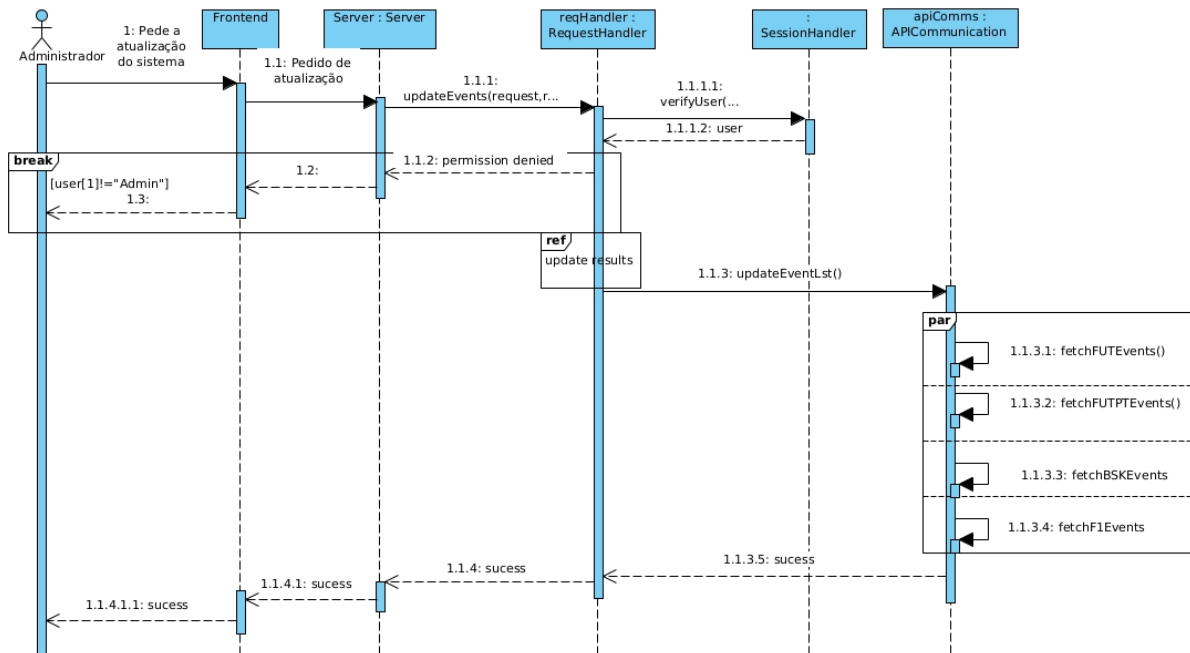


Figura 5.8: Diagrama de sequência da atualização da listagem

A função `updateResults()` mencionada acima, funciona da seguinte forma: para cada tipo de desporto são verificados os eventos que já ocorreram, e o seu resultado é atualizado com recurso à API (método `updateResults`); de seguida, cada um destes eventos desportivos é dado como finalizado na base de dados, atualizando as apostas onde estes estão presentes; por fim todos os respetivos apostadores serão notificados do resultado das suas apostas.

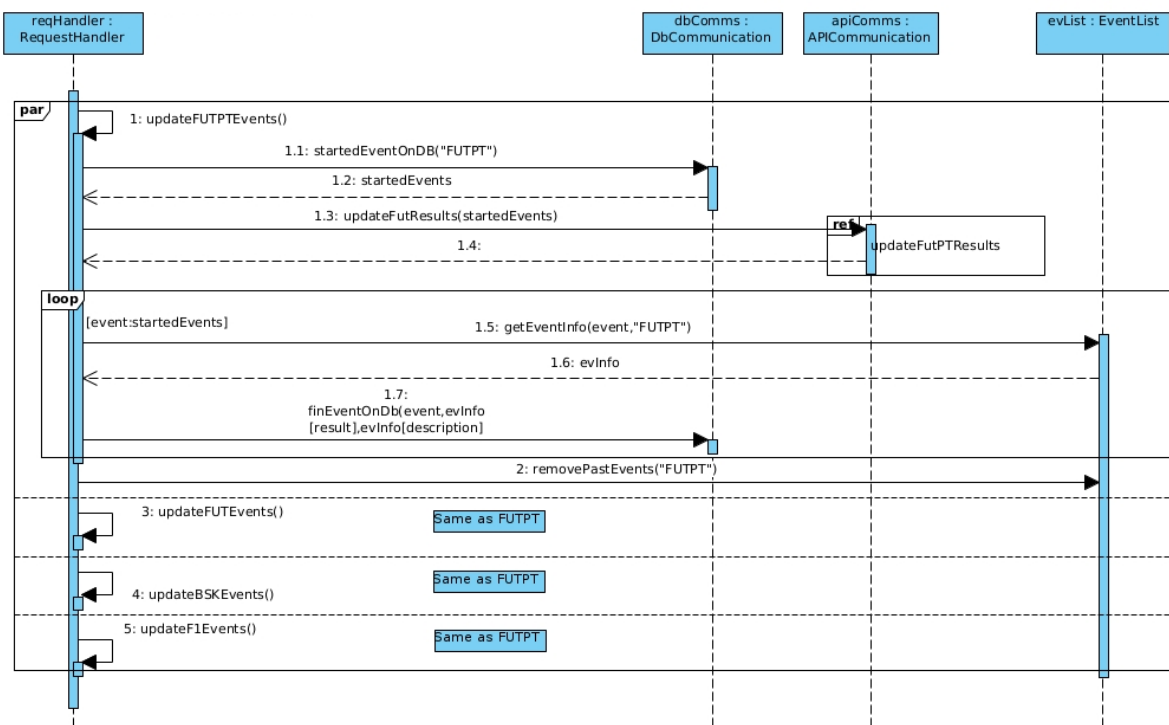


Figura 5.9: Diagrama de sequência da atualização dos resultados dos eventos

Para atualizar os resultados dos eventos desportivos, é feito um pedido à API correspondente. A

partir da resposta a este pedido, o módulo ResponseParsing analisa todos os eventos, escolhe o vencedor das partidas que já decorreram, e atualiza a EventList com o método "updateWinner".

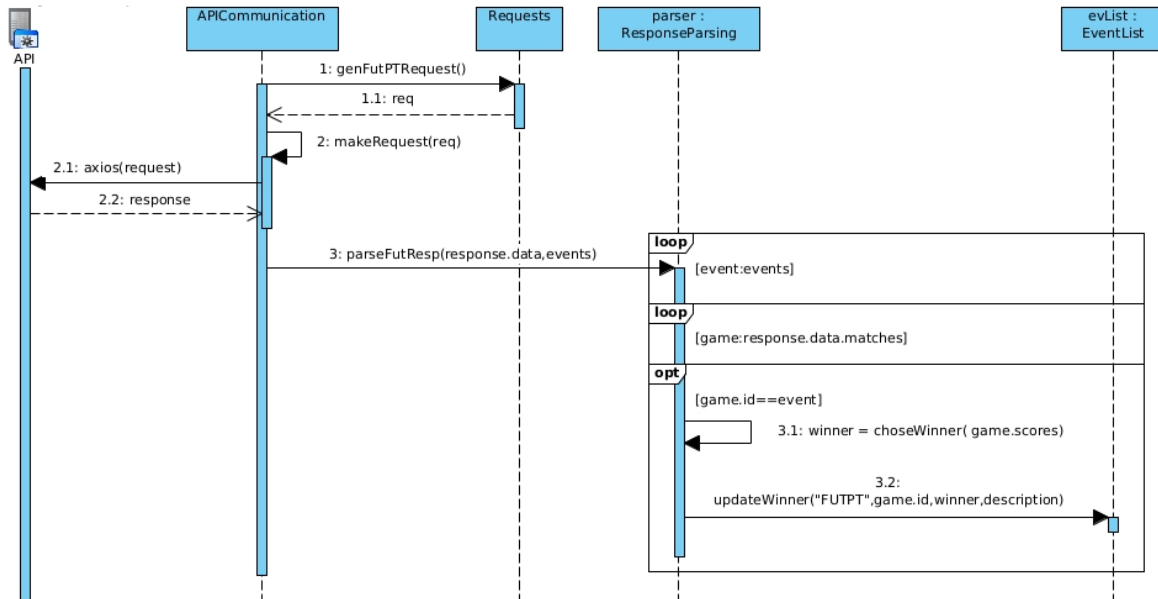


Figura 5.10: Diagrama de sequência da atualização dos resultados dos eventos desportivos da 1ª Liga portuguesa

5.3 Deployment View

Quanto à vista sobre o deployment esta não será muito complexa, dado que nesta fase, o projeto ainda estará contido dentro da mesma máquina. No entanto, no futuro, esta aplicação (da forma como foi desenhada) permitirá a criação de um serviço distribuído que fornecerá aos seus utilizadores, a aplicação RASBet, onde quer que estejam, e independentemente do número de utilizadores concorrentes, sem que estes fatores prejudiquem a performance da aplicação. Na figura 5.11 está apresentado um diagrama de deployment que apresenta o estado atual da aplicação; já na figura 5.12, está apresentado uma possível implementação distribuída do deployment da aplicação, que recorre a servidores de balanceamento de carga e a mais do que uma instância da base de dados para promover o escalonamento da aplicação quanto ao número de utilizadores.

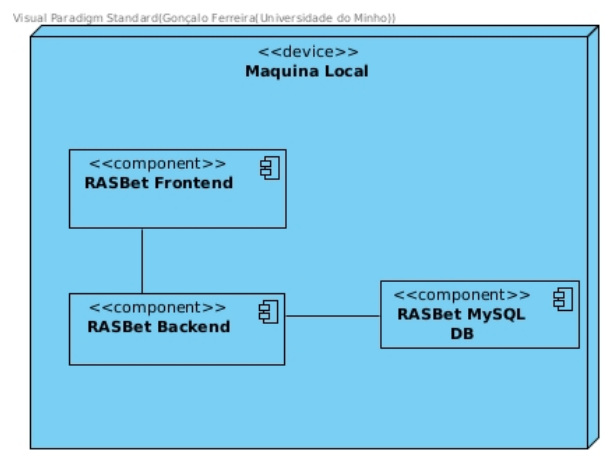


Figura 5.11: Estado atual do deployment da aplicação RASBet

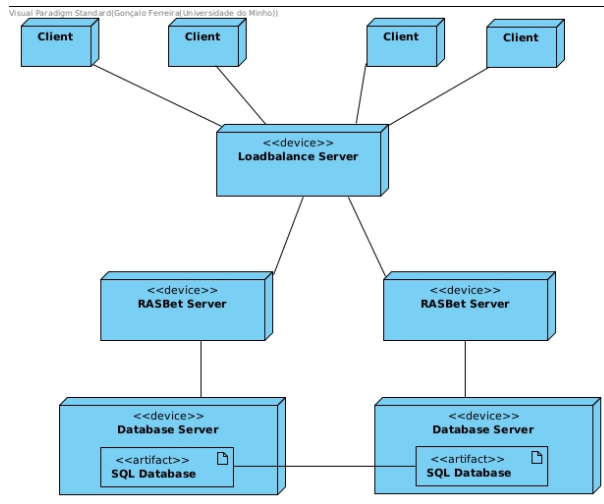


Figura 5.12: Possível desenho futuro do deployment aplicação RASBet

6. Exemplos de Padrões de Desenho explorados

Padrões de desenho (aka design patterns) são soluções típicas para problemas comuns no desenho e arquitetura de software. Cada padrão deve ser encarado como uma "blueprint" de boas práticas, que deve ser adaptado para cada problema da solução.

Nesta secção documentaremos alguns problemas que foram resolvidos com recurso a estes padrões de desenho.

6.1 Padrão comportamental: *Observer*

O padrão Observer foi implementado na nossa aplicação, no sistema de notificações sobre transações na carteira.

Para melhor explicar a implementação deste padrão temos primeiro de explicar o funcionamento da classe Session Handler.

Sempre que um utilizador realiza o login na aplicação, a sua sessão é guardada no backend, e uma ligação a partir do backend para o seu browser é criada (será sobre esta ligação que o backend e a classe Session Handler notificarão o utilizador). Quando esta ligação é quebrada, o utilizador passa a estar "logged out".

Quando for registada uma transação sobre a carteira, o backend é responsável por enviar uma notificação a cada um dos utilizadores "logged in", que poderão visualizar uma atualização ao seu balanço sem ser necessária uma atualização da página.

Aplicando a terminologia do padrão Observer a este caso, podemos considerar as sessões dos utilizadores como os observers, e o Session Handler como o subject que notifica a sua lista de observers.

6.2 Padrão estrutural: *Facade*

O padrão estrutural Facade é caracterizado pela necessidade de obter uma interface limitada mas simples, para um subsistema muito complexo. Na solução arquitetural do backend da aplicação, podemos verificar a implementação deste padrão de desenho na interface IRequestHandler: esta permite isolar a complexidade do código presente no RequestHandler, da classe Server, que passa assim a redirecionar todos os pedidos que recebe, para um dos métodos desta interface, abstraindo-se da sua implementação.

6.3 Padrão creacional: *Singleton*

O padrão creacional Singleton é muito útil quando se pretende que exista apenas uma instância de uma classe, o que é o caso, nesta aplicação, das classes EventList e SessionHandler. Estas duas classes, possuem o método getInstance, que ao ser chamado, verifica se já existe uma instância da classe: caso exista, devolve-a; caso contrário cria e retorna uma nova instância da classe. Na figura 6.1 está demonstrado o funcionamento deste padrão.

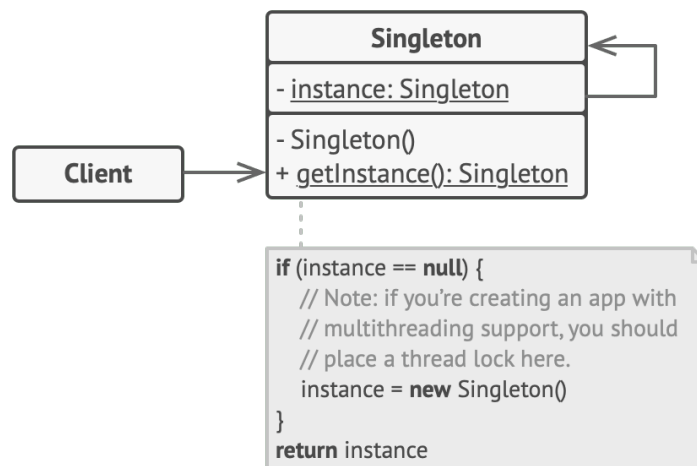


Figura 6.1: Singleton design pattern

A utilização deste padrão permitiu utilizar a `EventList` tanto na classe `APICommunication`, onde a mesma é preenchida com eventos, e na classe `RequestHandler` onde os eventos são consultados e alterados.

6.4 Padrão estrutural: *Adapter*

De forma a atualizar dinamicamente os diversos eventos desportivos, o grupo precisou de consultar APIs externas, para obter estas informações. No entanto, quando é realizado um pedido à API, a resposta geralmente está contida num ficheiro JSON, que precisa de ser tratado de forma a retirar os dados relevantes para a aplicação, e a transformá-los nos objetos internos à backend.

O padrão `Adapter`, é um padrão de desenho estrutural, que permite a objetos com interfaces diferentes colaborar e interagir. No caso do problema anteriormente descrito, este padrão foi a chave para resolução do problema: aplicado à nossa solução, o padrão `Adapter` poderá ser visto na colaboração existente entre as classes `RequestParsing` e `IUpdateEvents`, onde a primeira é responsável por tratar e moldar a informação que é proveniente da API, e a segunda está encarregue de através destes dados, gerar Eventos ou atualizar os mesmos, criando assim um processo de tradução entre a classe JSON proveniente das APIs e os diferentes objetos que representam eventos desportivos.

7. Manual do Utilizador

Quando o utilizador entra na aplicação, começa na página de *login*.

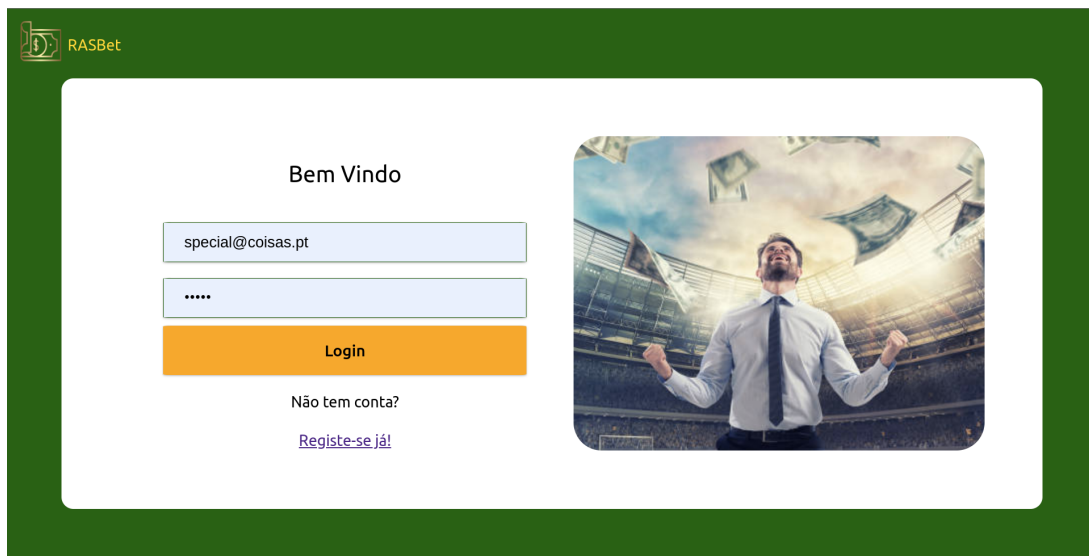


Figura 7.1: Página de Login

Nessa página, o utilizador vai introduzir as credencias da sua conta, ou se não tiver, seleccionar a opção **Registe-se já!** que vai direccionar o utilizador para a página de registo.

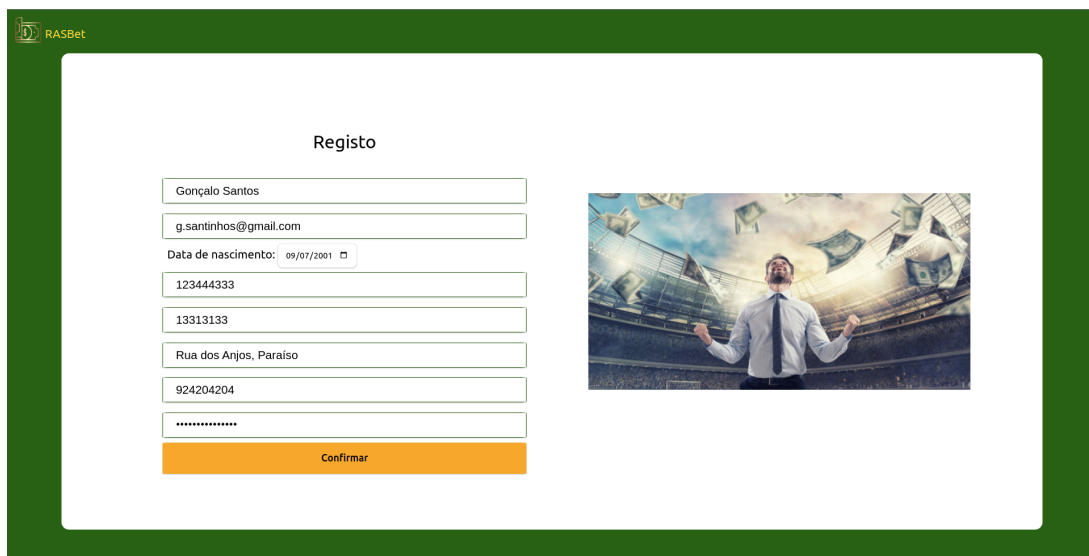


Figura 7.2: Página de Registo

No registo, o utilizador terá de preencher todos os campos com os seus dados e escolher uma palavra-passe. A palavra-passe tem de possuir, no mínimo, 8 carateres, das quais pelo menos

uma letra, terá de ser maiúscula, um carácter especial e um número. Ao seleccionar o logótipo, o utilizador volta à página de *login*.

Depois de logado, ou acabado o registo, é carregada a página principal, que é a página de apostas do futebol português. O formato desta página vai depender do tipo do utilizador e do tamanho da página aberta. Também é para esta página que o apostador volta se clicar no logótipo em qualquer página depois de logado.

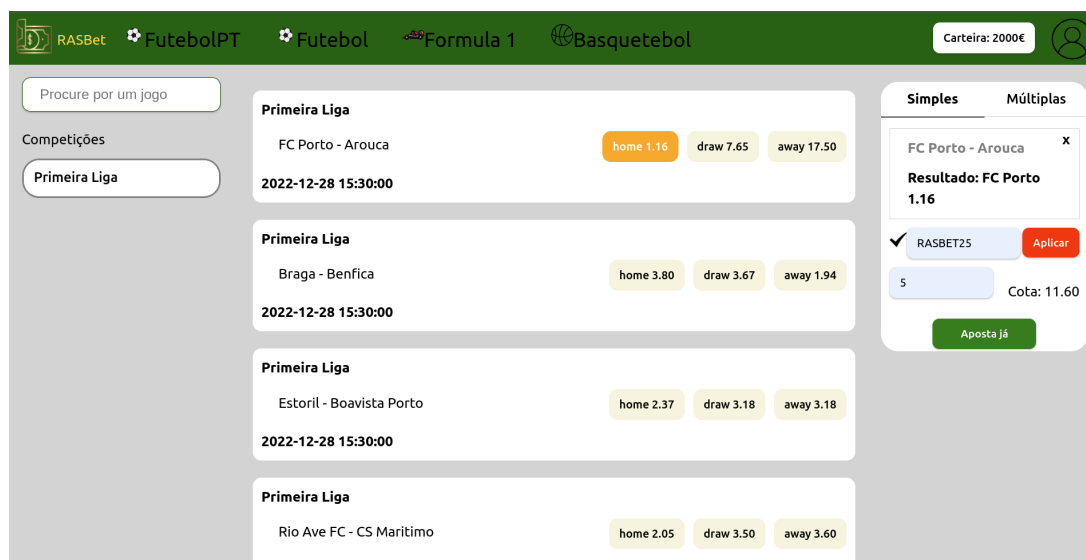


Figura 7.3: Listagem de eventos de Futebol Portugueses para um apostador

Nessa página, o apostador vai realizar o objetivo principal da aplicação, fazer apostas. No cabeçalho da página, ao seleccionar um dos desportos, a listagem de eventos muda para as do desporto seleccionado. No canto direito está presente o botão que permite visitar a carteira onde se pode ver o montante que o apostador possui, e ao pairar o cursor sobre a imagem de perfil, surgirá uma lista de opções para visitar outras páginas, sendo estas o perfil, o histórico de apostas, o histórico de transações, e a opção para voltar à página do *login*, ou seja, um "*log out*".

A parte principal desta página, está dividida em três colunas. A coluna de filtragem, que fica à esquerda, permite procurar por uma equipa em específico ao escrever na barra de procura, e filtrar os jogos pelo nome da competição, seleccionando a competição na lista, sendo possível filtrar por diferentes competições ao mesmo tempo.

Na coluna central, a listagem de eventos, onde estes são apresentados num *card format*, onde à esquerda é possível ver a informação do evento, e nos botões à direita, é possível começar uma aposta ao seleccionar o resultado do evento. Ao seleccionar o resultado, o evento irá aparecer na caixa da coluna direita, onde para concluir uma aposta, o apostador terá que introduzir o montante a apostar e poderá introduzir um código promocional que adicionará um certo montante à aposta. Depois de preenchido, conclui-se a aposta ao seleccionar o botão **Aposta**. É possível mudar o tipo de aposta nos botões **Múltiplas** e **Simples**, na coluna direita.

O administrador, nesta página, terá a coluna dos filtros como o apostador, mas na listagem de eventos, em vez das opções de resultado do evento terá a opção de cancelamento do evento, uma área para introduzir a *SuperOdd* do evento e um botão para a submeter. Na coluna da direita, o administrador pode criar e eliminar códigos promocionais. Para os criar, terá de introduzir o código e o valor associado à promoção, e para os eliminar, selecciona o "x" no canto superior direito da caixa do código promocional correspondente.

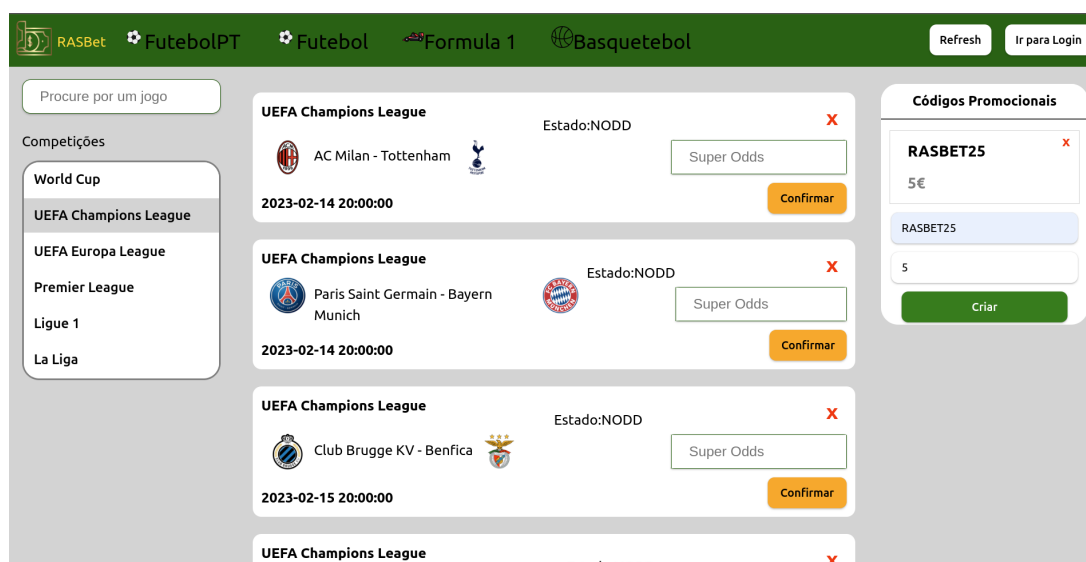


Figura 7.4: Listagem de eventos de Futebol para um Administrador

Para o especialista, a coluna dos filtros permanece igual, e nas listagens de eventos, para além das suas informações, são também apresentadas as Odds sugeridas pela API, podendo submetê-las selecionando o botão **Submeter** com as áreas de preenchimento vazias, ou preencher as mesmas, e submeter as Odds que definiu.

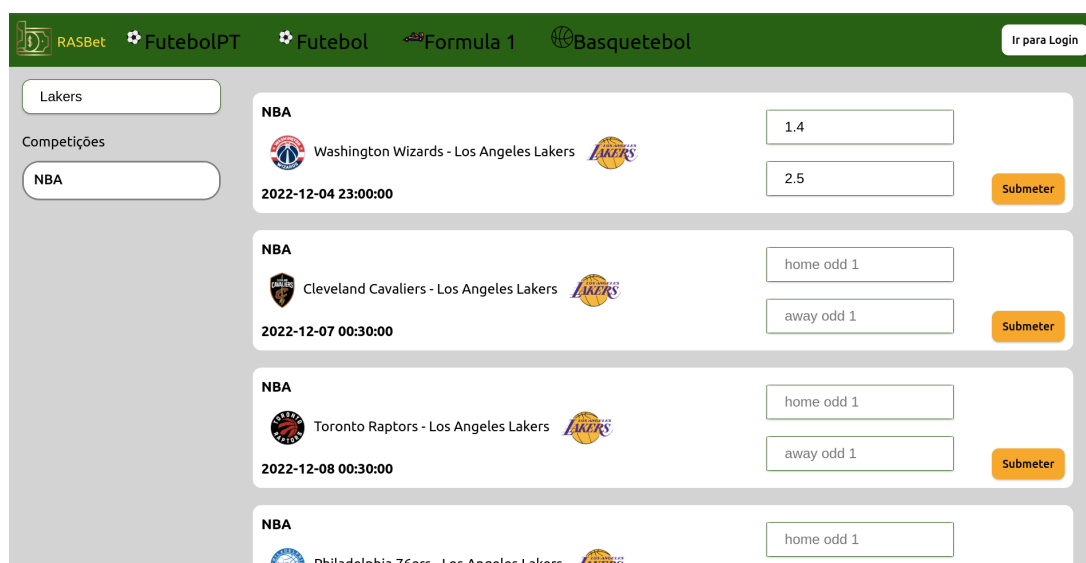


Figura 7.5: Listagem de eventos de Basquetebol para um Especialista

No histórico de apostas, o apostador pode observar as apostas que fez e o seu resultado, de notar que na *Escolha*, *0* corresponde à equipa da casa, *1* corresponde à equipa da casa e *2* corresponde a empate nos jogos com empate, e o *2* não existir nos jogos com empate, e o número do piloto nas corridas.

| Histórico de Apostas | | | | | | |
|----------------------|--|----------|---------------------|---------|--------|--------|
| Tipo | Campeonato:Jogo | Montante | Data | Escolha | Estado | Odd |
| Simples | UEFA Champions League: (0)Bayer Leverkusen - FC Porto(3) | 20 | 2022-12-04 20:19:50 | 1 | WON | 1 |
| Múltipla | Primeira Liga: FC Porto - Arouca Primeira Liga: Braga - Benfica | 15 | 2022-12-04 20:19:01 | 0 1 | PEN | 2.2504 |
| Simples | UEFA Champions League: (4)Napoli - Ajax(2) | 15 | 2022-12-04 20:18:48 | 2 | LOST | 1 |

Figura 7.6: Página de Histórico de Apostas

Na página da carteira, o sistema apresenta as opções de **Levantar** e **Depositar**, onde ao apostador selecionar a que pretende, os sistema exibe a lista dos meios de transação, **MBWay** ou **Multibanco**, para o apostadores escolher. Depois de selecionada uma opção, o sistema mostra a área de introdução do montante, do número de telemóvel ou IBAN (dependendo do modo de transação) e o botão para concluir a transação.

| | | | | | |
|--------|-----------|---------|-----------|-------------|-------------|
| RASBet | FutebolPT | Futebol | Formula 1 | Basquetebol | Carteira: € |
|--------|-----------|---------|-----------|-------------|-------------|

Levantar

Depositar

Selecionar modo:

MB

MB
MULTIBANCO

Montante €

IBAN

Levantar

Figura 7.7: Página da Carteira

Bibliografia

- [1] Refactoring and Design Patterns. (n.d.). Refactoring.Guru. Retrieved November 28, 2022, from <https://refactoring.guru/>