



UMinho

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

RASBET

João Silva - PG50495
Duarte Lucas - PG50345
João Vieira - PG50501
José Magalhães - PG50528



((a)) José Magalhães



((b)) João Silva



((c)) Duarte Lucas



((d)) João Vieira

Braga, 20 de Janeiro de 2023

Conteúdo

1	Introdução e Objetivos	3
2	Restrições	4
3	Contexto e Âmbito do Sistema	5
3.1	Perspetiva de Domínio	5
3.2	Perspetiva Técnica	5
4	Estratégia da Solução	6
5	Building Block View	7
6	Runtime View	9
6.1	Registo	10
6.2	Bet	11
6.3	Login	12
6.4	Levantar Dinheiro	13
6.5	Depositar Dinheiro	14
6.6	CashOut	15
6.7	Criar Promoções e Editar Opções	16
6.8	Alterar as odds de um jogo	17
6.9	Alterar o estado de um jogo	18
7	Deployment View	19
8	Adição dos Novos Requisitos	19
9	Alterações Realizadas	20
9.1	Base de dados	20
9.2	Mudanças no Server	20
9.3	Apostas Múltiplas	20
9.4	Refactoring	20
9.4.1	Model View Controller	20
9.4.2	Facade	20
9.4.3	Observer	20

1 Introdução e Objetivos

A segunda fase do trabalho de Requisitos e Arquiteturas de Software, implica o seguimento da aplicação RASBET já inicializada na primeira fase, uma aplicação informática que permite a realização de apostas desportivas em relação a vários jogos de vários desportos. Nesta fase do projeto era requerido a implementação dos requisitos não funcionais apresentados no relatório entregue na primeira fase, focando-se mais no aprimoramento de alguns requisitos funcionais, e na implementação dos não funcionais.

Dos vários requisitos realizados, destacam-se a criação de uma frontend atrativa e de fácil entendimento,, capaz de realizar todas as funcionalidades impostas pelo projeto em questão:

- Registar - O sistema permite registar um apostador, um especialista e um administrador;
- Login - O sistema permite o login de um apostador, um especialista e um administrador;
- Apostar - O sistema permite o apostador realizar a aposta sobre o jogo que pretende e com a quantia desejada;
- Consultar jogos - O sistema permite o apostar consultar os jogos para uma possível aposta;
- Depositar e Levantar Dinheiro - O sistema permite ao utilizador depositar e levantar dinheiro;
- Histórico - O sistema permite ao utilizador consultar o seu histórico de apostas;
- Alterar Perfil - O sistema permite ao utilizador mudar o seu username, email e palavra-passe;
- Transferências - O sistema permite ao utilizador consultar todas as transferências realizadas, depósitos, levantamentos e apostas;
- Alterar odd - O sistema permite ao especialista alterar as odds de um jogo tendo em conta as odds de vários outras casas de apostas;
- Gerir Notificações - O sistema permite ao administrador gerir as notificações que irão aparecer na tela dos apostadores;
- Criar Promoções - O sistema permite ao administrador criar promoções, de modo a tornar a aplicação mais atrativa ao apostador;
- Gerir Jogos - O sistema permite ao administrador mudar o estado de um jogo, podendo o jogo estar no modo SLEEP, AWAIT e ACTIVE.

Tivemos o cuidado de implementar outros requisitos como por exemplo restrições de segurança (dados válidos,password forte). Ao longo do relatório serão apresentados vários diagramas que explicam todo o processo dos vários requisitos realizados.

2 Restrições

Para a realização desta fase do trabalho, encontramos algumas restrições relativamente ao projeto em questão.

Inicialmente, assumiu-se que para a frontend ia ser utilizada a biblioteca React em JavaScript e a backend, como já apresentada na fase anterior, em Python. Desta forma, com as linguagens e bibliotecas referidas foi possível desenvolver o Website capaz de comunicar com as funções e bases de dados da backend. O facto de nenhum dos elementos do grupo não ter nenhuma experiência com a frontend tornou o projeto mais desafiante e demorado, o que por sua vez implicou uma maior pesquisa sobre o mesmo.

Além disso, foram impostas restrições a nível técnico como também a nível temporal. O projeto deve ser capaz de compilar nos sistemas operativos mais conhecidos (Mac OS, Windows e Linux) e deve ser entregue(esta fase em questão) até dia 5 de Dezembro de 2022.

Tivemos que adaptar a aplicação RASBET de modo a conseguir fazer uso da API disponibilizada pelos docentes da disciplina, que por sua vez implica que o servidor tenha conectividade com a Internet, por forma a consultar e fazer uso da API externa RASBET API.

3 Contexto e Âmbito do Sistema

3.1 Perspetiva de Domínio

O utilizador deverá ser capaz interagir com o website de um modo fácil e simples, o sistema recebe e responde a todas as necessidades e funcionalidades disponibilizadas ao utilizador, é ainda capaz também de incorporar a API disponibilizada, RASBET API.

3.2 Perspetiva Técnica

A aplicação é capaz de receber três tipos de utilizadores (Cliente, Especialista e Administrador). O cliente é capaz de realizar apostas, depositar dinheiro, levantar dinheiro, realizar o cashout, entre outros. Já o especialista, tem como funcionalidade principal alterar as odds dos jogos, de forma a que o cliente possa apostar. Por fim, o administrador é capaz de realizar todas as tarefas de um especialista, porém pode também alterar o estado de um jogo.

Como já foi referido, a aplicação está separada em duas secções fundamentais, a frontend e a backend. Na primeira fase, as funcionalidades da aplicação foram apresentadas na linha de comandos, agora a aplicação é capaz de correr num browser numa página HTML.

Referente à backend, esta é constituída pela sua lógica de negócio(funcionalidades), tais como realizar uma aposta e levantar dinheiro, como também uma base de dados, onde são armazenadas todas as informações relativas a todos os utilizadores e a todos os jogos.

A aplicação conta também com o server, este é responsável por fornecer as páginas HTML ao utilizador.

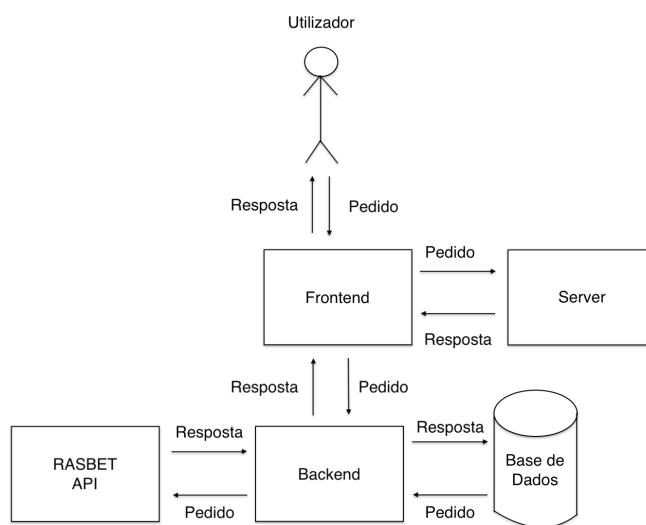


Figura 2: Diagrama de Arquitetura

4 Estratégia da Solução

Para a realização de qualquer projeto é necessária uma planificação organizada e bem estruturada. Assim sendo, começou-se por enumerar todas as etapas de forma a conseguir alcançar o resultado esperado.

Primeiramente, decidiu-se que a língua utilizada seria Python, devido ao facto da sua utilização ser bastante acessível. De seguida, começou-se por tratar da backend e os requisitos/funcionalidades obrigatórios, testando tudo no terminal. Como o projeto em questão recebe vários tipos de informação, utilizadores e jogos, também foi necessário criar bases de dados para cada classe, utilizando SQL.

As funcionalidades foram separadas de forma a gerir melhor o trabalho realizado. A primeira parte foi aceitar todos tipos de utilizadores, isto é, login e logout de clientes, especialistas e administradores. Após isso, e separadamente, adicionar as funcionalidades de cada tipo de utilizador, começando pelo utilizador, de seguida especialista e por fim administrador.

Por fim, desenvolvemos a frontend com a linguagem de programação React. React trata-se de uma biblioteca front-end em *JavaScript* em que o principal intuito é desenvolver uma interface em página web, tendo por base as mockups disponibilizadas no relatório da primeira fase do projeto. A frontend comunica com o servidor e com o backend através de pedidos HTTP.

5 Building Block View

A arquitetura desenvolvida do sistema começa a partir da identificação de 2 subsistemas, permitindo uma maior organização das classes e dos métodos usados. Deste modo conseguimos representar estes mesmo através dos diagramas de componentes seguintes:

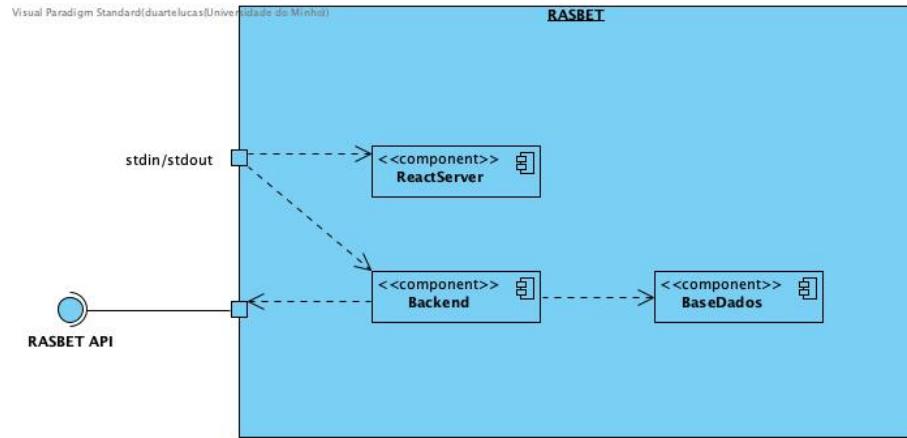


Figura 3: Diagrama de componentes (bloco de construção nível 1)

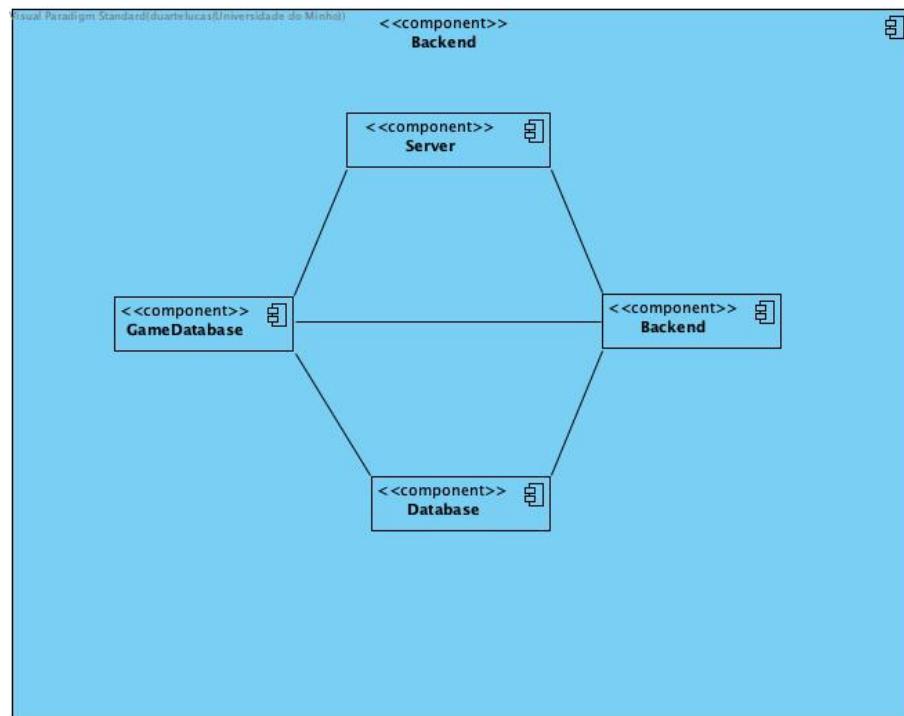


Figura 4: Diagrama de componentes (bloco de construção nível 2)

Como é possível verificar no diagrama de classes acima apresentado, todas as classes estão conectadas entre si.

Primeiramente, a classe database necessita de estar interligada a todas, pois é a base de dados de todo o projeto. Toda a informação existente irá ser armazenada aqui.

A classe Server é a responsável por inicializar a gameDatabase e a Backend. Além disso, é a classe server que recebe diretamente qualquer pedido vindo da frontend, e, com o auxílio das duas classes criadas primeiramente, lida e processa uma resposta para retornar à frontend.

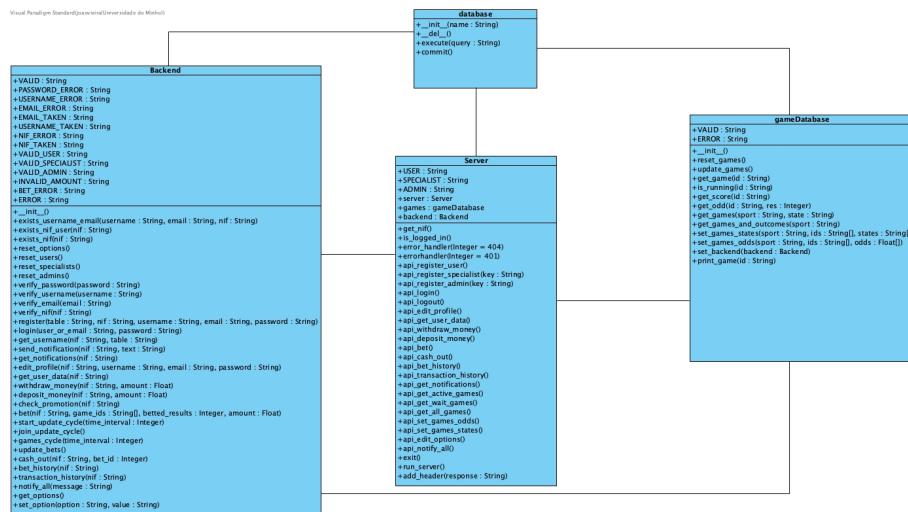


Figura 5: Diagrama de Classes

De seguida, a classe backend possui não só todas as funcionalidades relativas aos utilizadores como também cria as tabelas dos utilizadores, especialistas, administradores, notificações, apostas, entre outros. Quando é criada na classe server, necessita de receber uma base de dados para ser capaz de colocar todas as tabelas ditas anteriormente. A ligação existente entre a backend e a gameDatabase deve-se ao facto de, apesar das classes efetuarem tarefas diferentes, há certas funcionalidades na classe backend que dependem de funcionalidades que se encontram na gameDatabase, que é o caso da função bet(backend), que necessita de verificar se os jogos que vão ser apostados existem daí, na criação da backend esta precisar de receber a gameDatabase criada primeiro.

Por fim, a classe gameDatabase processa todos pedidos relativamente aos jogos desde o estado de cada jogo até à sua odd. Também é responsável por criar a tabela na base de dados dos jogos e dos outcomes.

Como é habitual nestes projetos, efetuou-se uma divisão do trabalho em si em diferentes componentes/subsistemas, de forma a simplificar a sua gestão e realização. Assim sendo, separou-se o trabalho em dois grupos fundamentais. Um grupo referente aos utilizadores e aos seus tipos(Cliente,Especialista e Administrador) e outro grupo referente aos bookmakers e jogos.

6 Runtime View

Para entender melhor a forma de como cada funcionalidade ocorre, foram realizados vários diagramas UML de sequência.

Todos os processos envolvem a classe frontend, que irá comunicar diretamente com o utilizador, e a server, que irá processar os pedidos e enviar para a backend ou para a gameDatabase, dependendo do pedido. A gameDatabase processa toda a informação relativamente aos jogos, as suas odds, as equipas e o estado do jogo. Por outro lado a backend efetua as funcionalidades relativas aos utilizadores, as apostas, levantamentos, login, entre outros.

Os diagramas em baixo apresentados foram realizados de forma a que seja possível compreender bem todo o seu processo, isto é, certas etapas de algumas funcionalidades foram excluídas, generalizando um pouco o seu diagrama para melhor entendimento.

6.1 Registo

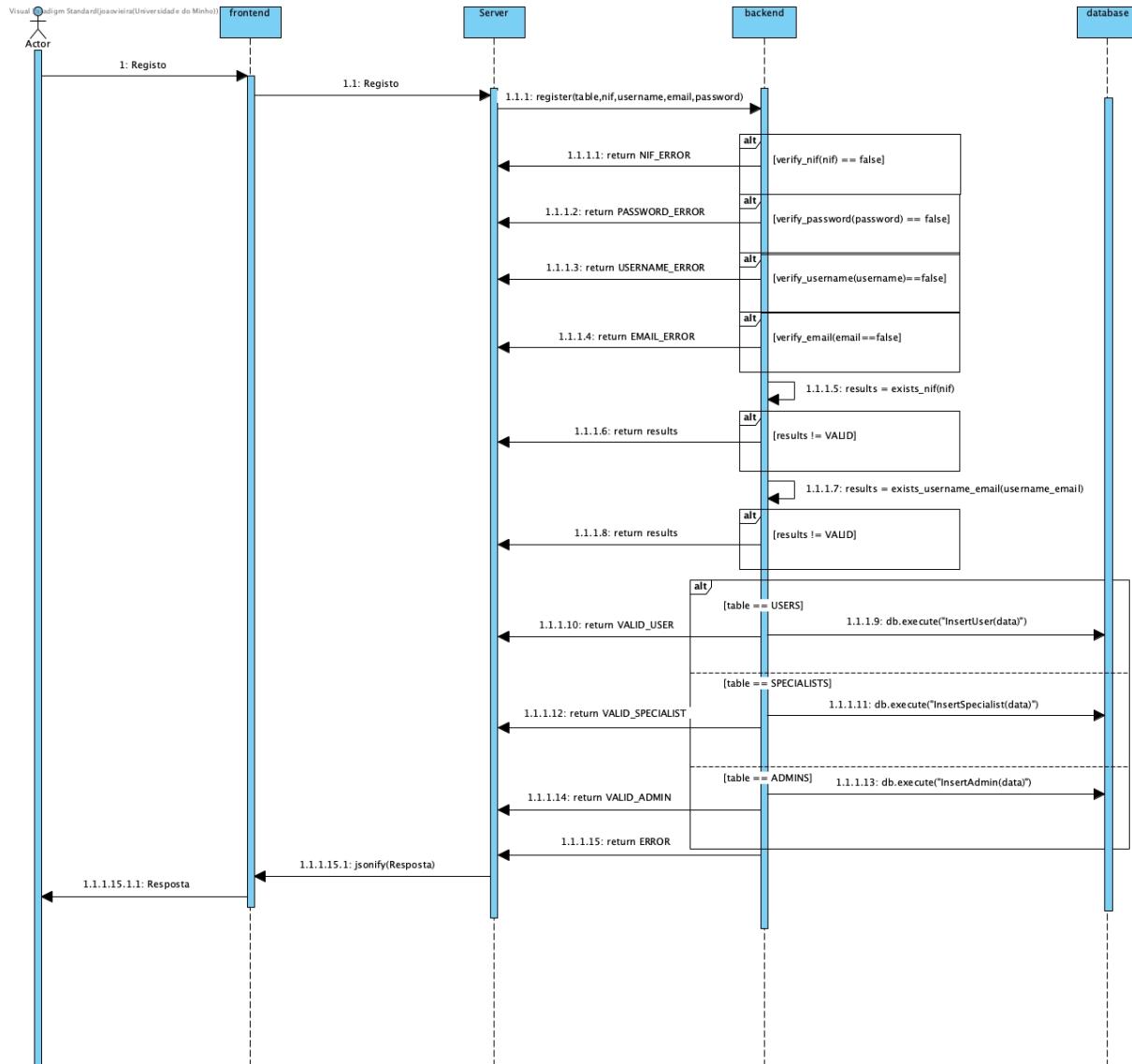


Figura 6: Registro

Para efetuar um registo, o utilizador deverá, como é possível verificar, fornecer todos os dados necessários e os mesmo necessitam de estar válidos, isto é, não podem existir nifs, nomes de utilizadores nem e-mails repetidos, a password tem de ser segura e o e-mail válido. Caso todos os dados sejam validados, o registo é efetuado com sucesso e o utilizador é inserido no sistema.

6.2 Bet

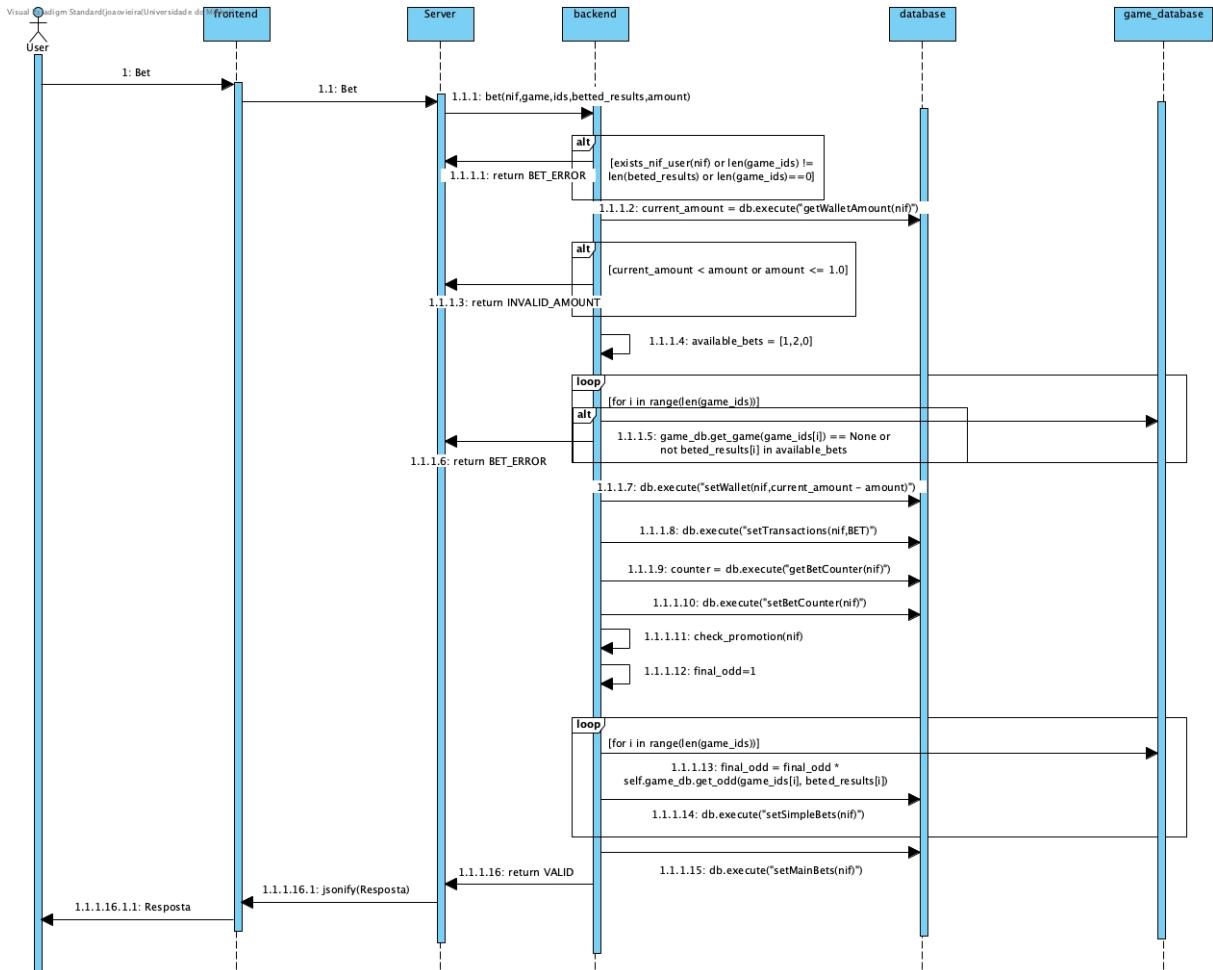


Figura 7: Bet

A funcionalidade aposta recebe quais os jogos que são para apostar, a quantia para apostar e o nif do utilizador. Primeiramente é verificado se os dados do utilizador são válidos e se há jogos para apostar, e de seguida é verificado se o utilizador em questão possui a quantia que quer apostar. Por fim, e em caso afirmativo é adicionada a aposta ao histórico de apostas do utilizador e apresentado o valor que pode receber caso ganhe.

6.3 Login

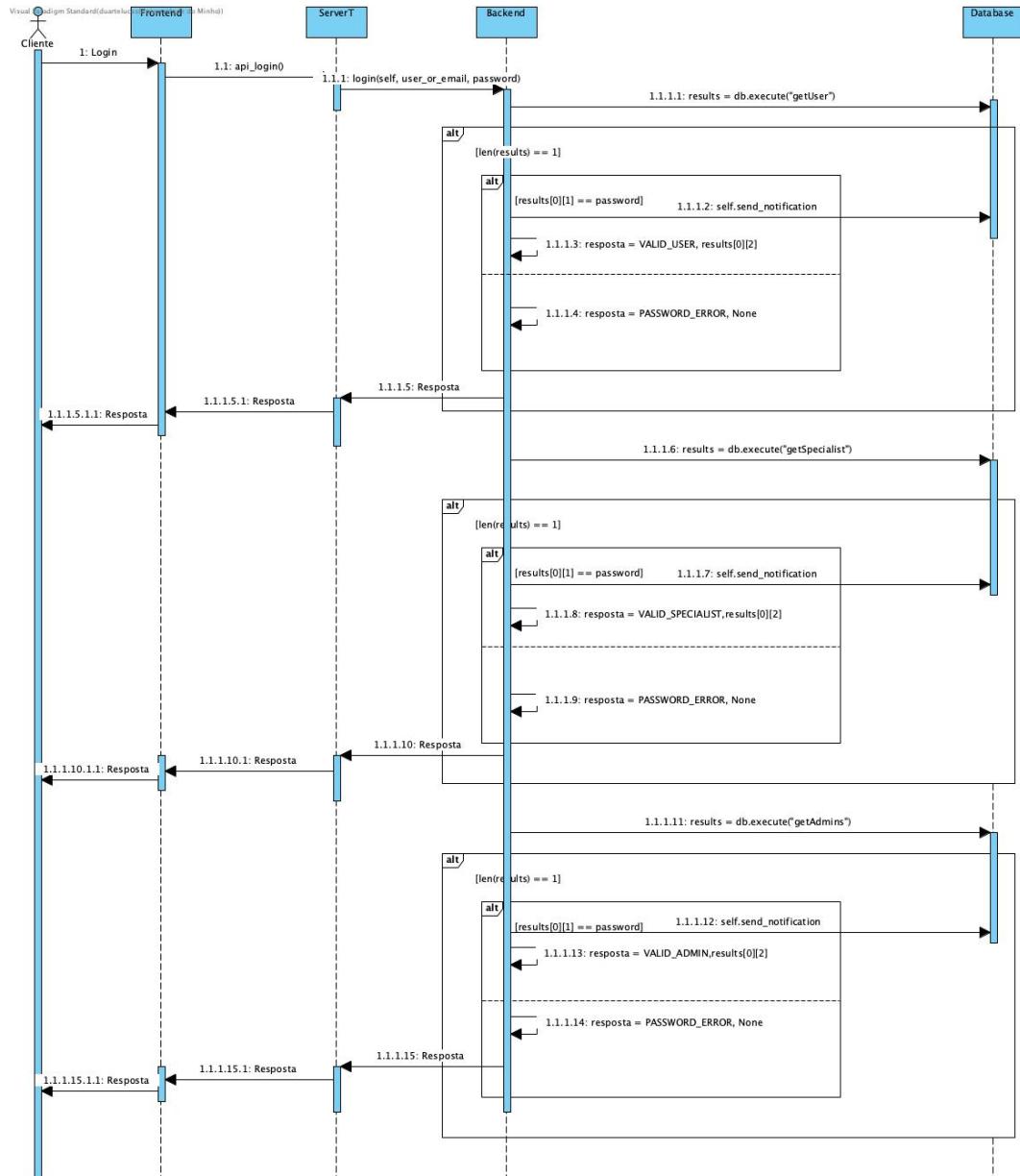


Figura 8: Login

A funcionalidade login permite a utilizador já criado entrar no website, caso o nome de utilizador não exista ou a palavra passe esteja errada, aparece um erro no ecrã. Os dados de um utilizador são sempre verificados nas três diferentes bases de dados: cliente, especialista ou administrador. Caso exista em alguma das bases de dados, o utilizador entra no site identificado como o seu tipo.

6.4 Levantar Dinheiro

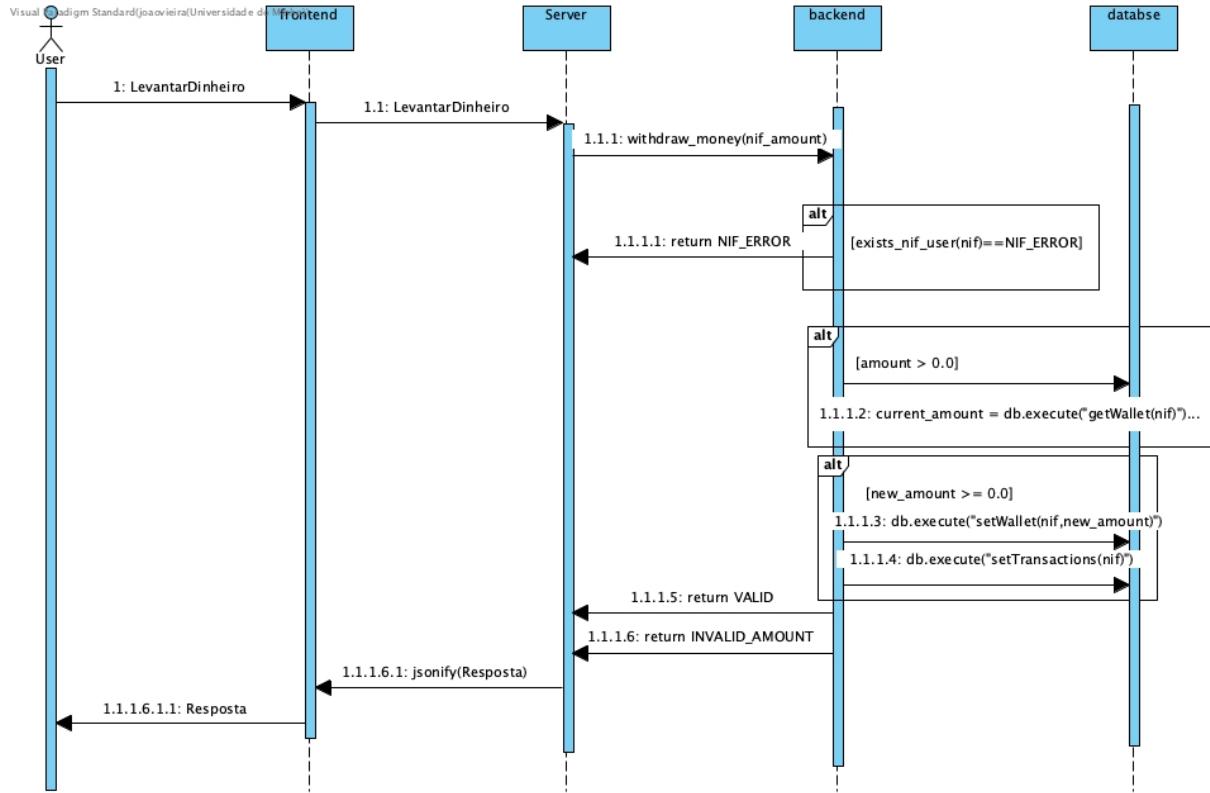


Figura 9: Levantamento

A funcionalidade levantar dinheiro permite a um utilizador levantar uma certa quantia. Primeiramente é verificado se o utilizador existe, através do nif. De seguida é verificado se o utilizador tem dinheiro e se possui a quantia que quer levantar. Em caso afirmativo, a quantia é levantada com sucesso, retirando o dinheiro levantado da wallet do utilizador.

6.5 Depositar Dinheiro

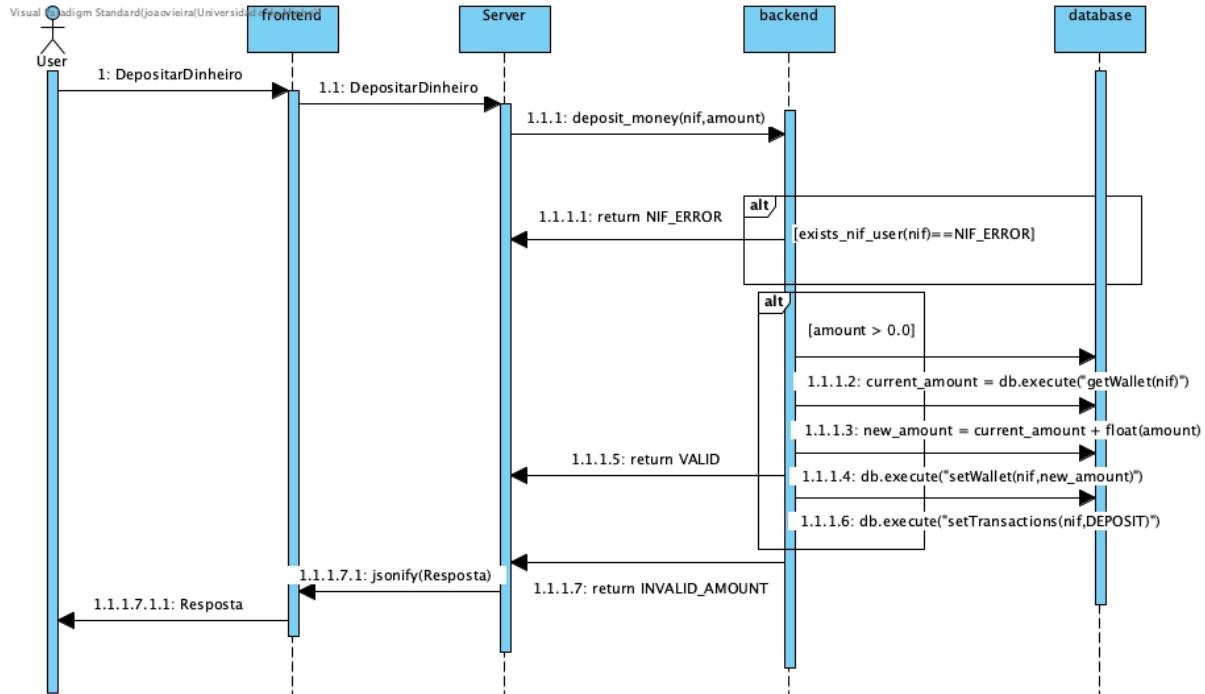


Figura 10: Depósito

A funcionalidade depositar dinheiro permite a um utilizador depositar dinheiro, desde que o nif dado à backend exista e o valor que o mesmo quer depositar seja superior a 0. Finalmente, é adicionada a quantia à conta do utilizador em questão.

6.6 CashOut

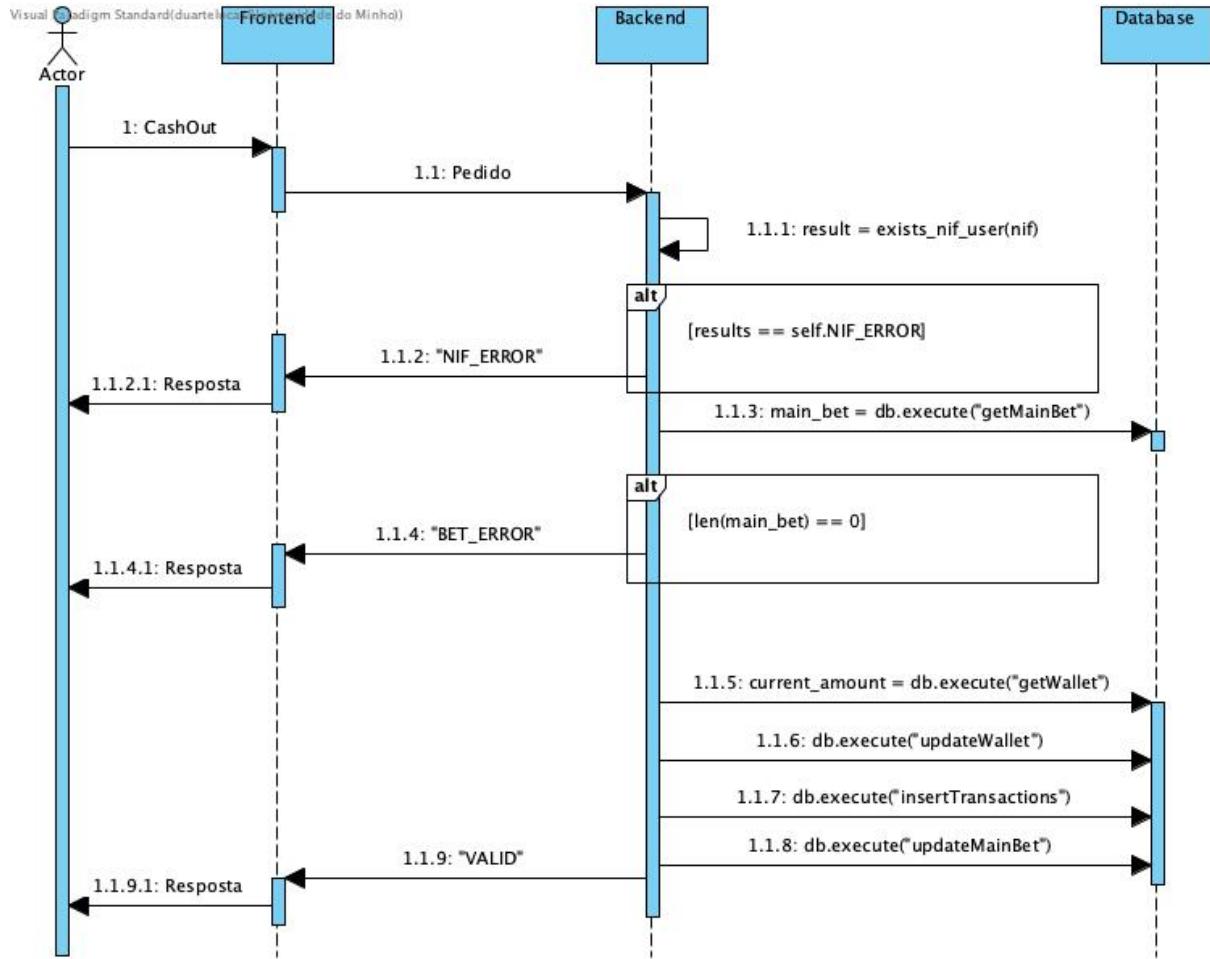


Figura 11: CashOut

A funcionalidade CashOut permite a um utilizador retornar parte do dinheiro de uma aposta que realizou ainda inacabada. Em primeiro lugar é verificado se o nif é válido e de seguida se o mesmo tem apostas realizadas. Em caso afirmativo, a aposta é cancelada e metade do dinheiro é retornado.

6.7 Criar Promoções e Editar Opções

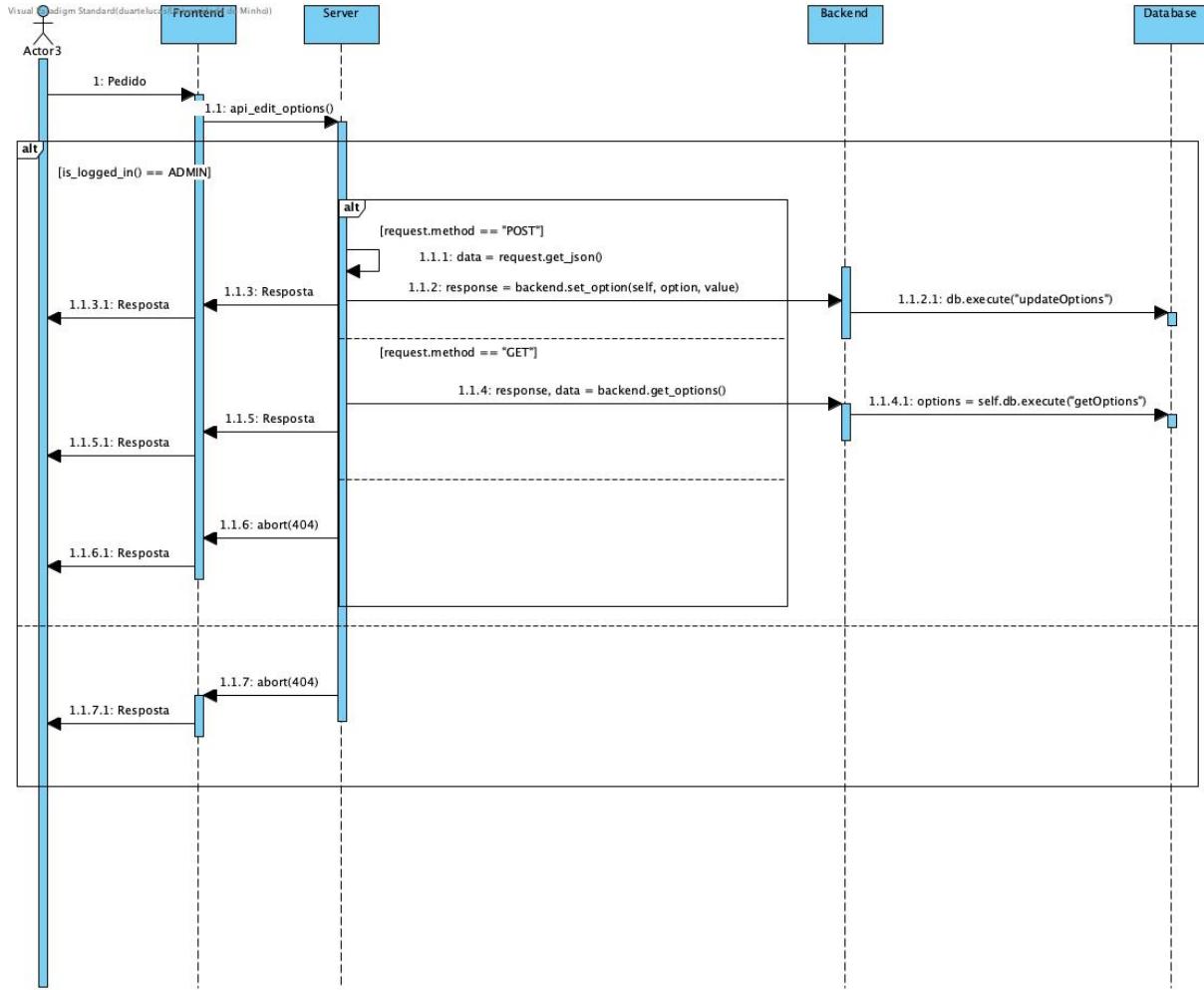


Figura 12: CriarPromoções

A funcionalidade em questão permite tanto criar promoções como editar opções de sistema. É uma funcionalidade apenas permitida aos administradores, e torna-o capaz de criar promoções, como diz o nome, ativar e desativar notificações, entre outros.

6.8 Alterar as odds de um jogo

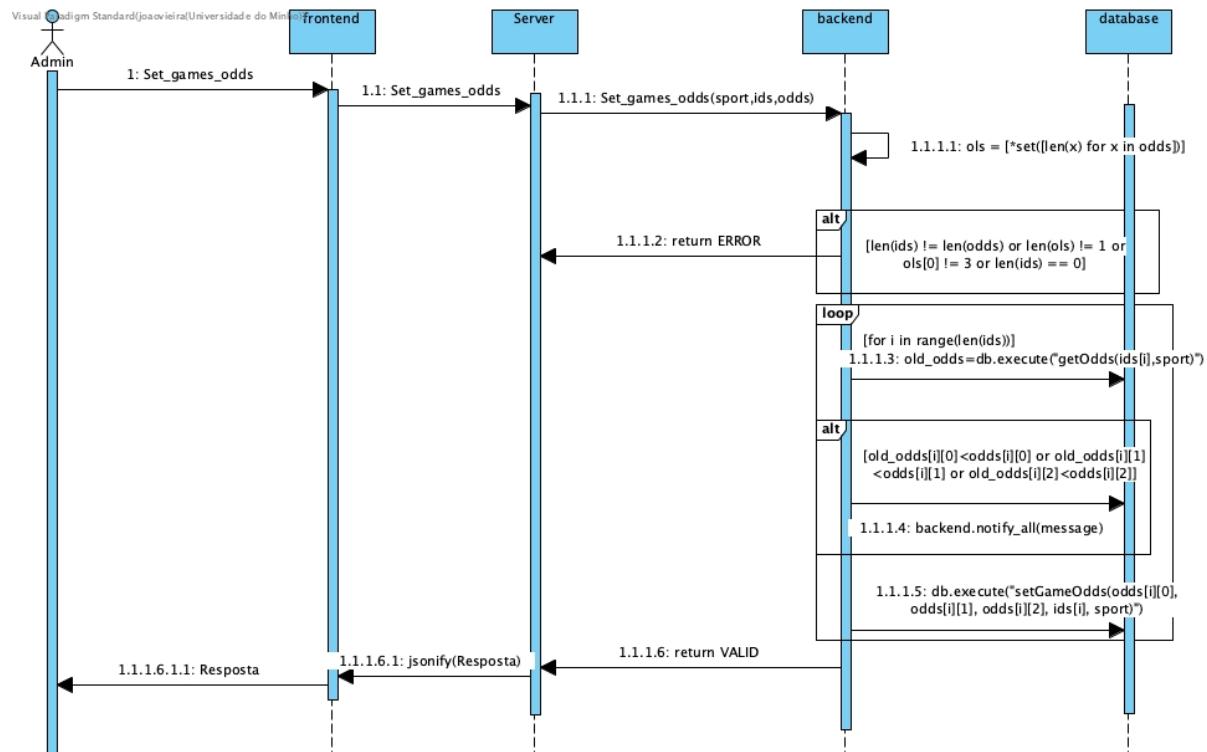


Figura 13: SetGameOdds

A funcionalidade de alterar as odds de um jogo é permitida aos especialistas e aos administradores. Caso o valor atualizado de uma odd seja superior ao valor antigo, é enviada a todos os clientes uma notificação a informar.

6.9 Alterar o estado de um jogo

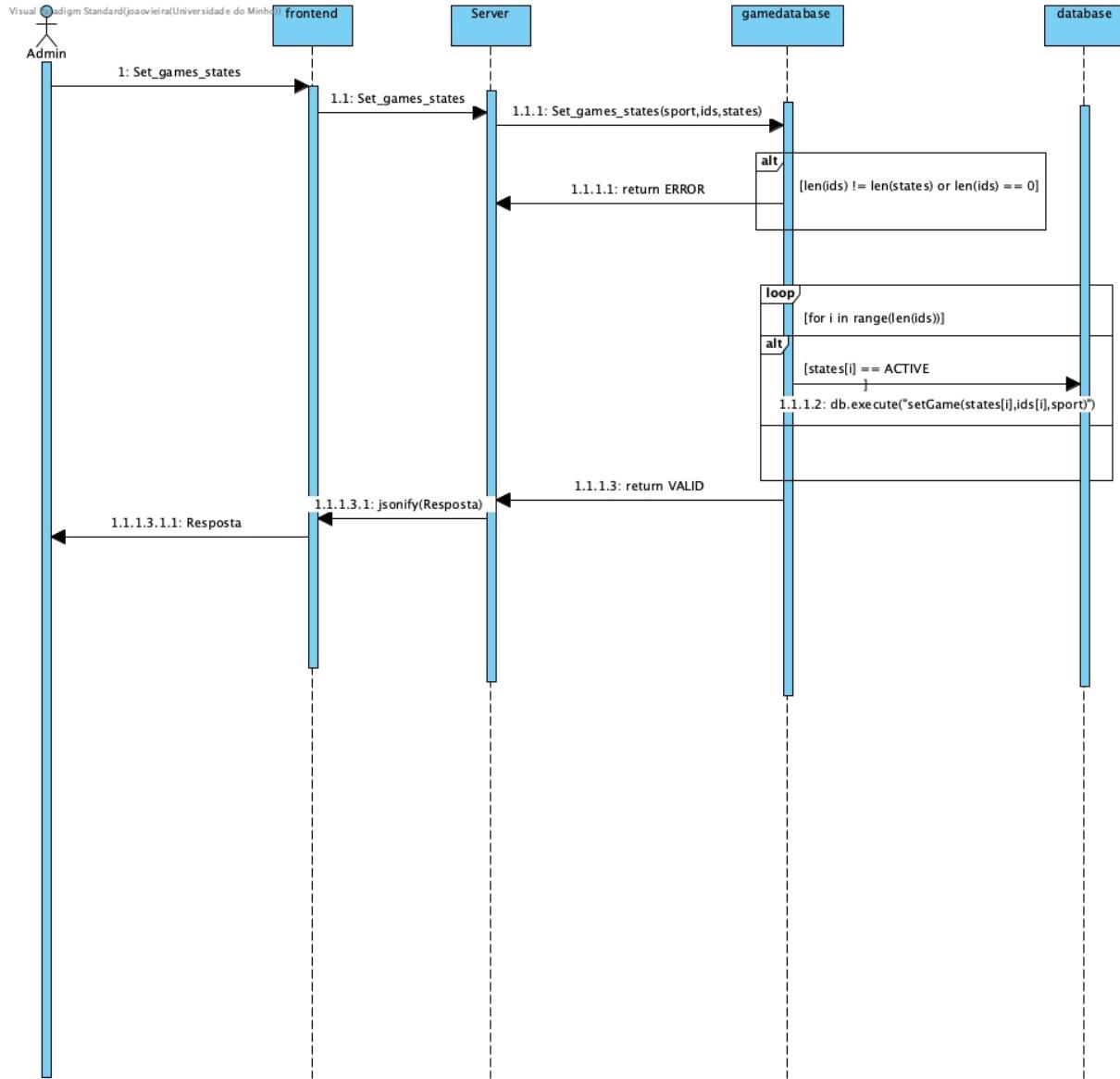


Figura 14: SetGameState

A funcionalidade de alterar o estado de um jogo é semelhante à de alterar as odds de um jogo. São igualmente apenas permitidas aos especialistas e administradores e, caso um jogo esteja ativo, altera o estado dele para o que o utilizador que o está a alterar assim pretender.

7 Deployment View

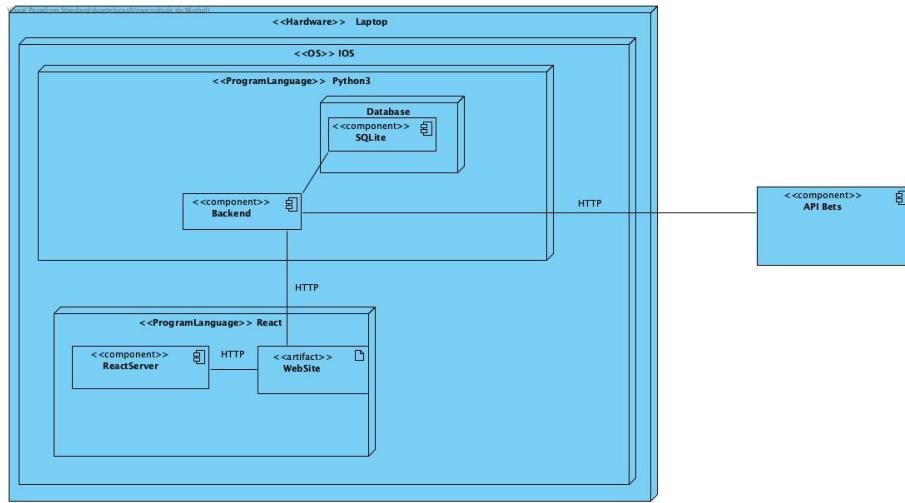


Figura 15: Diagrama de Implementação

Com o diagrama de implementação conseguimos perceber que o nodo Website comunica com o nodo React-Server e com o nodo da Backend através de pedidos HTTP. A Backend além de responder a pedidos vindos do nodo Website, comunica com o nodo API através de pedidos HTTP e comunica com o nodo da Database. Como exemplificado no diagrama, a aplicação desenvolvida apenas corre localmente.

De modo a que o programa execute é necessário que a backend seja corrida com python3 e que a frontend-react corra com as funcionalidades comando npm.

8 Adição dos Novos Requisitos

Nesta fase, a equipa docente pediu ao grupo de trabalho que modificasse o código desenvolvido de modo a que aplicação permita a adição de novas funcionalidades, tendo sido pedido que fossem implementados 2 novos requisitos, apostas múltiplas e interessados em seguir jogos.

- Apostas múltiplas: Como já referido, este requisito já tinha sido implementado na fase 2 deste projeto, tendo sido assim apenas necessário adaptar e reformular pequenos aspectos. Anteriormente, qualquer apostador era capaz de fazer um apostas múltiplas sem qualquer limitação em relação ao número de jogos apostados. Deste modo e para cumprir com o desejado, o apostador foi limitado de forma a apenas realizar apostas múltiplas com no máximo 20 jogos.
- Interessados em seguir jogos: Esta funcionalidade exigiu um pouco mais de trabalho em relação ao que anteriormente fora feito. Na entrega da última fase já tinha sido implementado um requisito que notifica todos os utilizadores sobre a subida de uma odd de qualquer jogo. De modo a cumprir com o pedido, foi necessário o desenvolvimento de código que permita ao utilizador *"seguir"* jogos, e adaptar o código responsável pelas notificações para que a aplicação notificasse qualquer alteração de odd, quer essa alteração seja um subida ou descida de odd, aos apostadores que tinham *"seguido"* o jogo ou que tinham apostado naquele jogo.

9 Alterações Realizadas

9.1 Base de dados

Relativamente às bases de dados, foi adicionada uma nova tabela com chaves estrangeiras denominada GAME_FOLLOW tem como propósito ligar os jogos e os seus seguidores.

9.2 Mudanças no Server

Para a questão de seguir ou não seguir jogos, foram adicionados três end-points do tipo post ao servidor, sendo eles follow_game, unfollow_game e check_follows_game que têm como objetivo seguir um jogo, deixar de seguir um jogo e verificar se um jogo está a ser seguido, respectivamente.

Assim sendo, as funções supracitadas foram também adicionadas à classe User, assim como a notify_about_game, com o intuito de ser chamada na backend sempre que a função set_games_odds é invocada.

9.3 Apostas Múltiplas

Por último, a funcionalidade que permitia a existência de apostas múltiplas já tinha sido implementada pelo grupo na fase anterior, apenas foi necessário adicionar a restrição que permite somente um número máximo de 20 jogos por aposta múltipla.

9.4 Refactoring

Nesta última fase do trabalho o grupo optou por refazer completamente a estrutura do código de modo a tornar a solução implementada muito mais fácil de modificar e/ou expandir as suas funcionalidades.

9.4.1 Model View Controller

Assim o primeiro aspeto a ser considerado foi pelo grupo foi qual a solução arquitetural a ser implementada tendo sido escolhido o padrão MVC sendo este muito semelhante à arquitetura que já tinha sido implementada nas fases anteriores.

Este padrão é constituído por 3 componentes principais, a view (representada pela frontend) que permite ao utilizador interagir com o sistema, o controller (representado pelo servidor http) e o model (representado pela backend e todas as classes que lhe estão associadas).

9.4.2 Facade

Este padrão é utilizado no que diz respeito à backend e as suas classes proporcionando uma interface de acesso única por parte do servidor. Isto permite encapsular toda a backend separando-a do servidor http e criar um ponto de acesso único ao modelo de dados.

9.4.3 Observer

Este padrão de design foi implementado para lidar com as notificações dos utilizadores quando as odds de um jogo subscrito pelo utilizador são atualizadas permitindo que este seja notificado quando tal acontece. Nesta situação os observadores são os utilizadores e os observados são os jogos subscritos.

Todas estas modificações permitem expandir as funcionalidades do código de maneira mais fácil e rápida, como por exemplo adicionar novos tipos de bets, moedas, desportos ou jogos.

10 Conclusão

Com a realização deste trabalho foi possível consolidar toda a matéria lecionada nas aulas teóricas e práticas, uma vez que nos permitiu estudar e praticar temas como "*design patterns*", análise de cada requisito quer funcional ou não funcional, entre outros, lecionados ao longo da unidade curricular.

Concluímos também que este trabalho prático constitui uma mais valia para o nosso conhecimento visto que permitiu adquirir boas bases para trabalhos futuros, quer académicos ou profissionais. O projeto desenvolvido exigiu do grupo vários momentos de reflexão e pesquisa devido ao contexto que o projeto se encontrava e foi apresentado.

Por fim, o grupo acredita que apresenta uma boa resolução que satisfaz de um modo gracioso todos os requisitos funcionais e não funcionais propostos. Por outro lado, sabemos que o trabalho desenvolvido poderá ser melhorado pois a nossa falta de experiência em React e alguma falta de tempo limitou um pouco o nosso produto final.