Universidade do Minho

Mestrado em Engenharia Informática

Requisitos e Arquiteturas de *Software*

Fase 3:

Solução Arquitetural

Grupo 1 – PL1


Ana Murta
pg50184


Ana Henriques
pg50196


Hugo Gomes
pg51242


João Lourenço
pg50464


Rui Armada
pg50737

Braga, 17 de janeiro de 2023

Conteúdo

Lista de Figuras

Capítulo 1

Introdução e Objetivos

No âmbito da Unidade Curricular de Requisitos e Arquiteturas de *Software*, foi solicitado o desenvolvimento de uma aplicação de apostas em jogos de vários tipos de desportos. Este capítulo proposita fornecer uma visão geral do projeto, incluindo os objetivos e as fases do desenvolvimento.

A **primeira fase** do projeto teve como principal foco a coleção e análise de requisitos, que foram detalhadamente documentados. No documento de requisitos mencionado, descreveu-se os requisitos funcionais e não funcionais, o domínio do problema e o objetivo geral do sistema.

Na **segunda fase**, foi definido o domínio da solução, começando por revisar os requisitos funcionais propostos na primeira fase. Como tal, foram implementadas algumas das funcionalidades principais, como o registo e o *login* de um usuário, a alteração das informações de perfil do usuário, a consulta de históricos de apostas, de transações e de notificações, entre outras.

Agora na **terceira fase**, foi solicitada a concretização de duas funcionalidades que podem potencialmente destacar a aplicação RASBET no mercado: efetuar apostas múltiplas e seguir jogos nos quais os apostadores estão interessados. Adicionalmente, o grupo procura cumprir a implementação de alguns dos requisitos não funcionais propostos na primeira fase, bem como enriquecer a aplicação com mais funcionalidades anteriormente requisitadas.

O grupo decidiu corrigir e melhorar a solução arquitetural submetida na segunda fase do projeto. Assim sendo, todas as alterações encontram-se sinalizadas a preto e o conteúdo que permaneceu exatamente igual ao original está sinalizado a cinzento.

Em resumo, o projeto tem como objetivo desenvolver uma plataforma de apostas em jogos de desportos, oferecendo funcionalidades intuitivas e fáceis de usar, escaláveis, seguras e de acordo com as regulamentações legais.

Capítulo 2

Restrições

Ao projetar a solução arquitetural para a plataforma de apostas proposta, o grupo enfrentou várias restrições que precisavam ser tidas em conta para garantir que a solução atendia às necessidades do negócio e cumpria com os requisitos regulatórios e de conformidade.

De modo a cumprir com um sistema intuitivo e de rápida aprendizagem de **utilização**, o grupo tomou decisões de implementação que criassem uma interface amigável e simples, juntamente com recursos de ajuda e suporte. Qualquer tipo de modificação necessária também é exiquível sem grandes problemas devido, então, à simples estrutura da solução. Além disto, a aplicação deve também conter um mecanismo de controlo de erros para assegurar a confiabilidade, estabilidade e constante evolução da aplicação (**manutenção**).

Em termos de **escalabilidade**, a plataforma deve ser capaz de suportar um número crescente de usuários e transações sem comprometer o desempenho e a disponibilidade do sistema. A solução arquitetural foi, então, projetada com um modelo escalável, utilizando técnicas de escalonamento automático de recursos que facilitassem a expansão da plataforma, quer a nível de utilizadores, quer a nível de desportos. Além disso, foram implementadas medidas de otimização de desempenho para garantir que se suporta mais de 85% da disponibilidade das funcionalidades do sistema.

A **segurança** da aplicação é garantida através da autenticação dos usuários, onde é preciso cumprir um conjunto de regras para ter acesso à aplicação. Embora não seja possível evitar ataques cibernéticos, existe um mecanismo de recuperação de dados na situação de ataque. Adicionalmente, a solução foi projetada para ser compatível com diferentes *browsers* e em diferentes sistemas operativos, visando permitir o acesso à plataforma de maneira segura e fácil.

A aplicação também valida restrições **legais** estipuladas pelo Decreto-Lei n.º 66/2015, garantindo que os usuários possam realizar apostas de forma segura e legal. Relativamente a limitações no alcance do projeto, o desenvolvimento foi restringido por um prazo e por um orçamento de 200€, o que teve impacto nas decisões de *design* e de implementação.

Adicionalmente, como restrição **cultural**, a solução incluiu medidas para suportar um grande número de idiomas de maneira a abraçar a diversidade e incluir vários usuários de diferentes partes do mundo. No entanto, devido a limitações de prazo, a plataforma está, de momento, disponível apenas em português, mas existem planos de suportar mais idiomas futuramente.

Capítulo 3

Contexto e Âmbito do Sistema

Assim como foi referido anteriormente, este sistema será desenvolvido mediante a construção de uma aplicação *web*, a qual possuirá duas componentes distintas, uma referente ao *backend* e outra referente ao *frontend*.

É na componente do *backend* que se encontram as seguintes camadas da aplicação:

- **Model:** Representa os dados da aplicação e as regras de negócio (*Bet*, *User*, *Game*, ...);
- **DB:** Inicialização da base de dados em *MySQL*;
- **Routes:** Responsável por trabalhar as rotas para cada página da plataforma;
- **Utils:** `API`'s necessárias;
- **Controller:** Reponsável pela lógica do sistema, além de servir como ligação entre o *backend* e o *frontend*.

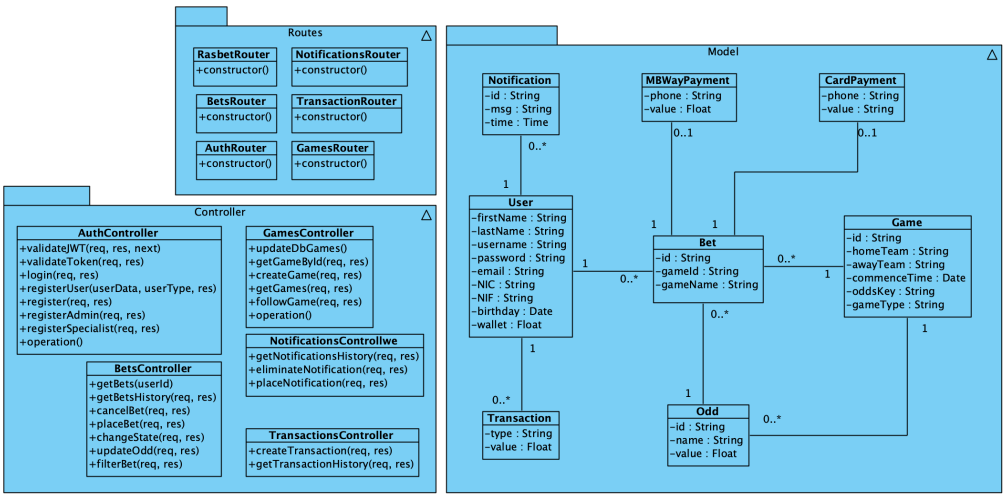

Figura 3.1: Diagrama de Classes

Capítulo 4

Estratégia da Solução

A plataforma de apostas proposta foi desenvolvida com base na utilização da linguagem *Javascript* e do *framework React*, tanto para a concretização do *frontend* como do *backend*. Outra linguagem utilizada foi *Sass*, para auxiliar na construção da componente *frontend* do *website*.

A solução foi projetada com base no **padrão de *design Observer***, o qual é especialmente relevante na funcionalidade de seguir jogos que despertam interesse. A ideia é que, quando as *odds* de um jogo são alteradas, os apostadores interessados (observadores) nesse jogo (objeto observado) são notificados das tais alterações. O mesmo se aplica no caso de apostas múltiplas. O apostador pode cancelar esse interesse em seguir determinado jogo, deixando de ser notificado. Isso garante que os apostadores ficam informados sobre as alterações das *odds* de forma rápida e eficiente, aumentando a confiança e satisfação dos utilizadores.

A estratégia de solução foi baseada nas metas de qualidade estabelecidas para o projeto, incluindo segurança, operabilidade, usabilidade, escalabilidade, manutenção, legalidade e cultura. Perante estas metas, foi necessário pensar onde é que a aplicação carecia para as atingir e proporcionar a melhor experiência ao utilizador. Assim sendo, serão apresentadas as propostas de solução para cada uma das restrições introduzidas na Secção 2.

4.1 Usabilidade

Cenário: O produto deve ser simples e intuitivo de usar, principalmente para iniciantes.

Solução proposta: Desenvolver uma *interface* amigável e de fácil utilização de maneira a reduzir a taxa média de erros de um utilizador inexperiente depois de 25 minutos de utilização para, no máximo, 25%. Além disso, incluir recursos de ajuda e suporte, como páginas de ajuda, guias de utilizador e tutoriais, para facilitar compreender o uso da aplicação. Utilizar uma base de dados para prevenir algum *delay* e oferecer respostas mais rápidas às ações solicitadas pelo utilizador, rejeitando a necessidade de estar constantemente a aceder à `API`.

4.2 Segurança

Cenário: O utilizador deverá possuir os seus dados guardados de forma segura.

Solução proposta: Implementar uma estratégia de encriptação dos dados de modo a preservar a privacidade e a segurança dos dados pessoais e financeiros do utilizador, protegendo-os de acessos não autorizados. Criar cópias de segurança regulares para que, mesmo que os dados originais sejam corrompidos ou apagados, a equipa de desenvolvimento possa restaurar os dados a partir de uma cópia feita anteriormente. Essas cópias devem ser armazenadas em servidores externos para combater a sua vulnerabilidade contra o mesmo ataque aos dados originais.

4.3 Operabilidade

Cenário: O produto deve ser operacional em diferentes *browsers* e sistemas operativos.

Solução proposta: Desenvolver a aplicação com tecnologias *web* padrão e testar regularmente em diferentes *browsers*, como *Chrome*, *Firefox*, e *Safari*, assim como em sistemas operativos como *Windows*, *MacOS* e *Linux*. Utilizar ferramentas de teste automatizadas para assegurar a compatibilidade em diferentes dispositivos e plataformas.

4.4 Escalabilidade

Cenário: A plataforma deve conseguir atingir um alto volume de utilizadores e transações, sem causar lentidão no sistema e sem comprometer a disponibilidade das funcionalidades.

Solução proposta: Utilizar técnicas de escalonamento automático de recursos, como o uso de balanceadores de carga, para garantir que a plataforma consegue lidar com um número crescente de utilizadores e transações sem comprometer o desempenho. Adicionalmente, implementar medidas de otimização de desempenho, como a otimização de consultas à base de dados, para garantir que se suporta mais de 85% da disponibilidade das funcionalidades do sistema.

4.5 Legal

Cenário: O produto deve cumprir o que é estipulado pelo Decreto-Lei n.º 66/2015.

Solução proposta: Implementar verificações de idade e localização dos utilizadores, para garantir que somente os utilizadores maiores de 18 anos e localizados em territórios onde as apostas são legais possam aceder à plataforma. Além disso, garantir que as transações financeiras realizadas na plataforma correspondem com as regulamentações legais. Também deve ser implementada uma política de conformidade que inclui a monitorização regular da plataforma e a tomada de medidas para garantir a conformidade legal.

4.6 Cultural

Cenário: O produto deve ser acessível para utilizadores de diferentes partes do mundo, que falam diferentes idiomas.

Solução proposta: Permitir a seleção de um idioma na *interface*, permitindo que os utilizadores escolham o idioma preferido. Traduzir todo o conteúdo da plataforma para os idiomas mais comuns e atualizar regularmente as traduções para incluir novos idiomas. Para garantir a qualidade das traduções, seria importante contar com a ajuda de profissionais nativos em cada idioma.

4.7 Manutenção

Cenário: A aplicação deve conseguir armazenar novos dados de utilizadores e das `APIs`, garantindo a confiabilidade e estabilidade do sistema.

Solução proposta: Manter um código limpo e bem documentado, utilizar boas práticas de programação e utilizar ferramentas de teste automatizado para garantir a qualidade do código. Estabelecer uma estratégia de gestão de versões e de controlo de mudanças de modo a assegurar que a aplicação pode evoluir controladamente. Além disso, implementar um sistema de relatórios de erros para que o administrador possa identificar e corrigir problemas rapidamente, bem como um sistema de *log* para rastrear as atividades do sistema e identificar a origem de qualquer problema.

Capítulo 5

Building Block View

A arquitetura do sistema foi concebida visando identificar os principais subsistemas e agrupá-los de forma adequada, para uma melhor organização. A Figura 5.1 apresenta o diagrama de componentes elaborado para o nível 1 do sistema, o qual permite analisar como o sistema está dividido em subsistemas e as dependências entre eles. Desta forma, a arquitetura do sistema é dividida em vários subsistemas que trabalham em conjunto para fornecer uma experiência de utilizador satisfatória e garantir a segurança dos dados.

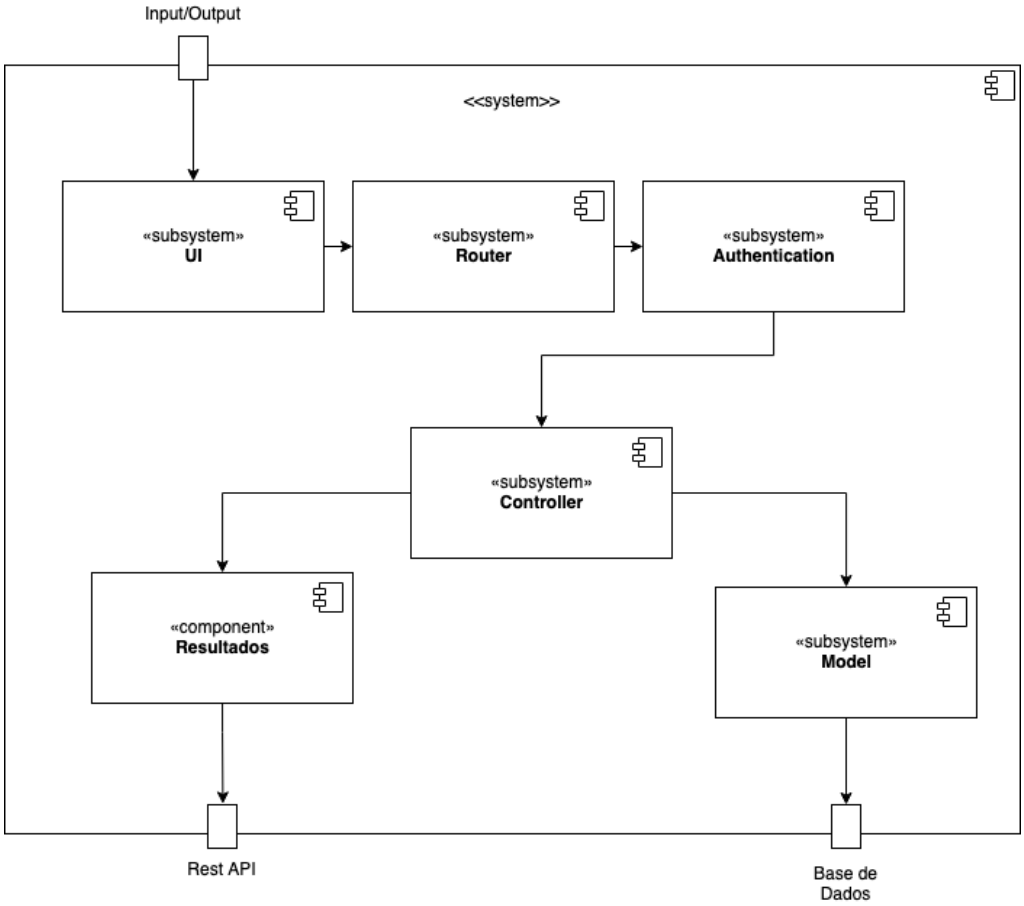

Figura 5.1: Diagrama de Componentes

O **subsistema *UI*** é responsável por criar uma *interface* gráfica que o utilizador utilizará e pela qual interagirá. Como tal, engloba o *frontend* da aplicação e com oeste comunica com o *backend* por pedidos `HTTP`. O *UI* envia e recebe pedidos `HTTP` do *Router*.

O **subsistema *Router*** é o subsistema responsável por redirecionar os pedidos `HTTP` para o *Authentication*, sendo o primeiro a ser acionado quando um utilizador acede à plataforma. A sua tarefa é analisar o pedido e redirecionar para o controlador correto. Isso é feito por rotas pré-configuradas, que especificam que controlador é que deve ser acionado para determinada *URL*.

O **subsistema *Authentication*** atua como uma camada intermediária entre o *Router* e o *Controller*, verificando se o cliente que efetuou o pedido está autenticado no sistema e se tem permissões para fazer o pedido, através da validação de credenciais. Se o utilizador não estiver autenticado ou não tiver permissões, o *Authentication* pode bloquear o acesso ou redirecionar o utilizador para uma página de *login* ou de erro.

O **subsistema *Controller*** é o cérebro da aplicação, tomando decisões sobre como responder a cada pedido e garantindo que a lógica da aplicação é seguida. Isso inclui comunicar com a base de dados, usando o *Model*, para obter informações relevantes e processá-las de conforme as necessidades do pedido. O *Controller* também gera os resultados que serão enviados de volta para o cliente, usando os dados obtidos pelo *Model* e processando-os em função do pedido. Ele é o intermediário entre o componente *Router* e os componentes *Model* e *Resultados*.

O **subsistema *Model*** está encarregue de gerir a interação com a base de dados, desde a criação da mesma até à sua manipulação. Além disto, também preserva a integridade dos dados, garantindo que as operações são executadas corretamente e as regras do negócio são respeitadas. Este fornece uma camada de abstração para o acesso à base de dados, o que permite que os desenvolvedores acedam e manipulem os dados de forma consistente e segura.

O **subsistema *Resultados*** é responsável por gerar e fornecer os dados que serão enviados como resposta ao utilizador, após o *Controller* processar os pedidos `HTTP`. Para tal, os dados são formatados e organizados para serem mais facilmente compreensíveis e utilizáveis pelo utilizador. Este subsistema pode também inclui a lógica de geração de mensagens de erro ou de sucesso, dependendo do resultado das operações realizadas pelo *Controller*.

Capítulo 6

Runtime View

6.1 Registar Utilizador

No Capítulo 2, mencionou-se que não foi cumprida a hierarquia de funcionalidades respetivas a cada tipo de utilizador tal como foi proposto na primeira fase. Por motivos de má gestão de tempo, o grupo desenvolveu o registo de um utilizador sem diferenciar ainda se ele é apostador, especialista ou administrador, dependendo dos dados impostos para validar cada um dos tipos.

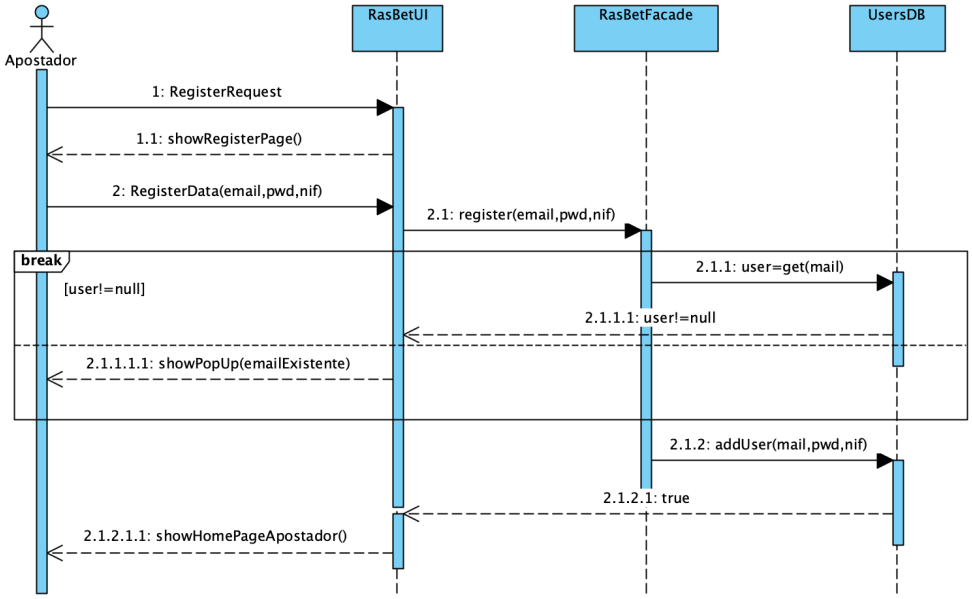

Figura 6.1: Diagrama de Sequência de Registar Utilizador

6.2 *Login* Utilizador

Considerando novamente o incumprimento da diferenciação do tipo de utilizador, esta funcionalidade foi concebida, fornecendo apenas os dados que o utilizador precisa para poder entrar na aplicação. No entanto, como as informações requeridas dizem respeito apenas às do apostador, ao aceder à plataforma, o utilizador só terá acesso às funcionalidades de um apostador.

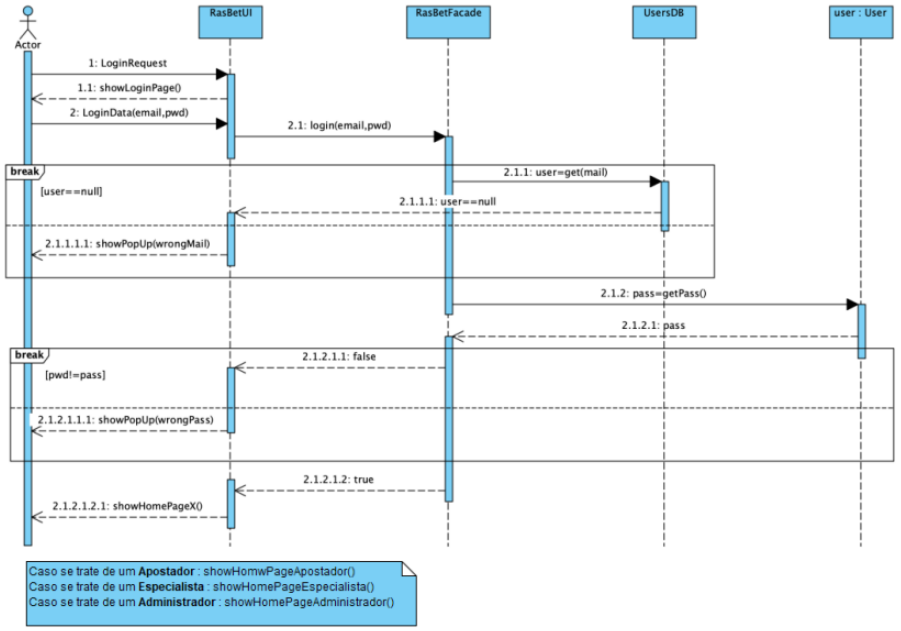

Figura 6.2: Diagrama de Sequência de *Login* Utilizador

6.3 Consultar Jogos Desportivos

O utilizador pode consultar os vários jogos disponíveis para futebol e basquetebol, selecionando primeiramente o tipo de desporto em questão e, posteriormente, pode até fazer uma pesquisa para chegar ao resultado pretendido mais rapidamente. Relativamente aos outros desportos, como ténis e *motoGP*, ainda não existem jogos disponíveis.

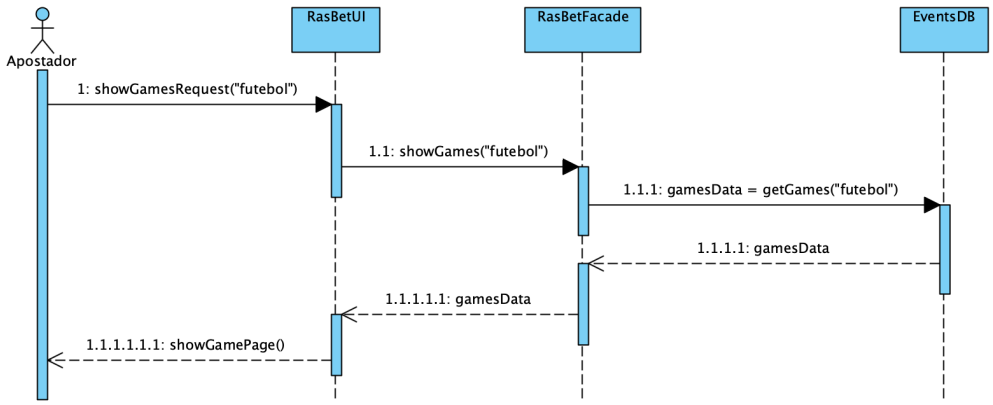

Figura 6.3: Diagrama de Sequência de Consultar Jogos

6.4 Fazer Aposta Simples ou Múltipla

O utilizador pode efetuar uma aposta simples, múltipla ou combinada, identificando em primeiro lugar o tipo da aposta. Seguidamente, é preciso escolher o(s) jogo(s) desportivo(s) no(s) qual/ais quer apostar e o valor da *odd*. Para terminar o processo, tem de pagar, optando por um meio de pagamento. A aposta será finalizada assim que o pagamento for validado.

O processo de fazer múltiplas apostas, onde o apostador escolhe até 20 jogos e combiná-los todos numa única aposta múltipla, segue o mesmo processo que de uma aposta simples, com a diferença em que são introduzidos mais que um jogo e possivelmente mais do que valor.

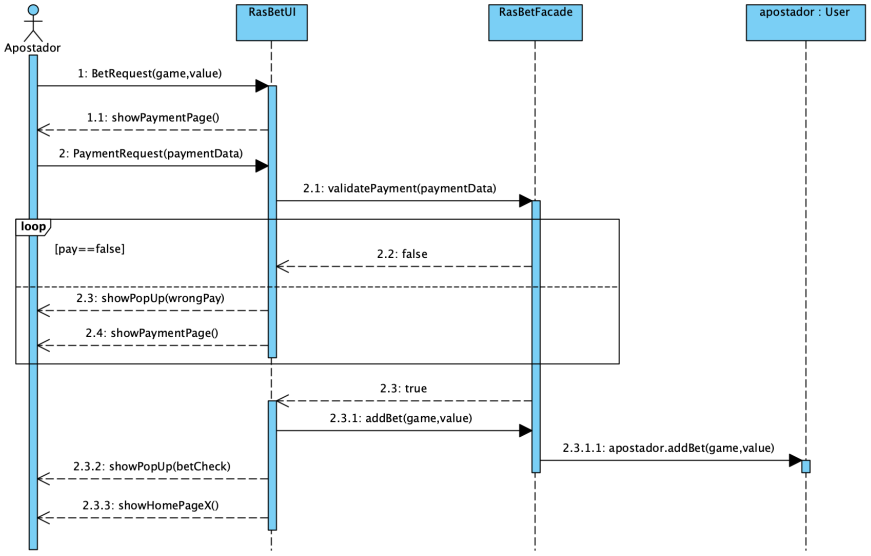

Figura 6.4: Diagrama de Sequência de Fazer Aposta

6.5 Inserir *Odd*

A funcionalidade de inserir uma *odd* (e também alterá-la) pertence a um especialista. Todavia, e como igualmente mencionado no Capítulo 2, embora não tenham sido estabelecida a ligação com o resto do *website*, existem algumas funcionalidades destinadas a especialistas e administradores que foram desenvolvidas soltamente, é o caso desta. Para inserir, então, uma *odd*, o utilizador precisa de selecionar o desporto, o jogo e o valor.

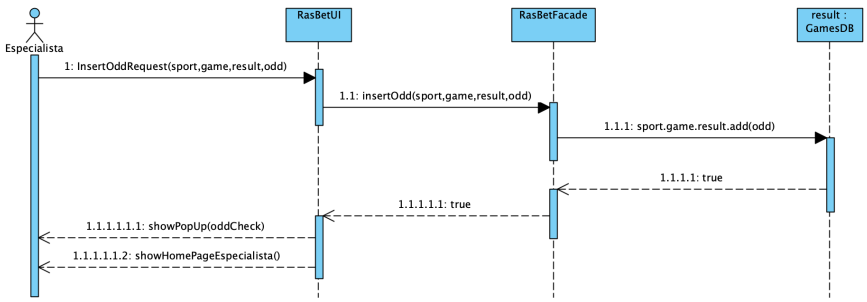

Figura 6.5: Diagrama de Sequência de Inserir *Odd*

6.6 Consultar Histórico de Apostas/Transações

Para consultar tanto o histórico de apostas, como o histórico de apostas, o utilizador precisa de apenas de aceder ao conjunto de funcionalidades no botão identificativo do seu perfil. O processo de consultar o painel de notificações, é exatamente igual.

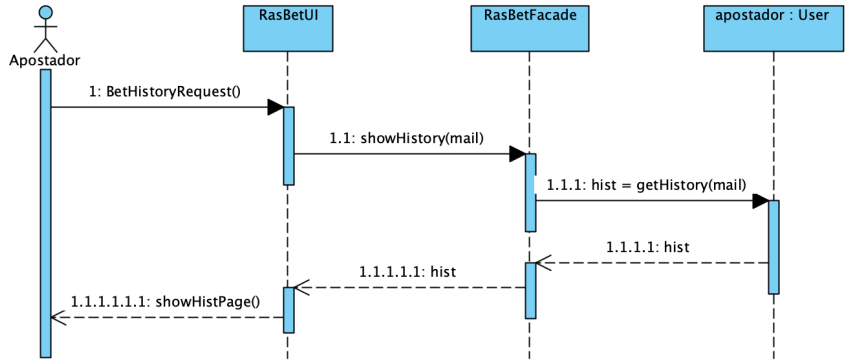

Figura 6.6: Diagrama de Sequência de Consultar Apostas

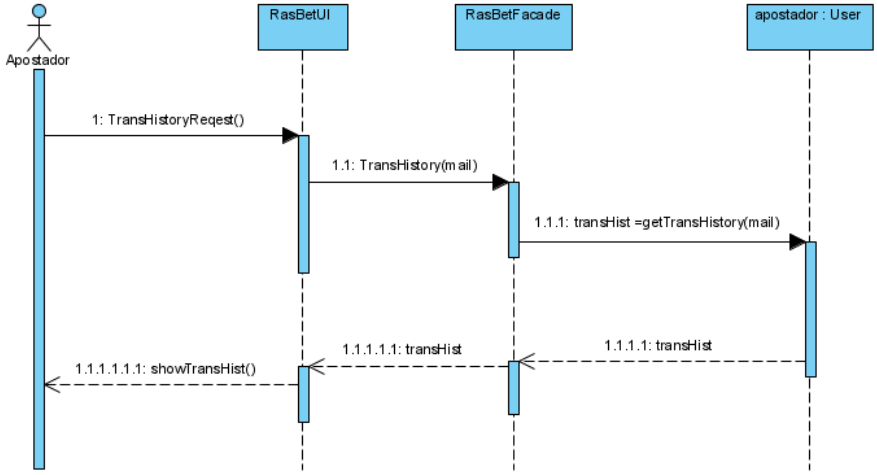

Figura 6.7: Diagrama de Sequência de Consultar Transações

6.7 Seguir Jogo

Para seguir um jogo, o utilizador precisa de selecionar o jogo no qual está interessado e, perante isto, poderá optar por continuar o processo ou cancelá-lo. Se optar por continuar, o jogo será então adicionado à lista de jogos interessados.

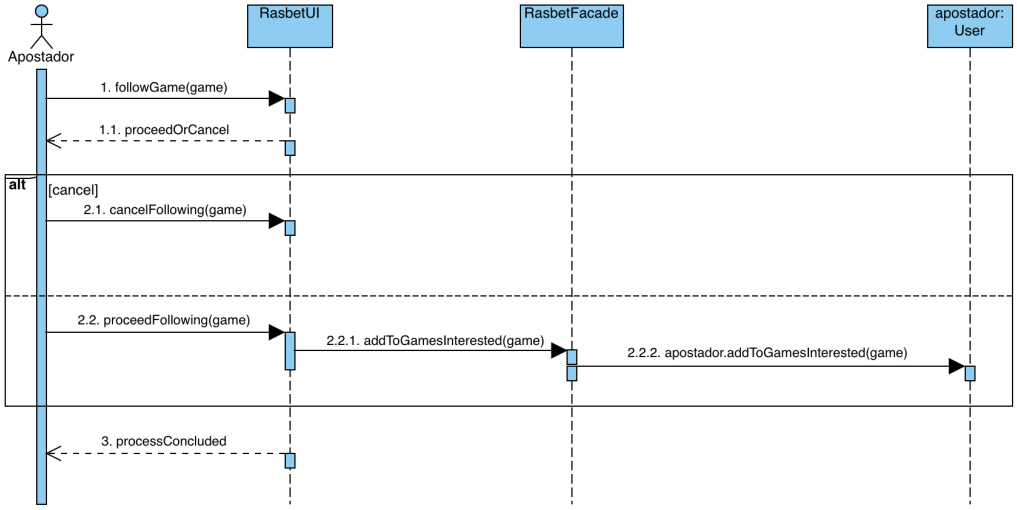

Figura 6.8: Diagrama de Sequência de Seguir Jogo

6.8 Informar o apostador de alterações nas odds

Se forem efetuadas alterações às *odds* de algum jogo seguido pelo apostador, o mesmo deve ser informado da situação e, para tal, usámos a estratégia de, mudando o estado da aposta, o apostador interessado no jogo em questão é notificado.

Capítulo 7

Deployment View

Tal como se pode observar pela Figura 7.1, a solução desenvolvida tem por base várias componentes interligadas, cuja interação contribui para o bom funcionamento da aplicação. O primeiro passo foi construir diversas *APIs* que contêm informação acerca de jogos de futebol, de basquetebol, de ténis, entre outros tipos de desportos. Também foi implementada uma base de dados em *MySQL*, na qual estão armazenadas informações acerca dos utilizadores da aplicação.

As *APIs* e a *Database* interagem com o *backend* da aplicação construída, que, por sua vez, consegue aceder, alterar e eliminar informação destas componentes. O *frontend*, apresentado ao utilizador, interage com o *backend* com o objetivo aceder a métodos, alterar informações *via user input* e apresentar informações das *APIs* que interagem com o *backend*. Isto possibilita a solicitação de dados ao *backend* para os apresentar ao utilizador. Finalmente, o utilizador recorre ao seu *web-browser* para estabelecer uma connexão conexãoHTTP/HTTPS com a plataforma.

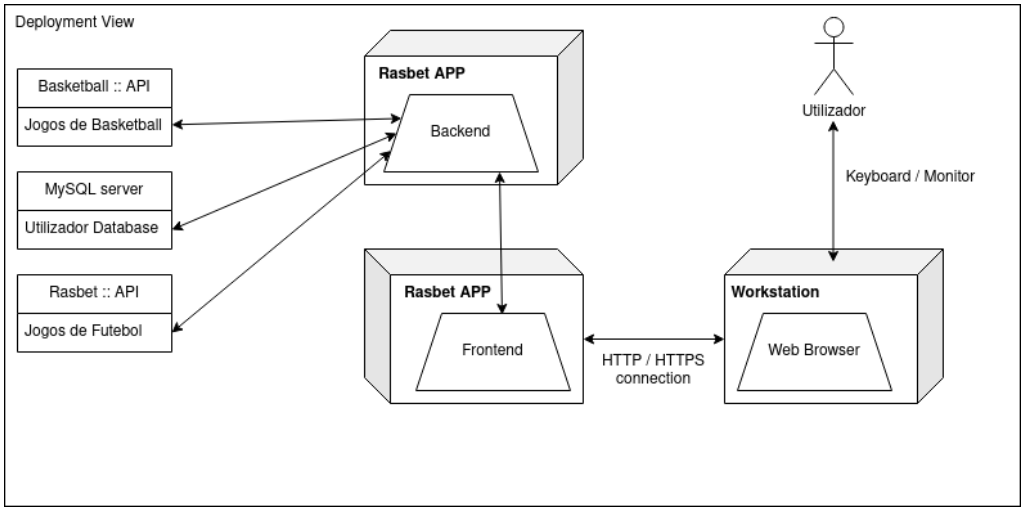

Figura 7.1: Diagrama ilustrativo do *Deployment View*

Capítulo 8

Conclusões e Trabalho Futuro

Nesta terceira fase do trabalho prático, foi possível elaborar o planeamento e implementação da arquitetura da aplicação a desenvolver, destacando as alterações e os melhoramentos feitos agora na terceira fase. Adicionalmente, foi necessário definir estratégias importantes que fornecessem uma boa experiência ao utilizador, melhorando, consequentemente, bastante a qualidade da aplicação.

Para além da implementação das duas funcionalidades propostas para esta fase, ainda foi prestado especial cuidado no cumprimento de mais requisitos levantados na fase anterior, dando mais importância aos requisitos não funcionais. No final, o grupo apenas ambicionou manter o rigor para a fase final deste projeto que se manteve nas outras.

Deste modo, foi desenvolvida uma solução satisfatória para os problemas e requisitos anteriormente propostos. Para trabalho futuro, a equipa de trabalho deseja concluir a implementação de todos os requisitos definidos na primeira fase, tal como aperfeiçoar algumas questões, se for necessário, construídas na presente fase do projeto.