

UMinho
Mestrado Engenharia Informática
Requisitos e Arquiteturas de Software

PL3 / Entrega 2

PG 50462	Nome: João André Faria de Freitas
PG 50487	Nome: João Pedro Fontes Delgado
PG 50271	Nome: Bruno Filipe Miranda Pereira
PG 50379	Nome: Francisco José Martinho Toldy
PG 50180	Nome: Ana Catarina Oliveira Gonçalves



4 de Dezembro de 2022

2022/23

Conteúdo

1	Introdução e objetivos	3
1.1	Overview dos Requisitos	3
1.2	Quality Goals	4
1.3	Stakeholders	5
2	Restrições de Arquitetura	6
2.1	Restrições à Solução	6
2.2	Restrições de Prazo/Agendamento	6
3	Contexto e Scope	7
3.1	Contexto de Negócio	7
3.1.1	Sistemas Informáticos	7
3.1.2	Utilizadores	7
4	Estratégia de Solução	9
4.1	Padrão de Arquitetura	9
4.2	Decisões de Tecnologia	9
5	Building Block View	10
5.1	Whitebox do Sistema global	11
5.2	Nível 1	11
5.3	Nível 2	11
5.4	Diagrama de Classes	15
5.4.1	Descrição das Classes	16
5.5	Descrição dos Serviços da Aplicação	18
5.6	Diagrama de Componentes	21
5.7	Diagrama de Packages	22
6	Runtime View	23
6.1	Diagramas de Sequência	23
6.1.1	Consultar histórico de apostas e Estatísticas de Ganhos	23
6.1.2	Fazer procura de jogos na API RASBet	24
6.1.3	Alterar estado de um jogo	25
6.1.4	Fazer uma Aposta	26
6.1.5	Fazer um depósito	27
6.1.6	Fazer um depósito	28
6.1.7	Criar um jogo	29
7	Deployment View	30
7.1	Infraestrutura de Nível 1	30

Lista de Figuras

5.1	Diagrama WhiteBox	10
5.2	Diagrama Classes	15
5.3	Diagrama Classes	16
5.4	Diagrama de Componentes	21
5.5	Diagrama de Packages	22
6.1	Diagrama de Sequência de consultar histórico de apostas e estatísticas de ganhos.	23
6.2	Diagrama de Sequência de fazer procura de jogos na API RASBet	24
6.3	Diagrama de Sequência de alterar estado de um jogo	25
6.4	Diagrama de Sequência de realizar uma aposta.	26
6.5	Diagrama de Sequência de realizar um depósito na conta.	27
6.6	Diagrama de Sequência de realizar um depósito na conta.	28
6.7	Diagrama de Sequência da criação de um jogo por parte de um especialista . . .	29
7.1	Diagrama de Deployment	30

1. Introdução e objetivos

Nesta **segunda** terceira fase do projeto apresentado, o grupo começou por uma análise das alterações a serem efetuadas de modo a implementar todas as funcionalidades desejadas. Após essa análise foi construída a nova solução e finalizamos a solução com o seguinte documento, isto é, uma documentação da arquitetura do software desenvolvido, **juntamente com todas as alterações feitas para incluir os novos requisitos**.

Relativamente à solução criada, podemos confirmar que a solução desenvolvida satisfaz os principais objetivos de negócio, os quais são:

- O apostador conseguir fazer várias apostas nas modalidades desejadas
- O administrador conseguir controlar aspetos relacionados com os especialistas
- O especialista conseguir controlar aspetos relacionados com os jogos e apostas
- O apostador conseguir seguir um jogo

1.1 Overview dos Requisitos

A satisfação dos objetivos pode ser diretamente associada aos requisitos funcionais que foram levantados. Assim sendo, apresentam-se a seguir todos os requisitos funcionais:

- O Apostador tem de se registar na aplicação para a poder utilizar
- O Apostador tem de iniciar sessão na aplicação para a poder utilizar.
- O Especialista tem de iniciar sessão na aplicação para a poder utilizar
- O Administrador tem de iniciar sessão na aplicação para a poder utilizar
- O Apostador pode consultar o seu histórico de apostas
- O Apostador consegue realizar uma consulta estatística dos seus ganhos na aplicação
- O utilizador tem capacidade de alterar 'telemóvel', 'nome' e 'morada' do seu perfil
- O utilizador consulta o seu histórico de transações
- O Apostador valida os dados da aposta para a submeter
- O utilizador com credenciais de especialista pode adicionar eventos desportivos, bem como as odds associadas aos mesmos.
- O utilizador com credenciais de especialista pode alterar eventos desportivos, bem como as odds associadas aos mesmos.
- O Apostador tem a capacidade de procurar jogos pelos nomes das equipas

- O Apostador tem a possibilidade de adicionar pelo menos uma aposta ao boletim
- O Apostador tem a possibilidade de remover apostas do boletim
- O Administrador deve ser capaz de ver quais os especialistas registados no sistema
- O Administrador deve ser capaz de eliminar especialistas do sistema.
- O Apostador/Especialista deve ser capaz de recuperar a palavra passe
- O sistema tem de suportar apostas em diversos tipos de desportos
- Uma aposta tem obrigatoriamente um de 3 estados, Aberta, Fechada, Suspensa
- A aplicação deve ter um calendário de jogos
- A aplicação deve permitir ao utilizador realizar diversas apostas em eventos desportivos diferentes, recorrendo ao saldo disponível da carteira.
- A aplicação permite o registo de novos utilizadores
- A aplicação permite o carregamento/levantamento de saldo da sua carteira através de 2 métodos de pagamento distintos
- O sistema tem de notificar os apostadores dos resultados das suas apostas
- O sistema tem de ser um serviço web
- O sistema deverá suportar dois tipos de apostas, apostas simples e apostas múltiplas
- O especialista pode finalizar qualquer jogo criado por um especialista com o resultado que inserir
- O Apostador tem a possibilidade de seguir jogos.
- O Apostador tem a possibilidade de visualizar a lista de jogos seguidos
- O Apostador tem a possibilidade de deixar de seguir um jogo

1.2 Quality Goals

Assim sendo, a seguir são apresentados os requisitos funcionais e não funcionais com maior interesse a serem implementados:

Prioridade	Requisito
1	Sistema deve garantir que nenhum utilizador têm acesso à conta de outro utilizador
3	O sistema deve ter um calendário de jogos atualizado
2	O apostador deve ser capaz de realizar apostas
4	O especialista é capaz de adicionar eventos desportivos, bem como as odds
5	O administrador é capaz de criar credenciais para especialistas

1.3 Stakeholders

Segue-se a lista dos vários stakeholders para esta aplicação.

Nome/Papel	Descrição
Cliente	Entidade para a qual o sistema está a ser desenvolvido
Consumidor	Utilizadores finais do projeto, apostadores
Equipa	Grupo de developers que irá trabalhar no sistema

2. Restrições de Arquitetura

No desenvolvimento da solução, foram impostas 3 restrições pelo cliente. Devido à existência destas, a equipa de desenvolvimento teve um cuidado extra, durante o desenvolvimento da solução, para as conseguir cumprir.

2.1 Restrições à Solução

- **Descrição :** A aplicação tem que ser um web site

Justificação : A aplicação deverá poder ser usada em qualquer lugar, a partir do seu computador.

- **Descrição :** A solução deverá recorrer à API RASBet fornecida pelos docentes

Justificação : O uso desta API é obrigatório de acordo com o enunciado estabelecido pelos docentes, dado que a forma como esta é utilizada constitui uma das métricas de avaliação.

2.2 Restrições de Prazo/Agendamento

- **Descrição :** A equipa tem um período de quatro semanas para completar a solução, implementando todos os requisitos funcionais e não funcionais e documentação relativamente à solução arquitetural concebida.

- **Justificação :** Para que seja permitida a avaliação do projeto na sua fase intermédia, é necessário uma entrega que contenha a documentação pertinente à mesma. Essa documentação será constituída pelas alterações à contextualização e definição de requisitos bem como este documento relativo à arquitetura e a solução no seu presente estado.

3. Contexto e Scope

Nesta secção vão ser especificados todos os parceiros de comunicação do sistema, incluindo utilizadores, outros sistemas informáticos, etc.

3.1 Contexto de Negócio

3.1.1 Sistemas Informáticos

- **RASBet API** - esta API, apresentada pelos docentes, tem a funcionalidade de fornecer informações sobre os jogos da Liga Portuguesa de futebol. A cada jogo vão estar associadas várias informações, tais como a data, hora e odds de cada resultado de acordo com várias casas de apostas.

Esta API vê a sua informação atualizada 2 vezes por dia e comunica com o sistema desenvolvido através de pedidos HTTP.

O sistema implementado pede a informação da API a cada minuto. Dado que o sistema terá que estar protegido de eventuais falhas ou atrasos na entrega de informação, esta é inserida na base de dados do sistema de forma a que esteja sempre disponível na base de dados, podendo os pedidos de informação recorrer à mesma através de pedidos à back-end. Isto protege o sistema uma vez que mesmo que a API falhe, existe sempre uma versão dos dados na base de dados.

- **Serviço de envio de emails automático** - utilizado pela back-end para auxílio em várias funcionalidades do sistema. Este serviço envia emails automáticos através de um endereço fornecido, sendo a informação contida nos mesmos definida pela back-end consoante o pedido em tratamento. Isto permite ao sistema enviar códigos de segurança na funcionalidade de mudança de dados seguros, enviar a password do utilizador caso este a tenha esquecido e, no processo do término de uma aposta, permitir que o utilizador seja notificado com o resultado da mesma.
- **The Odds API** - Uma vez que a API fornecida pelos docentes só suporta jogos da Liga Portuguesa de futebol, será necessário complementar essa informação recorrendo a outra API, à qual será pedida informação sobre jogos de basquetebol da NBA. A integração desses dados no sistema funcionará de forma igual à integração da RASBet API, sendo que os dados recebidos serão inseridos na base de dados. Tal como a API fornecida pelos docentes, são feitos pedidos à The Odds API a cada minuto.

3.1.2 Utilizadores

- **Administrador** - O sistema terá um único administrador, com o seu próprio Login. Este irá interagir com o sistema para várias funcionalidades relativas aos Especialistas. Conseguirá criar contas para os especialistas utilizarem, bem como consultar a lista dos mesmos e ainda eliminá-las.

- **Especialista** - O sistema irá interagir com vários especialistas, também com Login próprio. Os Especialistas poderão interagir com o sistema para várias funcionalidades relativas aos Eventos, bem como às Apostas. As funcionalidades principais que o Especialista irá usufruir são a criação e listagem de Eventos, alteração de odds, atribuição de resultados e alteração de estado de apostas.
- **Apostador** - Tipo de utilizador mais frequente do sistema. Os Apostadores poderão interagir com o sistema para várias funcionalidades relativas às Apostas bem como movimentações de fundos da conta e alteração e consulta de informações relativas à sua conta. O Apostador será capaz de apostar em um ou mais eventos, recorrendo a fundos depositados na conta. Além disso, este tipo de utilizador poderá consultar o histórico de apostas e transações e alterar informações relativas à sua conta. Além disso poderá seguir vários jogos.

4. Estratégia de Solução

Uma das primeiras decisões necessárias relativamente à arquitetura do sistema seria a escolha entre Client side rendering e server side rendering (SSR). Dado que o produto a desenvolver seria uma aplicação web e a equipa de desenvolvimento tinha mais conhecimento e experiência com Client Side Rendering, foi esta arquitetura que foi escolhida. Além do fator do tipo de produto em mente, esta escolha foi feita com base noutras fatores como o suporte robusto por parte de várias bibliotecas de JavaScript, e ser o indicado para sites com grandes números de utilizadores, sendo este o objetivo para este produto.

4.1 Padrão de Arquitetura

A nível de arquitetura, uma vez que o sistema teria que ser capaz de processar os pedidos de vários utilizadores em simultâneo, foi tomada a decisão de seguir o padrão arquitetural Client-Server.

4.2 Decisões de Tecnologia

A nível de decisões tecnológicas, foi necessário selecionar quais as ferramentas a utilizar. Além dos IDEs, foi preciso escolher as linguagens de programação a utilizar para o desenvolvimento da front end e da back-end. O critério de seleção para as duas foi principalmente, além de serem adequadas para o desenvolvimento web, a familiaridade da equipa com as mesmas.

Para a front-end foi selecionado JavaScript, mais especificamente a biblioteca React. O React JS é uma excelente biblioteca para criação de componentes, que contribuíram para a modularização do código desenvolvido. Além disso, é capaz de suportar a criação de páginas interativas necessárias para o sistema a desenvolver.

Para a back-end foi selecionado o Java Spring, uma framework open-source que providencia uma infraestrutura que suporta o desenvolvimento de aplicações Java. Este permitiu adotar uma estrutura bem formada para a aplicação, tendo então 3 camadas principais, os controladores, que é onde são definidas as estruturas dos pedidos que terão de ser feitos, bem como os endpoints, os modelos, que é onde estão definidas as estruturas das classes e onde, através de anotações Spring é possível definir a estrutura da Base de Dados e relações entre as entidades, os serviços, que é onde é feita toda a lógica aplicacional e, por fim, os repositórios, que é onde é feita a ligação efetiva à base de dados, através do Spring Boot JPA Repository.
Para a base de dados recorreu-se ao MySQL, devido à experiência que o grupo já tinha adquirido com as ferramentas subjacentes a este motor de base de dados.

5. Building Block View

Nesta secção vai ser apresentada a decomposição estática do sistema, constituída por blocos que podem ser módulos, componentes, subsistemas, etc.

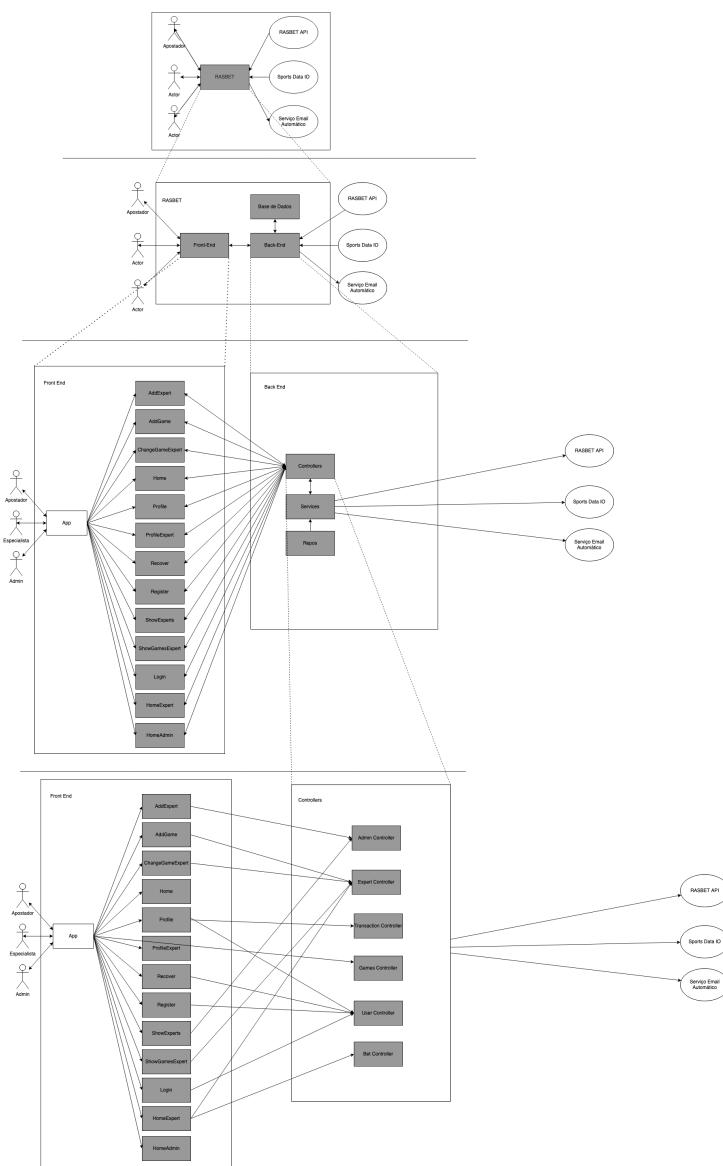


Figura 5.1: Diagrama WhiteBox

5.1 Whitebox do Sistema global

Nome	Responsabilidade
RASBET	Sistema usado pelos utilizadores
RASBET API	API com dados da Liga Portuguesa
The Odds API	API com dados de jogos de Basquetebol
Serviço Email Automático	Notificações diversas do utilizador

5.2 Nível 1

Nome	Responsabilidade
Front-End	Parte do sistema com que o utilizador interage
Back-End	Serviços do sistema
Base de Dados	Armazenamento de dados de todo o sistema

5.3 Nível 2

Nome	Responsabilidade
App	Página base para todas as outras
AddExpert	Página para o Administrador adicionar Especialistas
AddGame	Página para o Especialista adicionar um Evento Desportivo
ChangeGameExpert	Página de Finalização de Jogo
Home	Página principal do Apostador
HomeAdmin	Página principal do Administrador
HomeExpert	Página principal do Especialista
Login	Página inicial, Login dos utilizadores
Profile	Página com todas as funcionalidades relativas ao perfil do Apostador.
ProfileExpert	Página de perfil do Especialista
Recover	Página de recuperação de password
Register	Página de Registo do Apostador
ShowExperts	Página com a listagem dos Especialistas
ShowGamesExpert	Página com listagem de eventos criados pelo Especialista
FollowPage	Página com a listagem dos eventos seguidos

Nome	Responsabilidade
Controllers	Fornece um ponto para o qual devem ser encaminhados os pedidos
Services	Concentra a lógica de negócio
Repos	Oferece uma interface para interagir com a base de dados

- **App**

Página base que vai apresentar as diferentes páginas dependendo do desejo do utilizador.

- **AddExpert**

A página AddExpert é responsável pela criação de um especialista por parte do Administrador do sistema. Esta página tem um formulário responsável por guardar os inputs do Administrador relativos aos dados do Especialista a criar. Além disso, possui variáveis capazes de controlar o conditional rendering de mensagens de erro necessárias para cumprir os requisitos do sistema e melhorar a experiência do utilizador. Estas mensagens de erro são acionadas por verificações dos inputs no momento de confirmação de criação de especialista. Por último, a página tem a si associado um componente **PopUpAdmin** que é acionado após a confirmação da backend, mostrando ao utilizador os dados do Especialista que acabou de adicionar. Este componente é controlado também por uma variável. Além disso é possível mudar desta página de volta para página principal do administrador automaticamente após fechar o componente referido.

- **AddGame**

Esta página segue um raciocínio semelhante à pagina **AddExpert**, no entanto, é responsável pela criação dos eventos desportivos por parte de um especialista. A página tem um componente form associado, onde será armazenada qualquer informação introduzida relativa ao evento a adicionar.

A ação de mostrar certos componentes da página está dependente do valor escolhido no primeiro input - a escolha do desporto no qual o evento se enquadra . Os inputs que são mostrados ao utilizador após essa escolha estão em conformidade com a mesma, assim, caso o utilizador selecione um desporto sem empates, os inputs das odds são alterados de forma apropriada. Caso o utilizador escolha adicionar uma corrida, é mostrado uma lista de inputs consistente com o número de pilotos numa corrida de motoGP (componente auxiliar **PilotsForm**)

A página está completa com mensagens de erro apropriadas, resultantes de verificações feitas no momento de submissão do formulário. Tal como acontece na página referente a adicionar um especialista, existe um componente relativo a um popUp de confirmação de criação de evento (componente **PopUpAddGame**) em conditional rendering.

- **ChangeGameExpert**

Página que lida com o pedido do especialista de finalizar um dado jogo. Após a inserção dos dados dependendo do jogo escolhido, vai ser enviado um request para finalizar esse jogo com os dados fornecidos. Além disso, vai ser feito outro pedido para atualizar os jogos atuais, visto que um jogo finalizado já não deve aparecer nem para o especialista nem para o apostador

- **Home**

Nesta página o apostador vai poder ter acesso a todos os jogos ativos e não finalizados, juntamente com as apostas correspondentes. Aqui vai também poder selecionar várias apostas, que vão ser adicionadas ao boletim, e finalmente poderá finalizar a aposta se estiver em condições disso (aposta válida e saldo suficiente) Além disso, vai ter ao seu dispotar um ícon para adicionar/retirar um jogo dos seus favoritos (para ser notificado de alterações)

- **HomeAdmin**

Nesta página o administrador vai conseguir ter acesso à 3 opções, adicionar especialistas, consultar especialistas (onde vai conseguir remover) e finalmente o logout

- **HomeExpert**

Nesta página o especialista vai poder ter acesso a todos os jogos não finalizados. A partir desta página, vai conseguir editar odds de qualquer jogo, como também editar o seu estado.

- **Login**

A primeira página vista por qualquer um dos utilizadores, é constituída por um componente **InputsLogin** onde será recolhido o input do utilizador e enviado o pedido à backend para concretizar o Login. Também nesta página inicial o utilizador pode decidir entre dark e light mode.

- **Profile**

Página na qual são executadas todas as funcionalidades relativas ao perfil do Apostador.

A página é constituída por um componente **IdSaldo** presente independentemente de qualquer conditional rendering. Além desse componente, podem ser apresentados os componentes **ChangeData** ou **BetHistory** consoante o valor da variável **divChoice**. Existem ainda 3 componentes relativos a popUps

- **PopUpOperation**
- **PopUpCodeEmail**
- **PopUpCodeConfirm**

O componente ChangeData é responsável pelos inputs de dados que o utilizador pode mudar. Quer sejam estes secretos ou não, sendo essa distinção de renderização controlada através de conditional rendering. Esse conditional rendering é controlado através de 2 dos PopUps acima referidos (**PopUpCodeEmail** e **PopUpCodeConfirm**) relativos a operação de pedir e introduzir o código de segurança.

O componente BetHistory é responsável por mostrar ao utilizador os históricos de apostas e transações. Os dados das apostas são mostrados através de uma lista que reutiliza um componente genérico **BetHistoryBox** de forma a mostrar uma lista dinâmica com a informação pedida. Os dados relativos às transações são mostrados recorrendo a um componente **TableTransact**, capaz de mostrar uma lista de transações com comprimento dinâmico.

As operações monetárias também são possíveis a partir desta página, via os botões **Depositar** e **Levantar**. Estes botões irão ativar o componente **PopUpOperation**, sendo que no decorrer da operação de levantamento ou depósito, serão ainda usados os componente **PopUpMethod** para selecionar o método de transferência e depois **PopUpPaypal** ou **PopUpMaestro** consoante a escolha. Estes componentes possuem o seu próprio controlo e notificação de erros apropriado

- **FollowPage**

Com o componente "FollowPage" vai ser possível facilitar a distinção dos jogos que estão a ser seguidos. Esta página tem um aspeto semelhante à pagina relativa a consulta de histórico de transações, sendo que tem uma lista de componentes BetFollowDiv , cada um para cada evento marcado como seguido. Esse componente BetFollowDiv será depois constituído pela informação sobre o evento (componente BetFollow ou BetFollowMotoGP). Além disso contém ainda um ícone que permite remover o evento em questão da lista de eventos seguidos.

- **ProfileExpert**

Nesta página, o especialista vai ter as opções de criar evento, consultar evento (onde vai poder finaliza-los) e finalmente o logout

- **Recover**

Página relativa a recuperação de password de um dado utilizador, seja este Apostador ou Especialista

Constituída, tal como a página **Login** e **Register**, por uma imagem e um componente, neste caso **RetrievePass**.

- **Register**

Apesar do seu aspetto e código ser semelhante à pagina **Login**, foi tomada a decisão de separar as páginas invés de recorrer a conditional rendering para usar o componente **RegAccount**.

- **ShowExperts**

Esta página permite ao administrador consultar todos os especialista existentes, como também removê-los do sistema

- ShowGamesExperts

Esta página permite ao especialista ter acesso a todos os jogos criados por especialistas, e deste modo escolher resultados para finalizar o jogo.

5.4 Diagrama de Classes

O diagrama de classes que se segue apresenta as classes que compõe o sistema e os seus atributos e métodos, bem como as relações entre as mesmas.

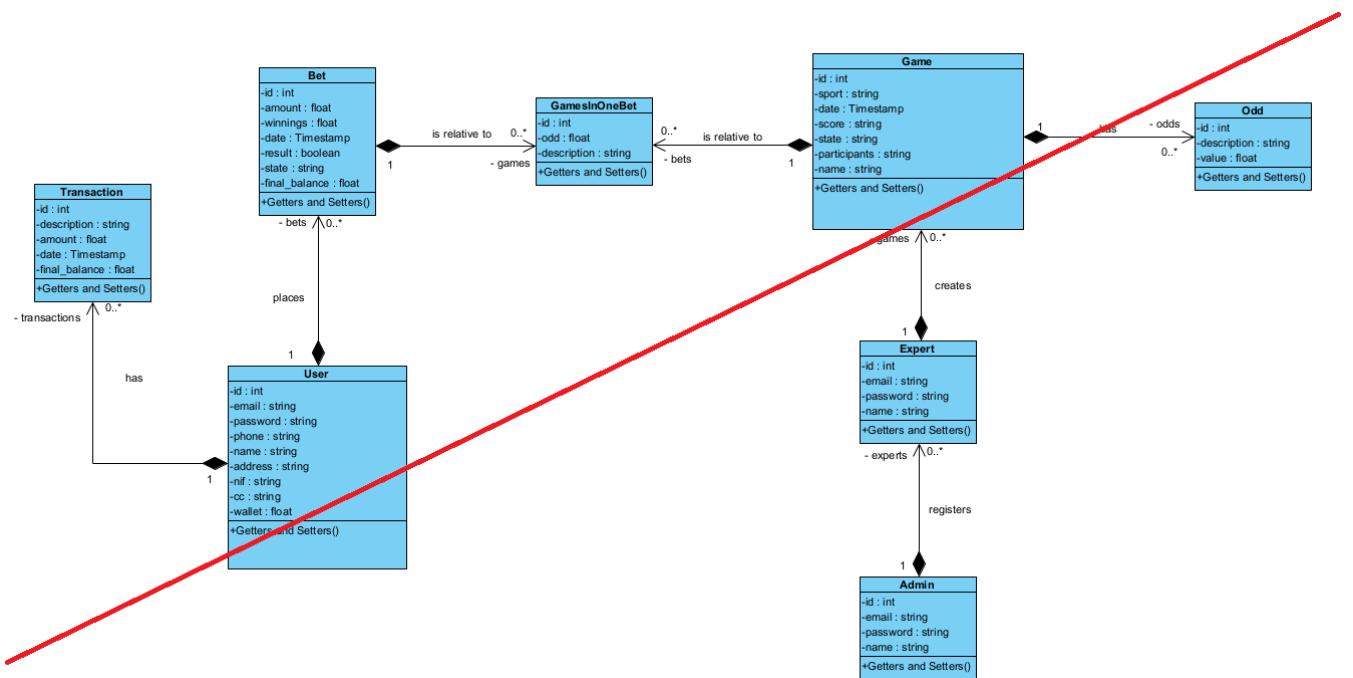


Figura 5.2: Diagrama Classes

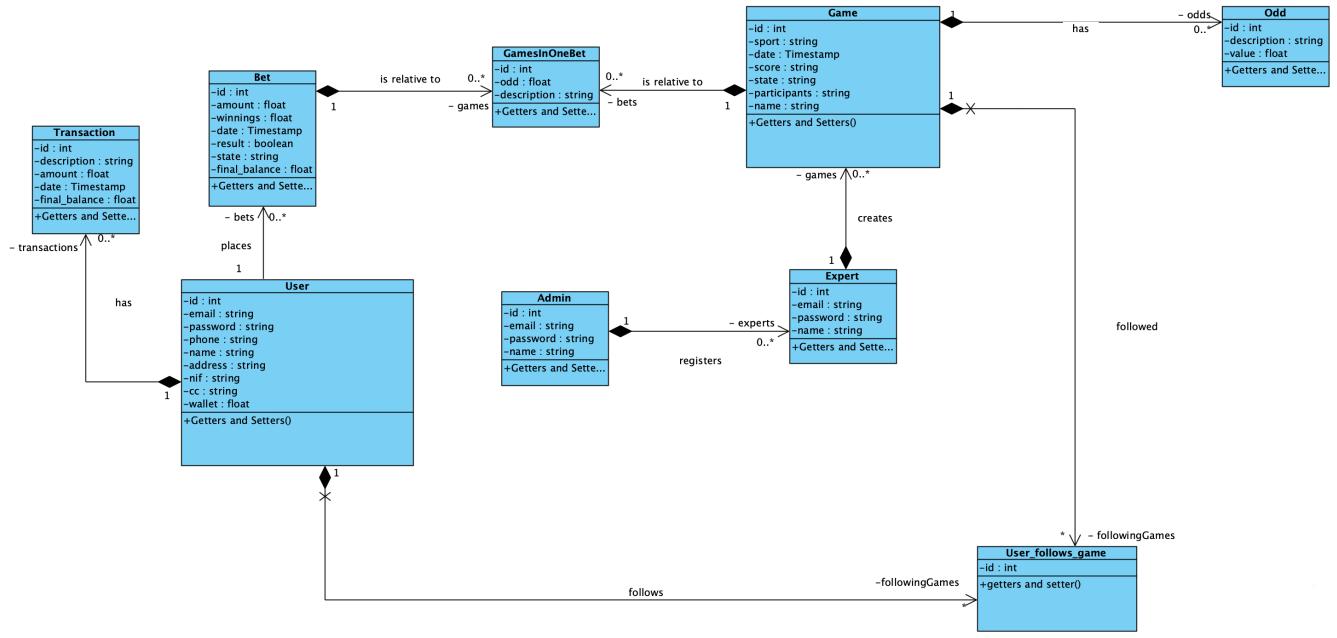


Figura 5.3: Diagrama Classes

5.4.1 Descrição das Classes

User

- É a classe que agrupa toda a informação necessária ao utilizador normal da aplicação
- Possui uma relação de 1 para muitos com a classe Transaction (1 utilizador pode fazer várias transações) e uma relação de 1 para muitos com a classe Bet (1 utilizador pode realizar várias bets)

Admin

- É a classe que contém a informação relevante para o administrador do sistema
- Esta tem uma relação de 1 para muitos com a classe Expert, ou seja, 1 administrador pode registar vários Experts

Expert

- É a classe que contém informação relevante para os experts do sistema
- Esta classe tem uma relação de 1 para muitos com a classe Games, já que 1 Expert pode criar vários jogos

Game

- É a classe que representa um evento desportivo e, por isso, agrupa toda a informação relevante à existência de um evento para a aplicação
- Esta possui 3 relações. A primeira com o Expert, já mencionada na descrição dessa classe. A segunda é uma relação de 1 para muitos com a Odd, já que um evento desportivo pode ter associado um certo número de odds, que os utilizadores poderão visualizar. Por fim, tem também uma relação de muitos para muitos com a classe Bet, já que 1 jogo poderá ter várias Apostas associadas a si, assim como uma Apostas pode conter vários jogos, caso seja uma aposta múltipla.

- **Odd**

- Representa a informação relativa a uma Odd, ou seja, um valor que representa quanto dinheiro o apostador ganhará se apostar 1 euro (se odd = 2, o apostador ganhará 2 euros se apostar 1 euro), e uma descrição, que indica a modalidade da odd (Empate, vitória de um dado participante, etc...)
- Esta possui uma relação de muitos para 1 com a classe Game, como referido na descrição da mesma.

- **Transaction**

- Contém toda a informação relativa a uma transação. Esta engloba o tipo de transação (depósito ou levantamento), método utilizado (cartão bancário ou PayPal), data em que foi realizada, bem como o saldo final após a transação.
- Como mencionado na descrição da classe User, esta possui uma relação de muitos para 1 com essa classe.

- **Bet**

- É a classe que possui informação relativa a uma aposta, ou seja, o montante apostado (amount), ganhos possíveis (winnings), data em que foi realizada, resultado da aposta, estado da aposta (se foi concluída ou ainda há jogos que estão por terminar) e o saldo final após a realização da mesma.
- Possui como já mencionado uma relação de muitos para muitos com a classe Game, explicada na descrição da mesma.

- **GamesInOneBet**

- É uma classe que tem como objetivo servir de tabela intermédia na relação entre Game e Bet. Esta possui informação relativa aos jogos que constam numa dada aposta, ou seja, se uma aposta for relativa a um jogo, a sua informação, valor da odd da aposta feita e descrição da mesma (Empate, Vitória de uma dada equipa, etc...), constará nesta classe.

- **UserFollowsGame**

- Contém a informação relativa à relação de um utilizador com o jogo, no sentido de o seguir. Esta armazena o id do jogo e o id do utilizador que o seguiu. Assim, esta tabela contém entradas relativas a todos os utilizadores que seguem jogos e todos os jogos que são seguidos por utilizadores.

5.5 Descrição dos Serviços da Aplicação

- AdminService
 - É o serviço onde são geridos os pedidos feitos por um administrador do sistema, tais como, criar um novo especialista, apagar um especialista anteriormente criado e visualizar todos os especialistas registados no sistema.
 - Métodos
 - * createExpert(String nome, String email, String password, Timestamp data_de_nascimento, String cc, String nif, String telefone, String morada)
 - * deleteExpert(int expert_id)
 - * getExperts()
- ExpertService
 - É aqui que são geridos o pedidos feitos por um especialista. Estes englobam a criação de um evento e o pedido de visualização dos jogos criados por especialistas.
 - Métodos
 - * createGame(String sport, List<String> participants, List<Float> odds, String eventName, Timestamp date, expert_email)
 - * createGame()
 - * createGameMotoGP(JSONNObject event, List<Game> games)
 - * createTwoParticipantEvent(JSONObject event, List<Game> games)
- UserService
 - É o serviço responsável por processar alguns dos pedidos feitos pelos utilizadores. Estes contém pedidos de login, registo, consulta de histórico de apostas e transações, mudança de informações do perfil, pedido de um código, recuperação da password e follow/unfollow de um jogo.
 - Métodos
 - * login(String email, String pass)
 - * register(String nome, String email, String password, Timestamp data_de_nascimento, String cc, String nif, String telefone, String morada)
 - * getBetHistory(String email)
 - * changeProfile(String name, String email_user)
 - * changeSensitive(String phone_num, String password, String new_add, String email_user)
 - * getCode(String email)
 - * getTransactionHistory(String email)
 - * recoverPassword(String email)
 - * checkAge(User user, String email_user)
 - * orderTransactions(JSONArray ja)
 - * recoverPasswordUser(User u, String email_user)
 - * recoverPasswordExpert(Expert e, String email_expert)
 - * followGame(String email_user, int id_game)
 - * unfollowGame(String email_user, int id_game)
 - * buildBet(Bet b)
 - * getUserTransactions(User u, JSONArray transactionsJsonList)

* getUserBets(User u, JSONArray transactionsJsonList

- **AppService**

- É o serviço responsável por processar alguns dos pedidos feitos pelos utilizadores e também por Experts, estes mais relacionados com a lógica aplicacional. Estes englobam a visualização dos jogos existentes na BD em que é possível apostar, realizar uma aposta, alterar uma odd de um dado jogo, inserir uma nova odd de um jogo e procurar jogos através de um filtro por nome de participantes. Possui também um método, updateStatus(), que tem como objetivo atualizar o estado dos jogos de uma forma periódica, fazendo, de forma colateral, a verificação do término das apostas realizadas pelos utilizadores da aplicação.

- Métodos

- * getGames()
- * placeBet(String user, String type, float amount, List bets)
- * changeOdd (int id, float odd)
- * insertOdd (int id, float odd)
- * getGamesFiltered(String participant)
- * updateStatus()
- * notifyObserver(Odd o)
- * placeBetMultiple(JSONArray bets, List<Game> games, JSONObject betslipForm)
- * placeBetSimple(JSONObject betslipForm, JSONArray bets, List<Game> games, List<Bet> betList, List<Odd> oddList)
- * suspendGamescheck()
- * updateBetStatus(Bet bet, List<GamesInOneBet> gamesInBet)
- * isGameFollowed(Game game, String email)
- * check_result(List<GamesInOneBet> giob)

- **TransactionService**

- É o serviço encarregue por fazer a lógica ligada às transações, ou seja, às operações de depósito e levantamento. Suporta o processamento de operações feitas por PayPal ou por serviços bancários.

- Métodos

- * transaction (String operation, int cardNum, int cardCCV, float operationValue, String email_user, String email)
- * deposit(JSONObject transactionsForm , User u)
- * withdraw(JSONObject transactionForm, User u)

- **EmailSenderService**

- É o serviço que tem como objetivo tratar do envio de mensagens email para os utilizadores. Estas são necessárias em vários cenários entre os quais o esquecimento da palavra passe, a notificação de término de uma aposta e o envio do código para a alteração de informações sensivas o perfil de utilizador.

- Métodos

- * sendSimpleEmail(String toEmail, String body, String subject)
- * betWonNotification(String email, float winnings)
- * betLostNotification(String email)

- * sensitiveInfoCode(String email, UUID code)
- * passwordRecovery(String email, String password)
- * updateOdd(String email)

- **RestService**

- É o serviço encarregue por fazer pedidos aos endpoints das APIs externas para obter informações sobre jogos. Neste momento estão apenas implementados pedidos à RasBet API, disponibilizada pela equipa docente e a uma API de jogos de basketball da NBA. Estes pedidos são feitos de forma periódica e só atualizam um jogo caso este não exista na BD ou, caso exista, se o seu resultado ainda não estiver atualizado.
- Métodos
 - * getGames()
 - * getGamesNBA()

5.6 Diagrama de Componentes

No diagrama de componentes apresentado pode visualizar-se as interações dos diferentes componentes que constituem o sistema na perspectiva da back-end. A back-end é constituída por quatro componentes principais, sendo que a front-end realiza a sua comunicação com a back-end através do componente "controller".

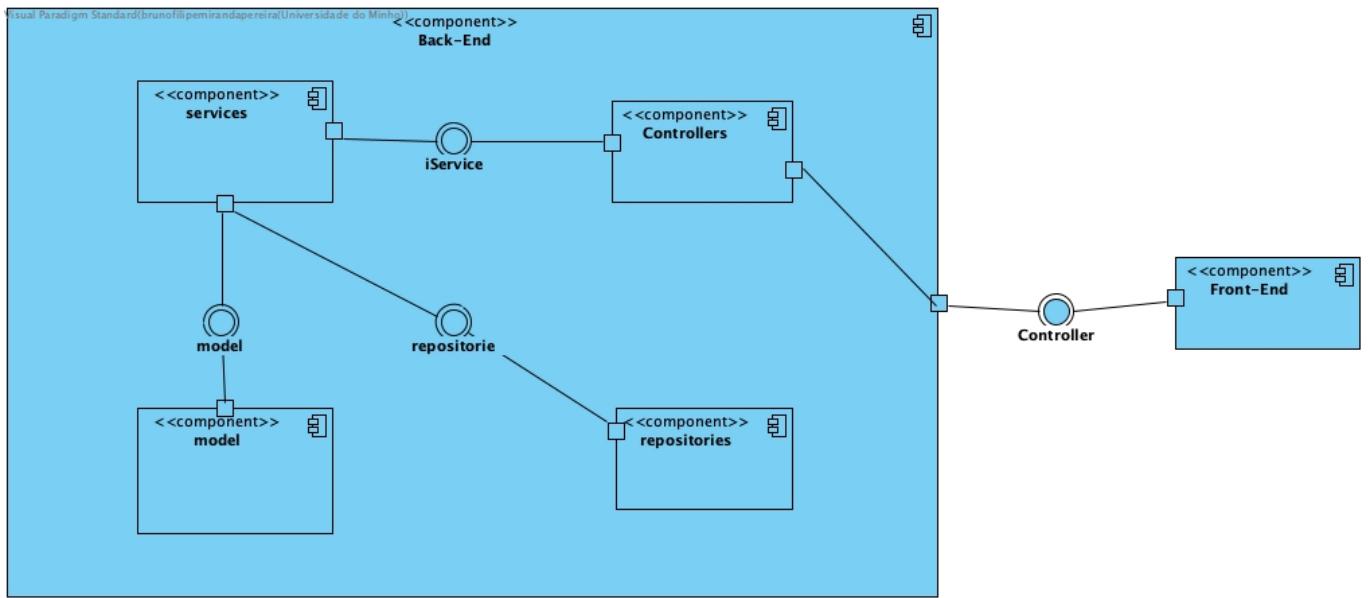


Figura 5.4: Diagrama de Componentes

5.7 Diagrama de Packages

No diagrama de packages pode ser observada a organização dos diferentes ficheiros de código. A organização do código do sistema da back-end é muito semelhante à organização dos diferentes componentes apresentados no diagrama de componentes.

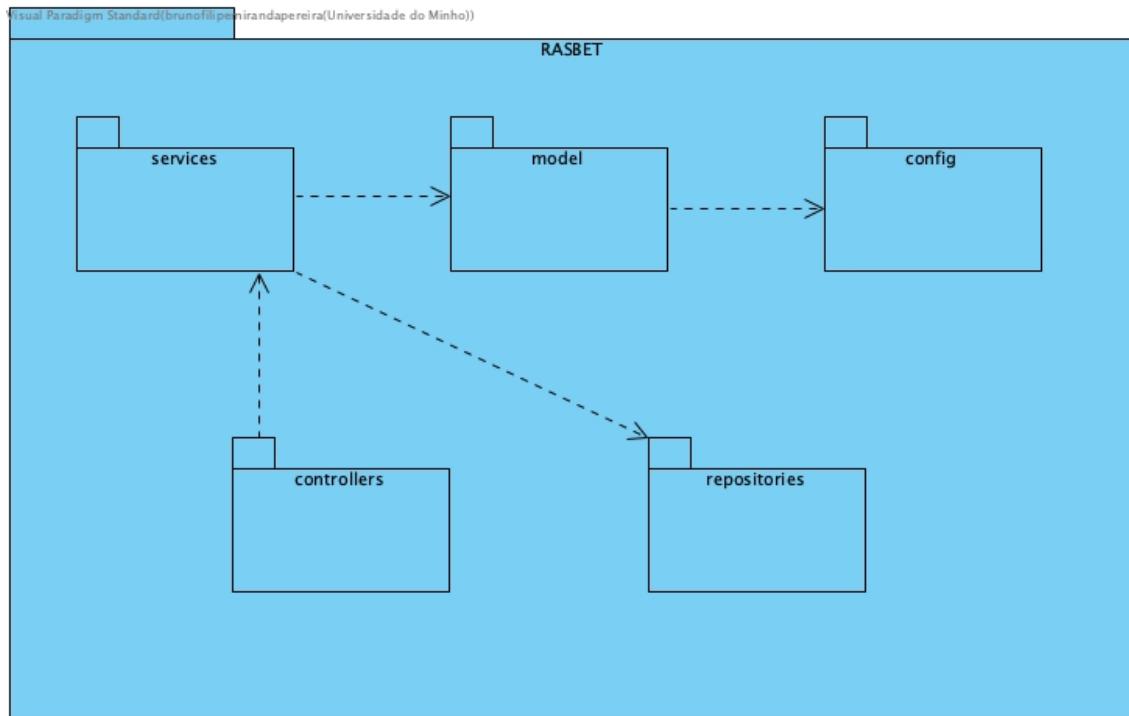


Figura 5.5: Diagrama de Packages

6. Runtime View

6.1 Diagramas de Sequência

Nesta secção são apresentados os diagramas de sequência dos métodos considerados mais importantes da Back-End, descrevendo o comportamento dos mesmos, estando estes já no domínio da solução, uma vez que referem componentes já integrados no sistema.

6.1.1 Consultar histórico de apostas e Estatísticas de Ganhos

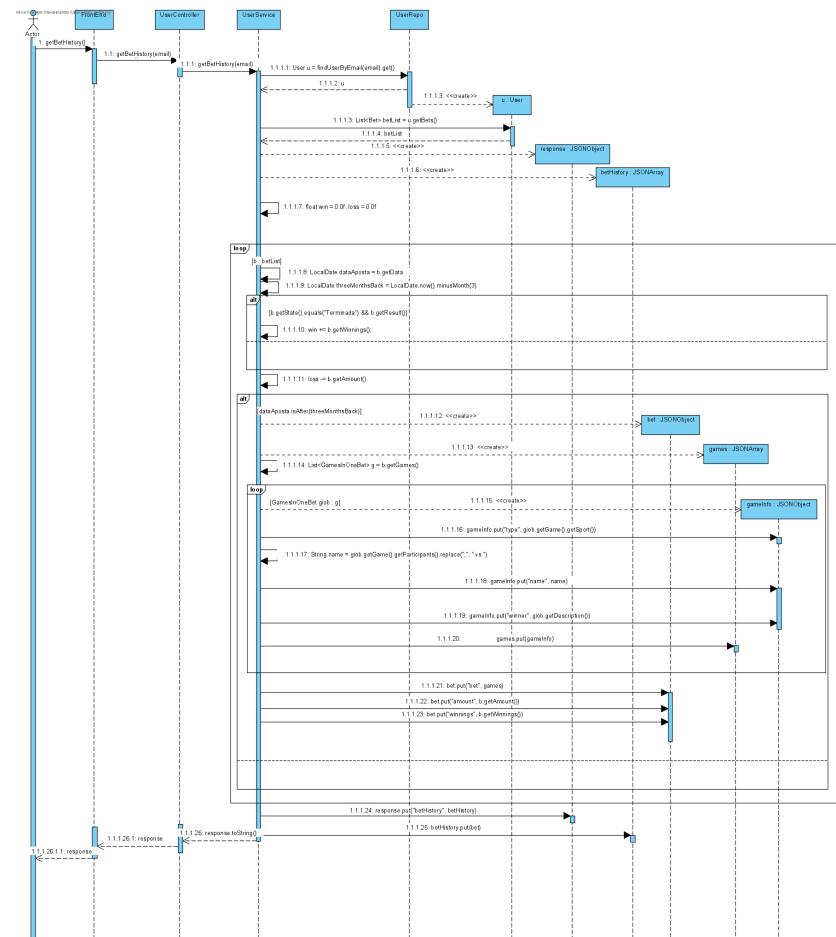


Figura 6.1: Diagrama de Sequência de consultar histórico de apostas e estatísticas de ganhos.

6.1.2 Fazer procura de jogos na API RASBet

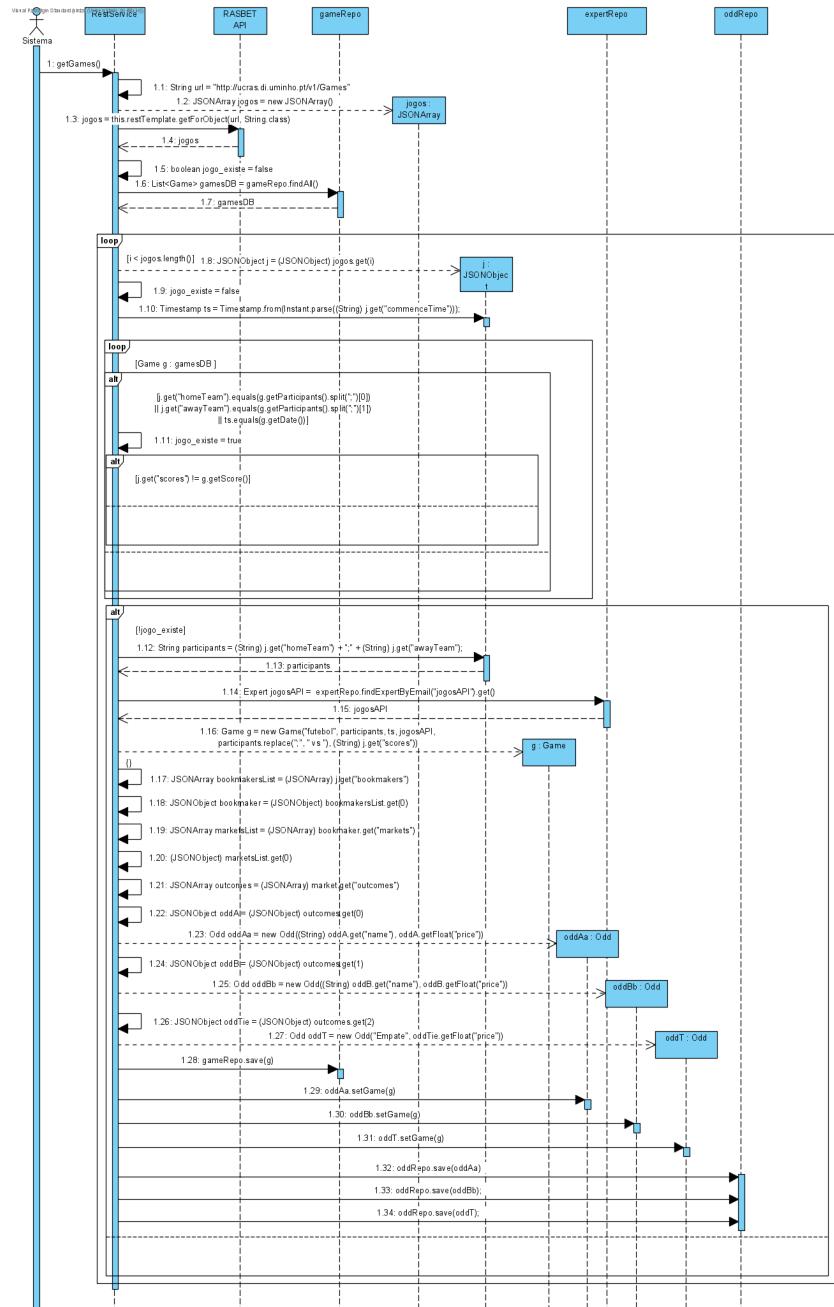


Figura 6.2: Diagrama de Sequência de fazer procura de jogos na API RASBet

6.1.3 Alterar estado de um jogo

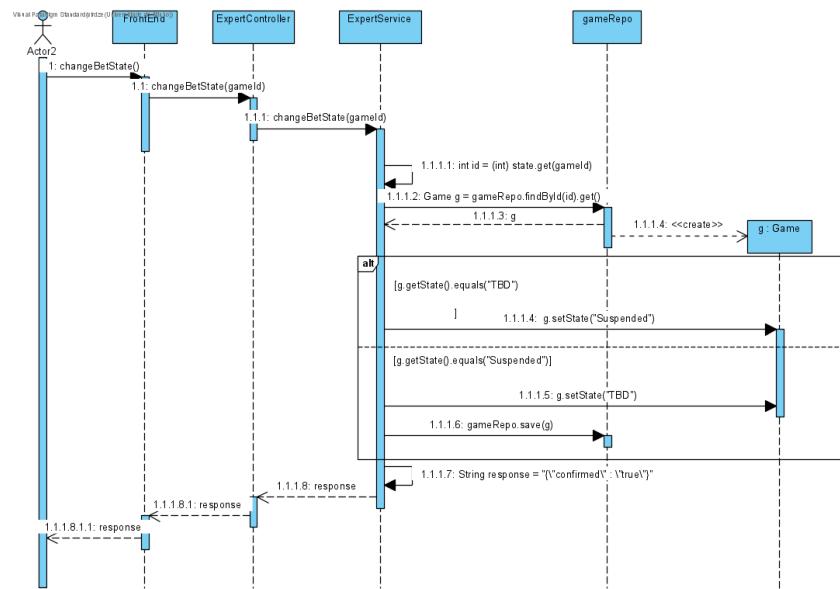


Figura 6.3: Diagrama de Sequência de alterar estado de um jogo

6.1.4 Fazer uma Aposta

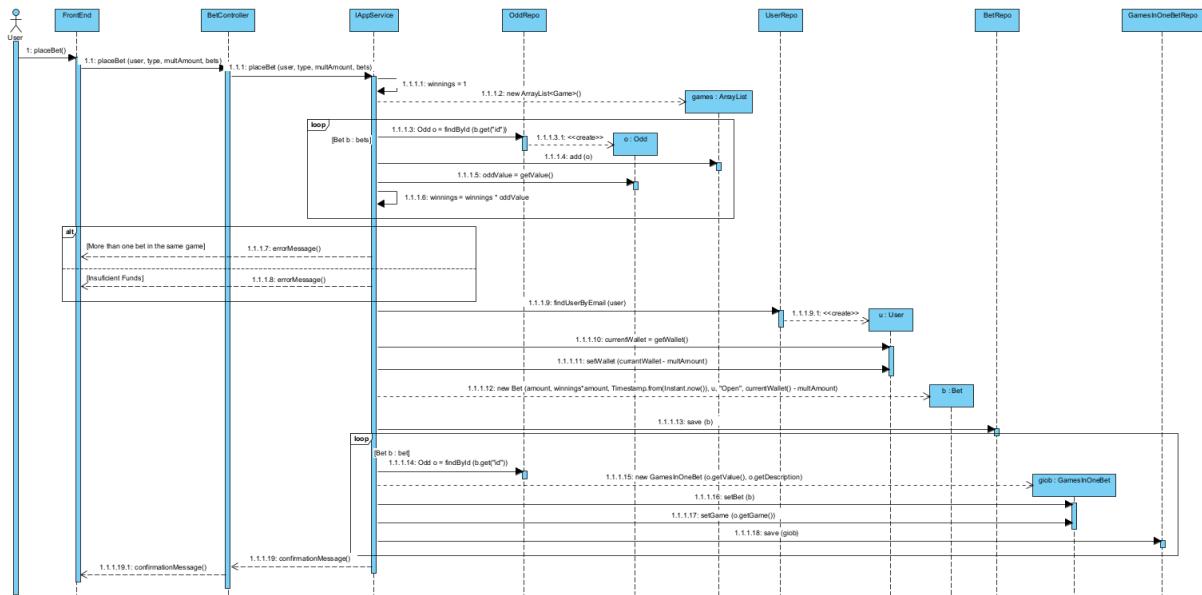


Figura 6.4: Diagrama de Sequência de realizar uma aposta.

6.1.5 Fazer um depósito

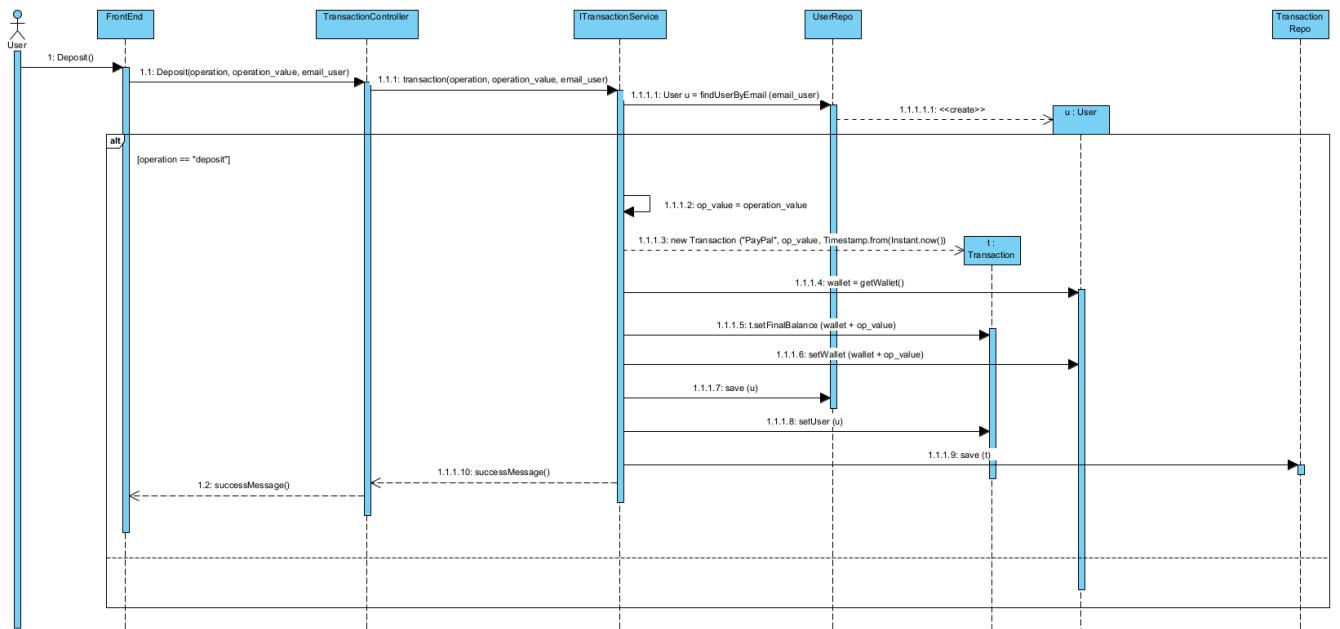


Figura 6.5: Diagrama de Sequência de realizar um depósito na conta.

6.1.6 Fazer um depósito

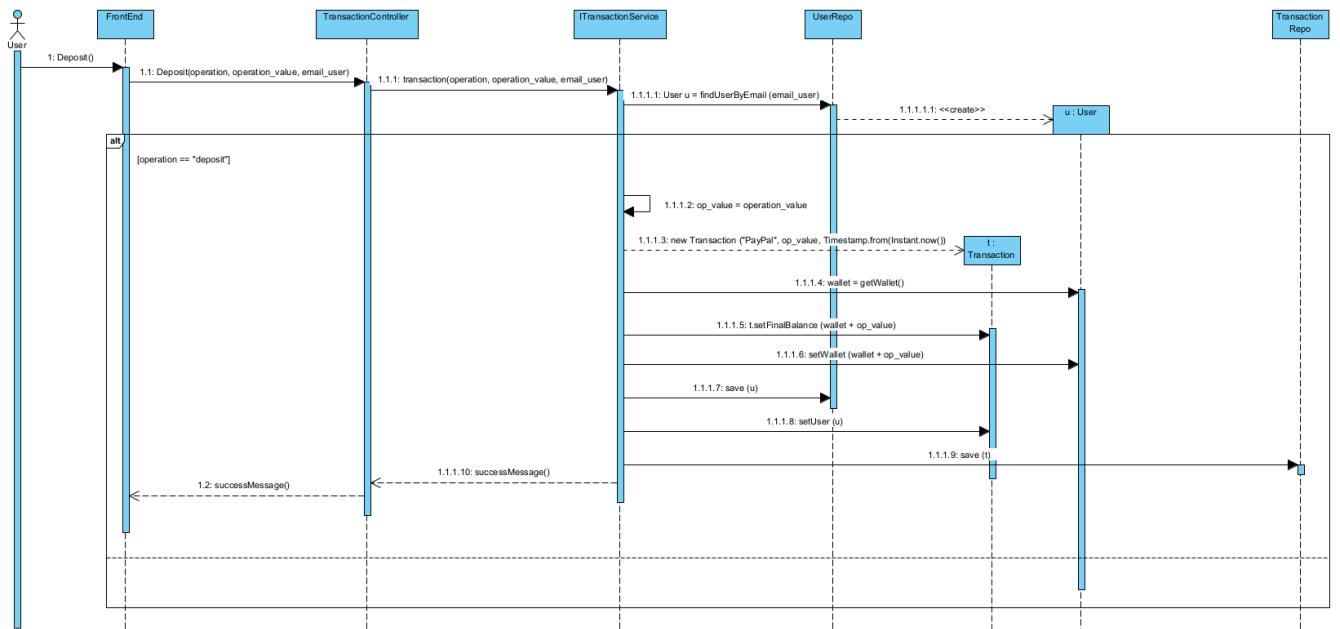


Figura 6.6: Diagrama de Sequência de realizar um depósito na conta.

6.1.7 Criar um jogo

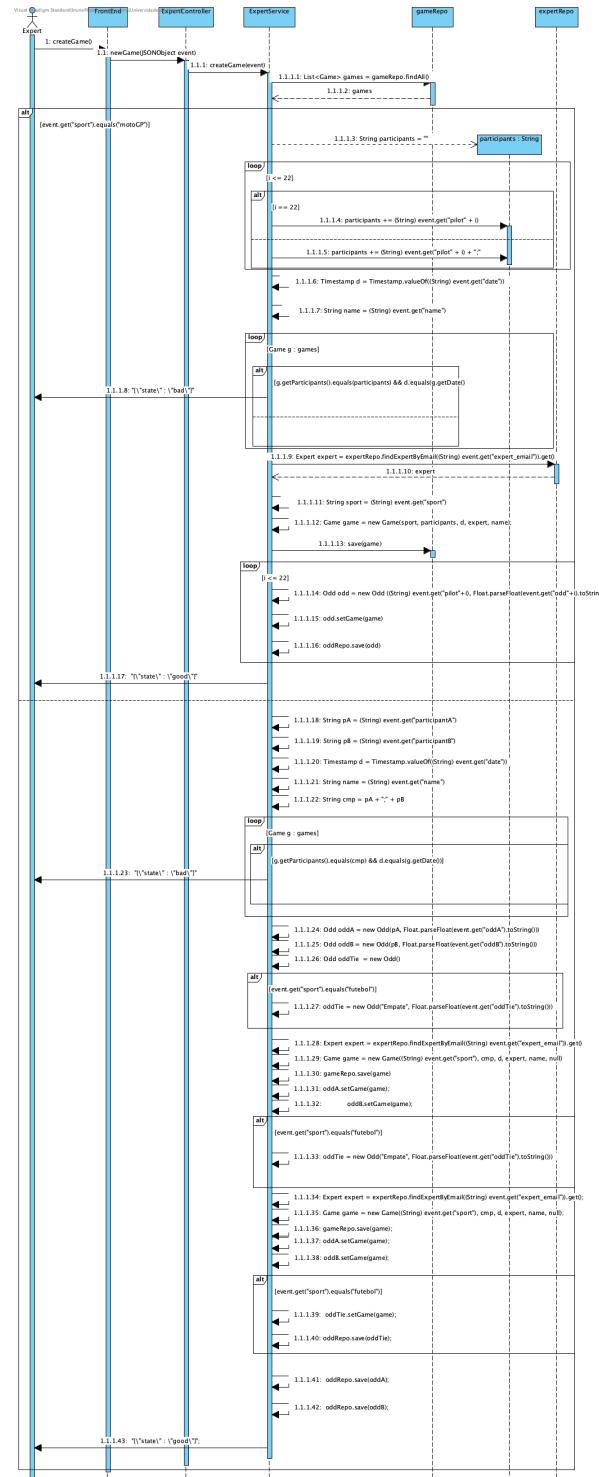


Figura 6.7: Diagrama de Sequência da criação de um jogo por parte de um especialista

7. Deployment View

Terminada a descrição da arquitetura do sistema e do funcionamento dos diversos componentes que ele constituem, é necessário ainda definir como será feito o deployment do sistema em cenário real de utilização. De seguida encontra-se um diagrama que detalha a distribuição do sistema numa versão ideal. Assim, de forma a correr a aplicação é necessário um computador cliente, com pelo menos 2 GB de RAM, um sistema operativo funcional e um Web Browser. Através do último componente, este irá poder comunicar com a aplicação, que estará replicada por 3 máquinas diferentes, todos a executar o mesmo código que irá atender os pedidos recebidos. Estes últimos, por sua vez, irão comunicar com a Base de Dados que estará geo-replicada. A DB1 será a máquina primária e terá a sua informação replicada e sincronizada pelas duas máquinas restantes. Desta forma, se uma Base de Dados for abaixo, os pedidos realizados pelas máquinas onde estará a correr a aplicação poderão ser redirecionados para as Bases de Dados secundárias.

7.1 Infraestrutura de Nível 1

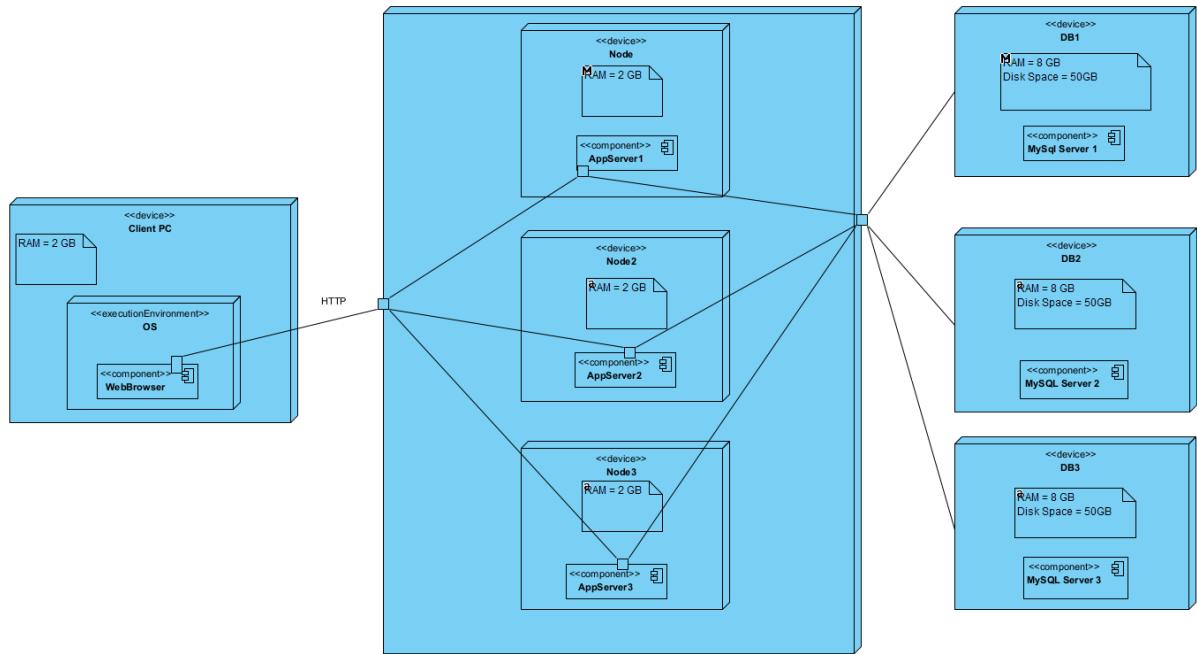


Figura 7.1: Diagrama de Deployment