UMinho

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

RASBET: Solução Arquitetural

Anabela Pereira PG49995
Fernando Lobo PG50002
Joana Alves PG50457
Vicente Moreira PG50799

Braga, 4 de dezembro de 2022

Conteúdo

Lista de Figuras

1. Introdução

Um documento de requisitos é responsável pelo domínio do **problema**, isto é, como o próprio nome indica, tratar dos requisitos que o sistema tem de cumprir para alcançar com sucesso o objetivo final. No entanto, um documento de requisitos é muitas vezes completado com um documento contendo a **solução arquitetural** do sistema. Este último trata o domínio da **solução**. Com a junção destes, obtemos uma ideia concreta e exata do que o sistema deverá ser capaz de fazer assim como os passos necessários para tal. Assim, este documento é uma continuação do documento de requisitos desenvolvido na primeira entrega, tendo por objetivo apresentar a solução arquitetural do sistema RasBet.

1.1 Contextualização

O presente documento de requisitos tem como objetivo especificar o conjuntos das funcionalidades do sistema RasBet a ser desenvolvido. Este sistema, de forma geral, é caracterizado por um website de apostas desportivas, onde possíveis clientes poderão efetuar apostas em vários eventos desportivos baseados nas informações disponíveis acerca desse evento.
Neste documento vão ser apresentadas várias secções, todas elas ligadas de alguma forma à solução arquitetural do sistema, como, por exemplo, diagramas comportamentais, diagramas de *deployment*, entre outros. Assim, começamos por apresentar um breve resumo dos requisitos funcionais e não funcionais mais importantes do sistema, presentes no documento de requisitos.

1.2 Requisitos Funcionais

Para que o sistema a desenvolver seja implementado com sucesso, assim como garantir a sua longevidade e operabilidade, o produto deverá disponibilizar registo de novos utilizadores, assim como a autenticação dos mesmos. Além disso, tem de ser capaz de apresentar uma listagem de eventos desportivos, eventos esses que poderão ser alvo de apostas por parte dos atores apostadores com registo. Os vários utilizadores poderão filtrar essa listagem de eventos por competição ou modalidade. No caso dos apostadores com registo, estes terão acesso a um histórico de apostas realizadas, assim como histórico de transações monetárias e balanço respetivo. Por último, sempre que um jogo termina, todos os apostadores que realizaram apostas sobre o mesmo devem ser notificados do resultado da aposta e da respetiva quantia creditada.

1.3 Requisitos Não Funcionais

Relativamente aos requisitos não funcionais, apresentamos aqueles que demonstram possuir um caráter importante no desenvolvimento do sistema. Para fácil correspondência, estes vão ser apresentados em conjunto com o número identificador presente no documento de requisitos.

- (7) - O utilizador tem acesso rápido às funcionalidades.
- (17) - O sistema deverá estar operacional 97% do tempo.
- (22) - O sistema deverá permitir escalabilidade.
- (30) - Encriptação de informação sensível dos utilizadores.

2. Restrições do Sistema

Este capítulo apresenta, para cada secção, os requisitos que de alguma forma restringiram o *design* ou as decisões de implementação da solução ou que têm presença obrigatória na solução.

2.1 RASBet API

Uma das restrições impostas pelos docentes trata a utilização obrigatória de uma API desenvolvida pelos mesmos, a RASBet API. Esta API, ao ser contactada, devolve uma listagem de jogos da modalidade Futebol, contendo toda a informação dos mesmos, como o nome das equipas, a data, o resultado (em caso de término), entre outros.
No entanto, esta restrição tem como consequência limitações técnicas não só ao nível de rede, como de performance do serviço, pois não só requer que o sistema esteja conectado à rede da Universidade do Minho (seja de forma direta, seja através da utilização de uma VPN), como também esta API possui uma taxa de atraso de resposta elevadíssimo, o que traz desafios relativos à velocidade de apresentação do conteúdo.

2.2 Requisitos de Sistema

- O sistema deve apresentar uma lista de eventos desportivos.
- O sistema deve apresentar uma lista de modalidades disponíveis.
- O sistema deve apresentar uma lista de competições das várias modalidades.
- O sistema, para cada evento desportivo, deve apresentar uma lista de resultados possíveis, apresentando a sua *odd* respetiva.
- O sistema permite realizar apostas.
- O sistema permite o registo de novos utilizadores.
- O sistema deve permitir a autenticação de utilizadores já registados no sistema.
- O sistema deve permitir a desautenticação dos utilizadores autenticados.

2.3 Requisitos de Apostador com Registo

- O sistema deve permitir a edição de dados pessoais do apostador.
- O sistema deve apresentar o histórico de apostas do apostador.
- O sistema deve apresentar o histórico de transações do apostador.
- O sistema deve permitir o levantamento e depósito de fundos na conta do apostador.

2.4 Requisitos de Especialista

- O sistema deve apresentar as modalidades, competições e eventos desportivos a que o especialista está autorizado a inserir *odds*.
- O sistema deve permitir a gestão de *odds* por parte do especialista.

2.5 Requisitos de Administrador

- O sistema deve permitir a gestão de apostas, jogos, notificações, promoções e utilizadores do sistema ao administrador.
- O sistema deve permitir a criação de contas de Especialista e Administrador por parte do administrador.

3. Contexto do Sistema

Nesta secção apresentamos um diagrama de contexto com o intuito de demonstrar a comunicação do sistema com os sistemas vizinhos (API RASBet fornecida pelos docentes) e utilizadores do sistema (apostadores com registo, apostadores sem registo, especialistas e administradores).

3.1 Diagrama de Contexto

Como referido acima, apresentamos um diagrama de contexto para evidenciar o alcance do sistema a ser construído. Este diagrama foi elaborado com base numa generalização do diagrama de *Use Cases* construído no documento de requisitos. Para além disto, o diagrama sintetiza todos as funcionalidades obrigatórias referidas no capítulo anterior.

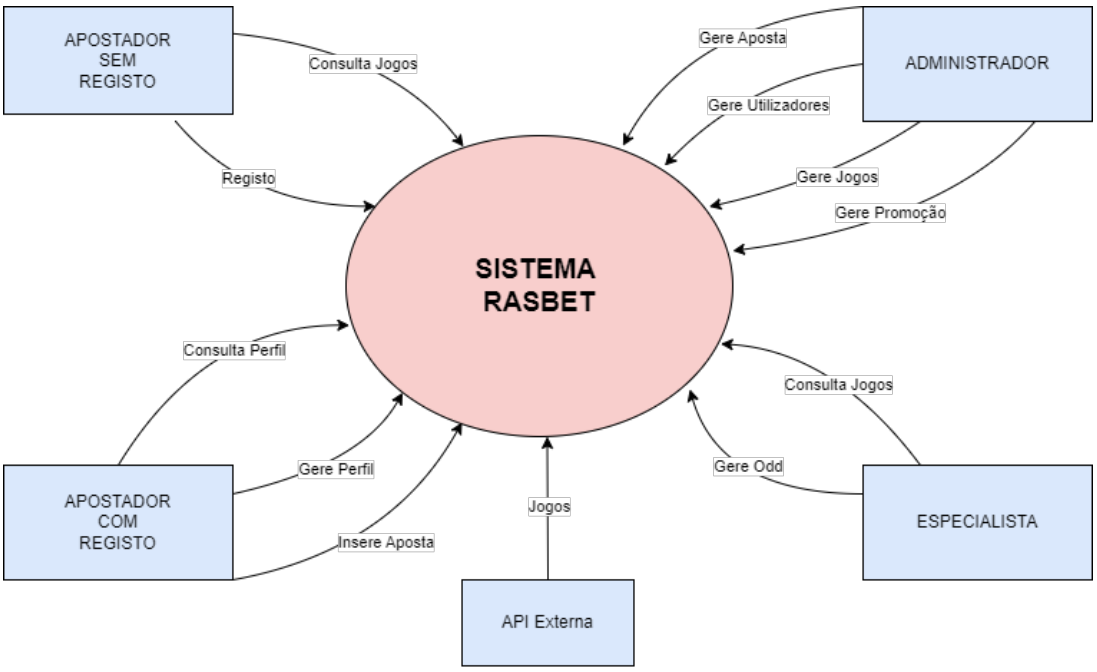


Figura 3.1: Diagrama de Contexto do Sistema.

4. Estratégia da Solução

Nesta secção serão apresentadas todas as estratégias de solução que deram forma à arquitetura final do sistema RasBet, incluindo tecnologias utilizadas, decomposição *top-level*, estratégias para atingir atributos de qualidade e decisões organizacionais relevantes.

4.1 Decisões Organizacionais

No que toca à organização da equipa para a execução deste projeto, a metodologia de desenvolvimento utilizada foi a ***Agile***. Esta metodologia incentiva o desenvolvimento e teste contínuos ao longo do desenvolvimento de um projeto. Ao contrário da metodologia *Waterfall*, a metodologia *Agile* permite desenvolver e testar em paralelo. Ao utilizá-la, economiza-se muito tempo otimizando as tarefas e evitando os erros que podem ocorrer durante as etapas de planeamento. Assim, os benefícios da utilização desta metodologia são vários:

- Rapidez e agilidade de desenvolvimento;
- Aumento da satisfação do cliente ao conseguir lidar com mudanças de última hora;
- Aumento da produtividade da equipa ao encorajar a partilha das suas ideias, fazendo estes sentirem-se valiosos e úteis;
- Probabilidade mínima de reestruturação do projeto na entrega final, uma vez que se vai adaptando às necessidades do cliente ao longo do desenvolvimento.

4.2 Atributos de Qualidade

Relativamente aos atributos de qualidade, escolhemos apresentar apenas os que definimos como requisitos não funcionais **chave** do sistema, isto é, os requisitos apresentados na secção 1.3. Assim, passamos a apresentá-los, mais uma vez identificados através do seu número identificador do documento de requisitos, e a explicar, de forma sucinta, como os implementamos.

Requisito	**Solução Arquitetural**
7	As várias funcionalidades do site podem ser acedidas em menos três cliques.
17	O site vai estar *offline*, no máximo, 12 dias por ano para atualizações.
22	Ao implementar uma arquitetura multi-camada, permitimos o aumento e diminuição dos recursos nas diversas camadas de forma independente.
30	Palavras-passe dos utilizadores são armazenadas com encriptação na base de dados. Identificador do utilizador acedido e testado através de encriptação (*token*).

4.3 Padrão Arquitetural

Visto que um dos requisitos não funcionais mais importantes refere que o sistema deverá ser dividido em três camadas principais e permitir escalabilidade, a equipa passou a investigar uma arquitetura que possibilitasse isso mesmo.
Decidimos assim que, relativamente ao padrão arquitetural adotado, a aplicação irá seguir uma arquitetura **multi-camada**. Neste tipo de arquitetura, os componentes do sistema estão organizados como camadas independentes entre si, cada uma com um cargo diferente. Com cada camada a agir como cliente da camada seguinte, temos como consequência uma modularização das mesmas, o que permite o *deployment* e a escalabilidade independente.

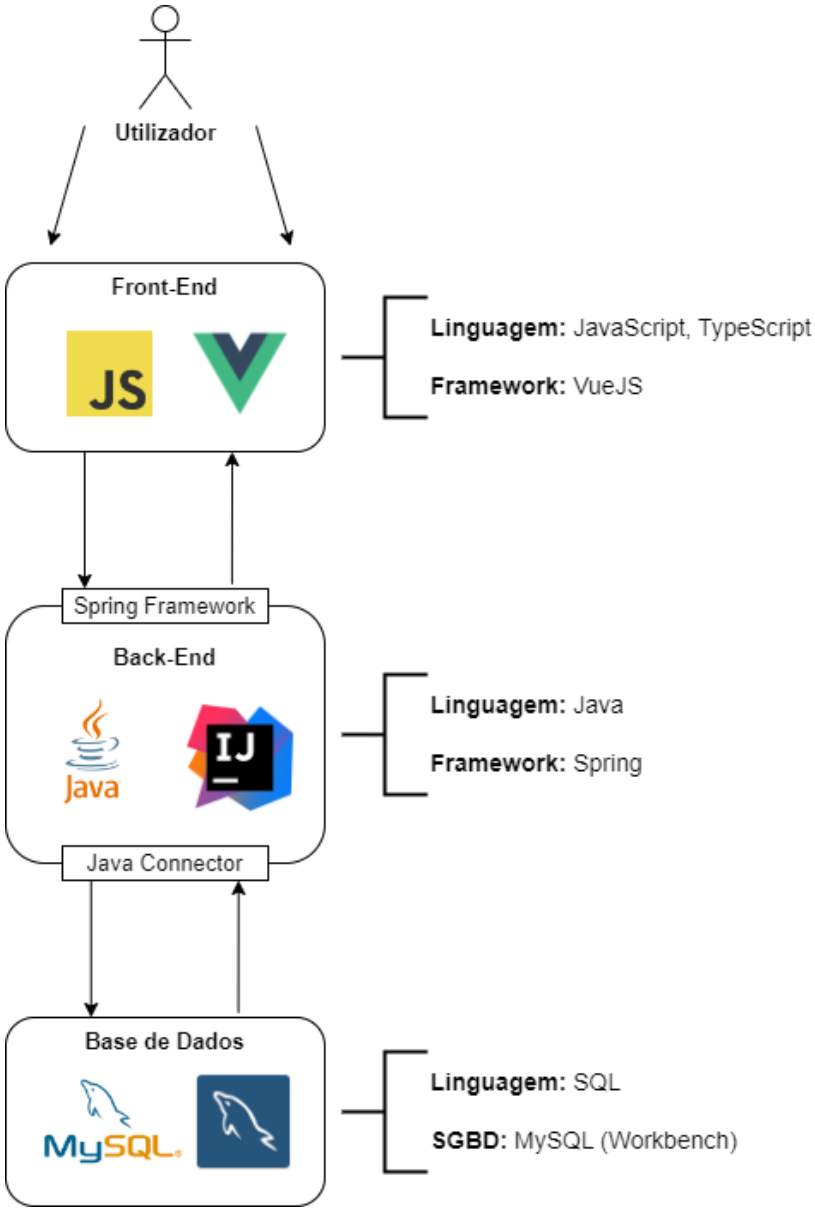


Figura 4.1: Arquitetura do Sistema.

4.4 Tecnologias

A partir da figura acima, conseguimos denotar algumas tecnologias utilizadas na construção da solução. Estas foram retiradas a partir dos requisitos não funcionais presentes no documento de requisitos. Deste modo, passamos a enumerar por extenso, as várias tecnologias utilizadas em cada nível do sistema.

4.4.1 Interface Gráfica

Na camada da interface gráfica, ou *front-end*, foi utilizada a linguagem de programação ***JavaScript*** em conjunto com a linguagem ***TypeScript*** e com o auxílio da *framework* ***VueJS***. Foi também utilizado ***Tailwind CSS***, uma ferramenta que facilita e agiliza o processo de escrita de CSS, tornando o desenvolvimento mais eficiente.

4.4.2 *Back-End*

Na camada de *back-end* foi utilizada a linguagem de programação ***Java*** e a *framework* ***Spring*** que facilita o processo de criação de APIs e consequente atendimento de pedidos *REST*. Para a comunicação com o servidor da base dados, foi necessário o uso de uma biblioteca adicional ***Java Connector***.

4.4.3 Base de Dados

Na camada de base de dados foi utilizada a linguagem de programação ***SQL***, sendo esta uma linguagem bastante popular e familiar entre os membros da equipa. Para o funcionamento deste nível, foi utilizado o ***MySQL Server*** para armazenar e gerir a base de dados do sistema. Para além disto, foi utilizado o sistema de gestão de bases de dados ***MySQL Workbench***, para facilitar o processo de análise e construção do servidor da base de dados.

5. Building Block View

Aqui serão organizados e demonstrados os elementos do sistema e suas implementações em diferentes componentes tendo em conta as suas funcionalidades e interações dentro do sistema. Serão também apresentadas, através de um diagrama, as diferentes classes dos vários componentes, tal como a análise das mesmas.

5.1 Componentes do sistema

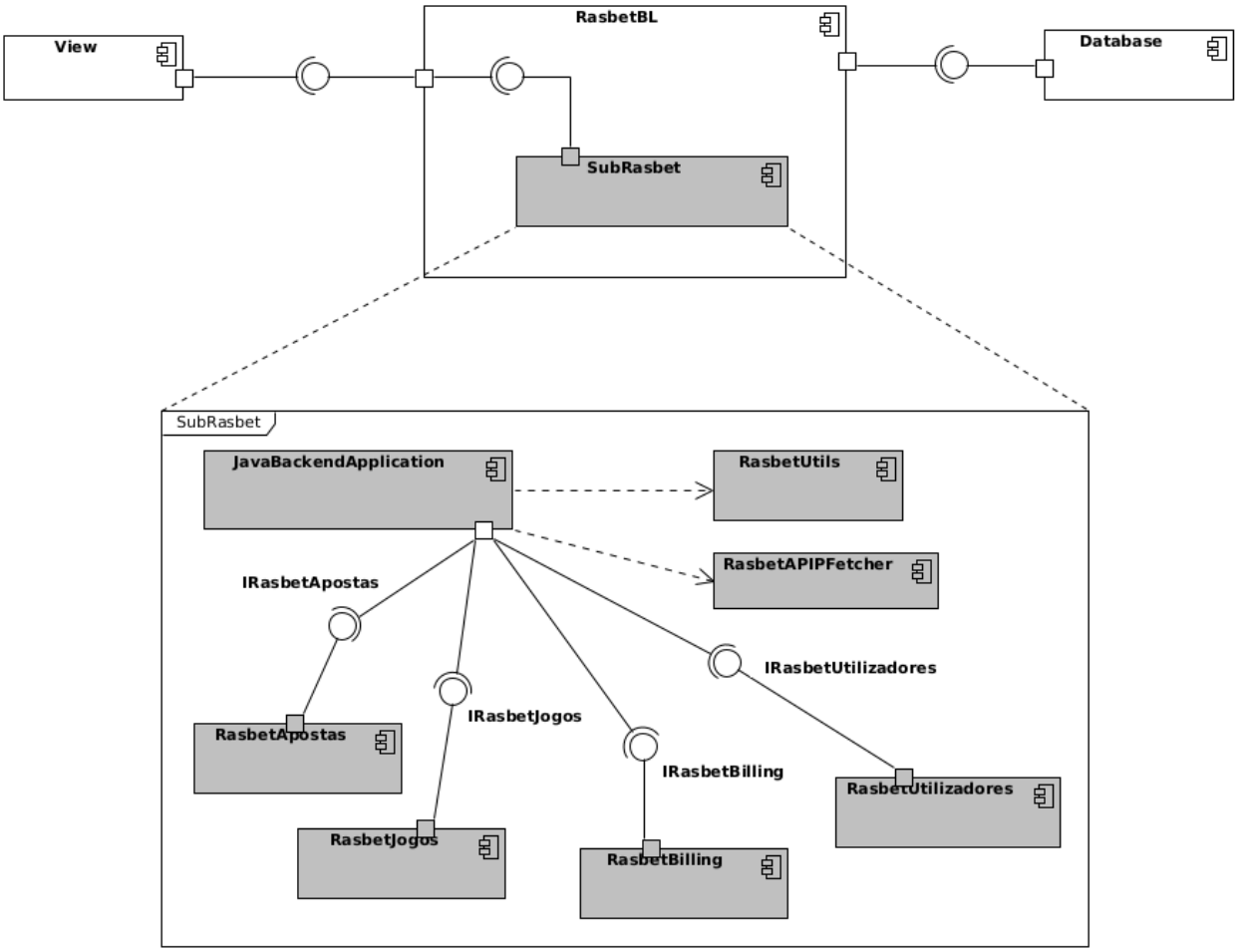


Figura 5.1: Diagrama de componentes.

5.2 *View*

Como referido anteriormente, a componente de *View*, ou *front-end* do sistema, é responsável pela apresentação da interface gráfica do sistema. A sua estrutura pode ser descrita como vários módulos interligados entre si que, na sua execução, resultam numa interface prática e completa. Assim, passamos a apresentar um diagrama de classes para melhor compreender a hierarquia e estrutura destes mesmos componentes.

5.2.1 Diagrama de Classes

Este diagrama apresenta os ficheiros principais utilizados na construção do *front-end*. O ficheiro ***main.ts*** é o ficheiro principal que trata de construir a aplicação, passando-lhe alguns argumentos, como o ficheiro de inicialização (***App.vue***), o ficheiro de *routing* das páginas (***router.ts***) e, por fim, a biblioteca de *stores* para *Vue* ***Pinia***.

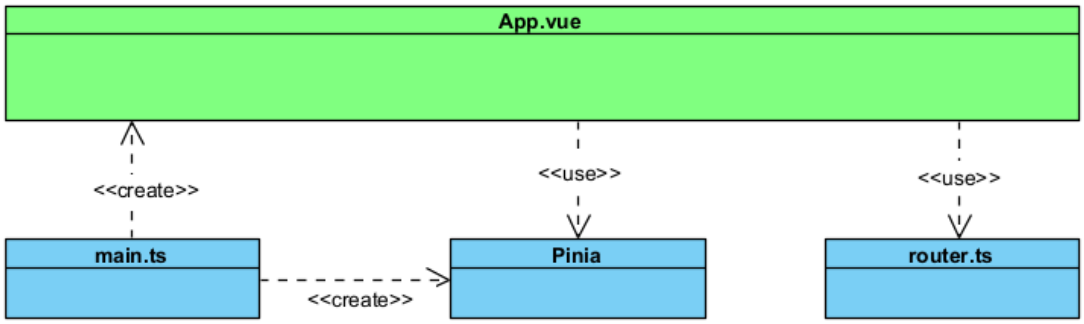
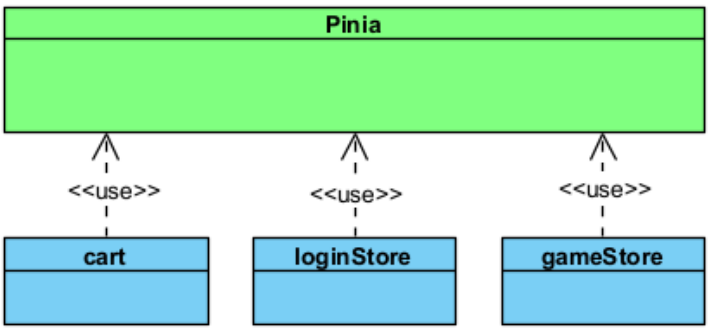


Figura 5.2: Diagrama de classes *View*.

5.2.2 *Pinia*

Pinia é uma biblioteca de *stores* para a *framework Vue*, que permite a partilha de estado entre componentes e páginas. A sua utilização é deveras útil e importante, por exemplo, para o boletim de apostas, onde muito conteúdo tem de ser partilhado por vários ficheiros. Ao estar tudo centralizado num só ficheiro/*store*, a troca e o acesso de informação entre componentes é muito mais eficiente. Assim, apresentamos um diagrama pormenorizado sobre os ficheiros que utilizam esta biblioteca:



cart.ts

Este ficheiro é responsável pela gestão do boletim de apostas dos apostadores. Tem métodos para adicionar novos resultados ao boletim, verificação caso se trate de uma aposta simples, entre outros. Para além destes, possui um método de envio da aposta para o *back-end*, onde realiza o *fetch* do pedido *POST* e espera a resposta.

loginStore.ts

Este ficheiro é responsável pela gestão da autenticação dos utilizadores do sistema. Este trata do processo de *log in*, *log out*, edição de perfil, registo, entre outros. Guarda em estado se o utilizador está com sessão iniciada, e, se sim, os seus dados pessoais para acesso mais rápido.

gameStore.ts

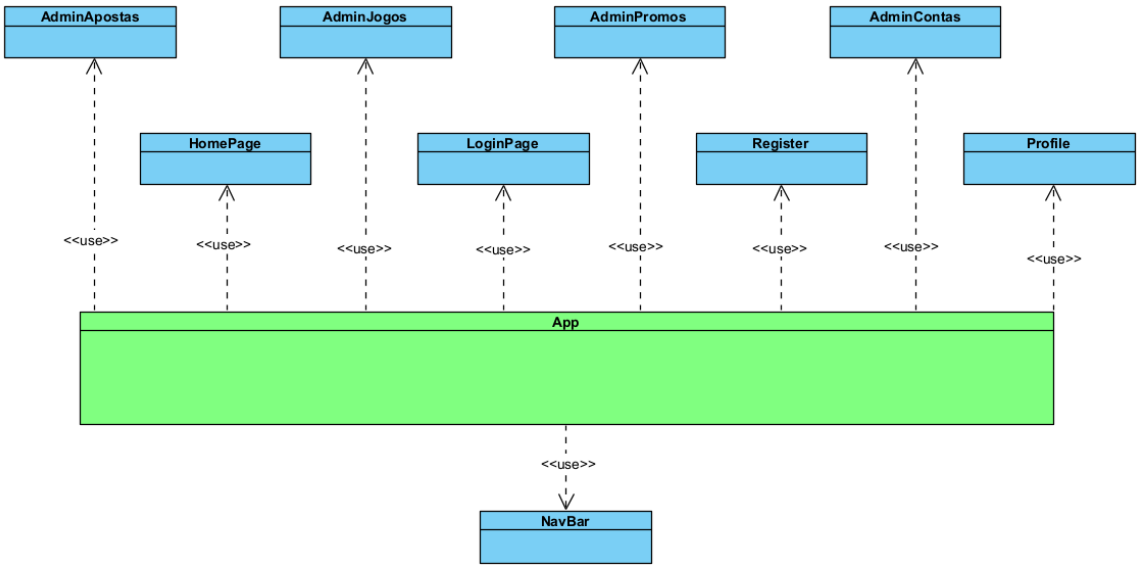
Este ficheiro trata da gestão dos jogos, modalidades e competições do sistema, assim como a filtragem dos mesmos quando o utilizador assim o requisita.

5.2.3 *Router*

O ficheiro ***router.ts*** é responsável pela definição das várias rotas possíveis na aplicação, isto é, dos vários endereços possíveis a serem acedidos. Para cada endereço é definido um ficheiro que contém o conteúdo da página, para assim ser exibido quando este é acedido. Para além disto, também é possível a definição de condições para o acesso às diversas páginas, como, por exemplo, no caso da página de *log in*, esta só pode ser acedida caso o utilizador não esteja autenticado.

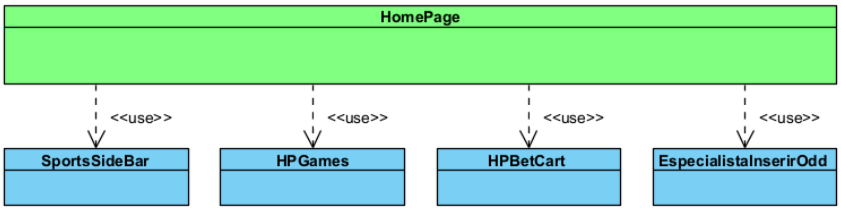
5.2.4 *App*

Este ficheiro é responsável pela seleção e organização das várias páginas a serem apresentadas ao utilizador, recorrendo ao ficheiro de *routing*. Este também contém uma componente fixa, a barra de navegação, responsável pela apresentação de atalhos para outras páginas dependendo do nível de autorização do utilizador.



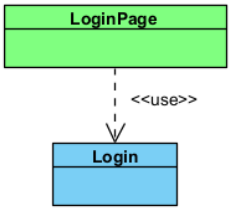
NOTA: Os ficheiros apresentados nesta e nas secções seguintes possuem todos a extensão '*.vue*', tendo sido omitido por questões de simplicidade.

5.2.5 *HomePage*



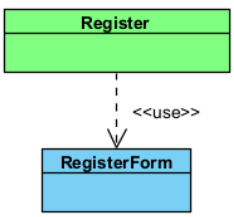
Este ficheiro organiza toda a informação apresentada na página principal da aplicação, variando esta conforme o tipo de encargo do utilizador. No caso do Administrador, é apenas apresentada uma lista de estatísticas em tempo real de algumas informações do sistema. No caso do Especialista e do Apostador, são exibidas listagens das várias modalidades, competições e jogos presentes no sistema (*SportsSideBar e HPGames* respetivamente), com a variação de que no caso do Especialista, os jogos apresentados são apenas aqueles aos quais ele está autorizado a inserir *odds*. Para além disto, no Especialista é ainda apresentado um componente de edição de *odds* (*EspecialistaInserirOdd*) e no Apostador o boletim de apostas (*HPBetCart*).

5.2.6 *LoginPage*



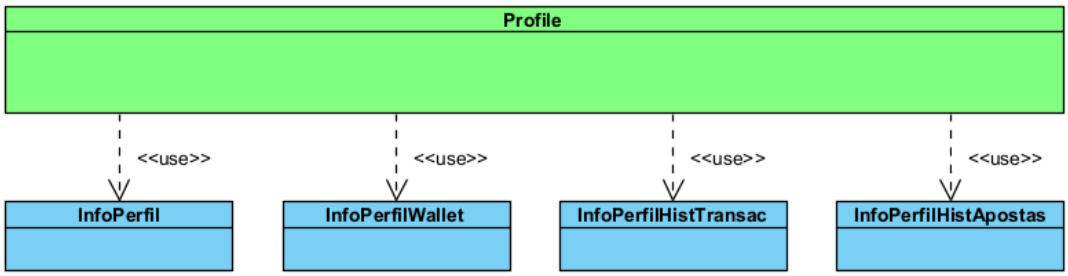
Este ficheiro é responsável pela autenticação dos utilizadores do sistema, quer sejam apostadores, especialistas ou administradores. Este coleta os dados de *log in* através de um formulário do ficheiro *Login*, envia o pedido de autenticação para a *Business Logic* e espera uma resposta de validação. Em caso de sucesso, o utilizador é novamente redirecionado para para a *home page*. Em caso de erro, é apresentada uma mensagem, ficando o pedido de início de sessão sem efeito.

5.2.7 *Register*



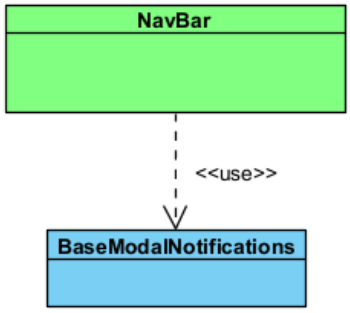
Aqui é apresentado todo o formulário de registo de novos utilizadores do sistema (*RegisterForm*). Este pede todos os dados necessários, fazendo uma pequena validação de tipo de *input* utilizando funcionalidades da linguagem HTML e expressões regulares. Em seguida, envia um pedido de registo à *Business Logic*, esperando por uma resposta. Em caso de sucesso, o utilizador é redirecionado para a *home page*, estando já com sessão iniciada.

5.2.8 *Profile*



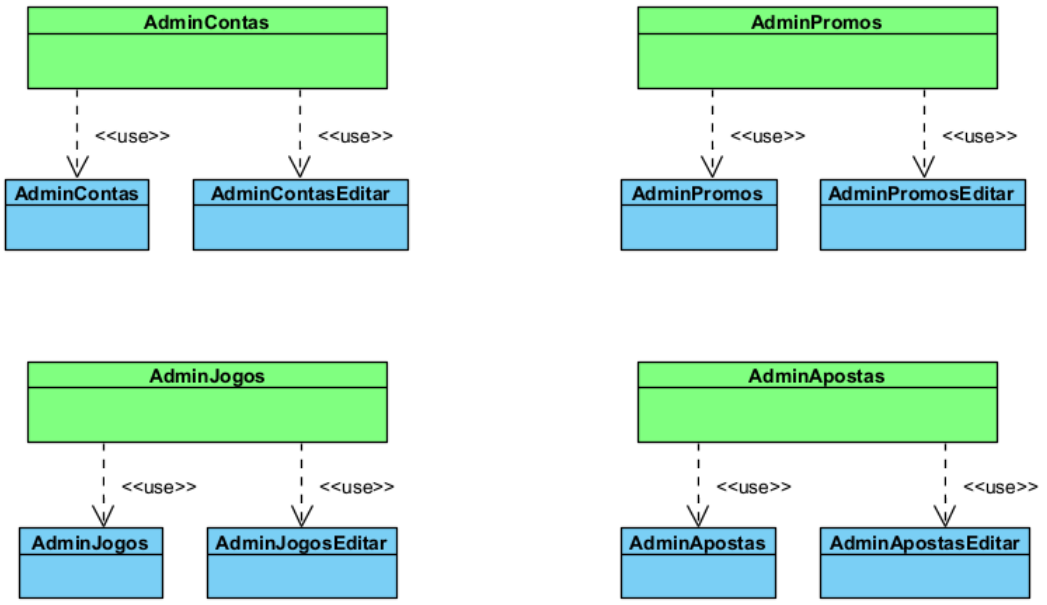
Este ficheiro é responsável pela apresentação da página de área pessoal dos apostadores, contendo os dados pessoais dos mesmos (*InfoPerfil*), o valor da carteira no momento (*InfoPerfilWallet*) e o histórico das apostas e das transações (*InfoPerfilHistApostas* e *InfoPerfilHistTransac* respetivamente). Esta página não pode ser acedida sem sessão iniciada, nem por especialistas ou administradores, uma vez que estes não possuem uma página de área pessoal. Para além disto, é nesta página que o apostador pode editar alguns dados pessoais (nome, e-mail, palavra-passe, entre outros), bem como depositar e levantar dinheiro na sua conta.

5.2.9 *NavBar*



Este ficheiro é responsável pela gestão da barra de navegação da aplicação. Esta apresenta alguns atalhos para certas páginas dependendo se o utilizador tem sessão iniciada ou se se trata de um utilizador com permissões especiais, isto é, especialistas e administradores.
Para todos os casos, a *navbar* apresenta um atalho para a *home page* (clicando no ícone da aplicação), um botão que redireciona para a página de *log in* caso o utilizador não esteja autenticado e um botão de *log out* caso o utilizador esteja com sessão iniciada, que, depois do processo de desautenticação, redireciona-o para a *home page*. Relativamente aos **apostadores com sessão iniciada**, a *navbar* apresenta um botão de 'Área Pessoal' que redireciona para a página de perfil e um ícone para aceder às suas notificações (*BaseModalNotifications*). Para os **administradores**, são apresentados quatro botões que redirecionam o mesmo para as quatro páginas referentes ao mesmo: *AdminApostas*, *AdminJogos*, *AdminPromos* e *AdminContas*.

5.2.10 *Admin*



Os quatro ficheiros com o prefixo *Admin* referem-se às páginas apresentadas apenas a administradores do sistema. Estas contêm informação sobre todas as apostas (*AdminApostas*), jogos (*AdminJogos*), promoções (*AdminPromos*) e, por fim, utilizadores (*AdminContas*) do sistema. Estas quatro páginas são muito semelhantes entre si, tendo uma parte de apresentação (AdminX) e uma parte de gestão (AdminXEditar), ou seja, por exemplo, na página das promoções, existe uma componente de apresentação das várias promoções presentes no sistema e, ao clicar em qualquer uma, esta é colocada na componente de gestão (*AdminPromosEditar*), podendo o administrador editar ou eliminar a mesma. Na componente de gestão, existe, também, um botão para criar novos elementos.

5.3 *Business Logic*

5.3.1 Diagrama de Classes

A *business logic* é representada pelas classes presentes no seguinte diagrama e os respetivos métodos. O sistema opera sobre uma classe principal que irá comunicar com a **View** e gerir o acesso à base de dados. Neste nível são obtidos os pedidos vindos do *front-end* e interpretados para serem posteriormente realizadas as chamadas à Base de Dados. Realizado este procedimento é então obtida uma resposta proveniente da conexão com a Base de Dados que será transmitida novamente para o *front-end*. Também é aqui que é realizado o acesso à API externa, e carregado toda a informação, devidamente tratada, na Base de Dados.

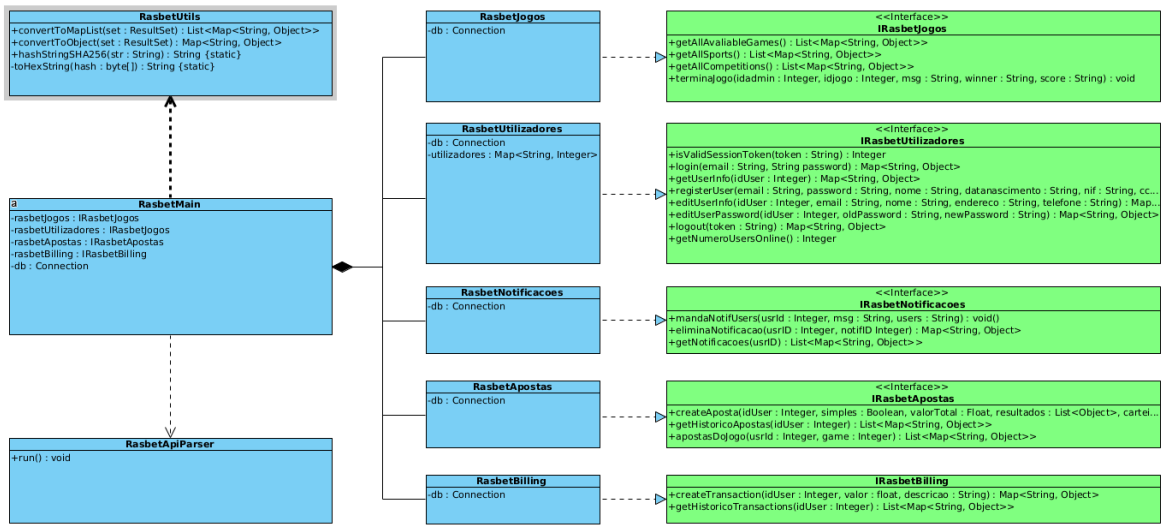


Figura 5.3: Diagrama de classes *Business Logic*.

5.3.2 Interfaces

IRasbetjogos

Esta interface contém os métodos que tratam de todos os pedidos relativos a jogos. Os métodos **getAllGames, getAllSports** e **getAllCompetitions** chamam as *procedures* que irão retornar uma lista com todos os jogos, desportos e competições, respetivamente. Esta interface também trata de transmitir a informação fornecida pelo administrador, para terminar um jogo (**terminaJogo**), para a Base de Dados, onde serão atualizados todos os valores correspondentes à finalização de um jogo.

IRasbetUtilizadores

Esta é a interface que possui os métodos relevantes para a transmissão dos dados dos utilizadores, para a Base de Dados, nomeadamente verificação de *tokens* e autenticação de contas (**isValidSessionToken**, **login**, **registerUser**, **logout**) e edição de dados dos utilizadores (**editUserPassword**, **editUserInfo**). Encontram-se também métodos usados para controlo sobre uma *Map* cujas chaves são referentes aos *tokens* dos utilizadores, que se encontram *online*, passados pelo *front-end*, e o valor do seu *id* na base de dados. Este *Map* encontra-se na classe *RasbetUtilizadores*.

IRasbetNotificacoes

Aqui encontram-se todos os métodos associados a operações sobre notificações, tal como, **mandaNotifUsers** que se certifica de chamar a *procedure* encarregue de atualizar as tabelas das notificações com base na lista de utilizadores a serem notificados. Este método só é usado por administradores.
Os métodos *eliminaNotificacao* e *getNotificacoes* são usados para eliminar e obter as notificações, respetivamente, associadas a um utilizador.

IRasbetApostas

Nesta interface encontram-se os métodos associados à realização de apostas por parte de um apostador registado (**createAposta**). Também possui o **getHistoricoApostas**, que garante a obtenção de todas as apostas realizadas pelo apostador que fez o pedido, e o método **apostasDoJogo** que obtém todas as apostas de um determinado jogo, sendo este só é chamado por administradores.

IRasbetBilling

Aqui permanecem todos os métodos usados para a realização de transações (**createTransaction**) e adquirição das informações das transações realizadas com base no utilizador fornecido.

5.3.3 Classes

RasbetMain

Como já foi referido, o sistema possui uma classe principal que contém os objetos relativos às interfaces descritas anteriormente como variáveis de instância. O contacto com a *View* é realizado a este nível, ou seja, é esta classe que trata de gerir os pedidos do *front-end* e receber os dados necessários à execução do pedido, assim como fornecer de volta uma resposta.

RasbetUtils

O *RasbetMain* recorre ao *RasbetUtils* para traduzir a informação proveniente da base de dados e do *front-end*, assim como auxiliar a encriptação de password através de *Hashing*.

RasbetApiParser

Esta classe foi criada à parte da *RasbetUtils* de forma a encapsular a sua funcionalidade e acesso, facilitando alguma alteração ou adição que seja realizada no futuro. O objetivo desta é realizar o *parse* da informação proveniente da API externa e é usada pela *RasbetMain*, onde é executada numa *Thread* de forma a evitar o bloqueio do Servidor e, após obter resposta da API com os dados fornecidos, faz a sua conversão e insere-os na Base de Dados do sistema.

5.4 Base de Dados

A Base de Dados do sistema **Rasbet** irá conter toda a informação persistente necessária para o funcionamento da aplicação. Esta será acedida e modificada exclusivamente pela camada de *Bussiness Logic* que irá chamar as *procedures* correspondentes, já inseridas na Base de Dados. No entanto, durante operações mais complexas, como o processar das transação de dinheiro, o controlo de sucesso e a garantia da atomicidade desta é responsabilidade da Base de Dados, através do comando *TRANSACTION* presente na linguagem SQL.

5.4.1 Modelo Lógico

Apresentamos de seguida o modelo lógico final da Base de Dados implementada, esta foi dividida em secções para uma melhor leitura e compreensão das várias tabelas presentes.

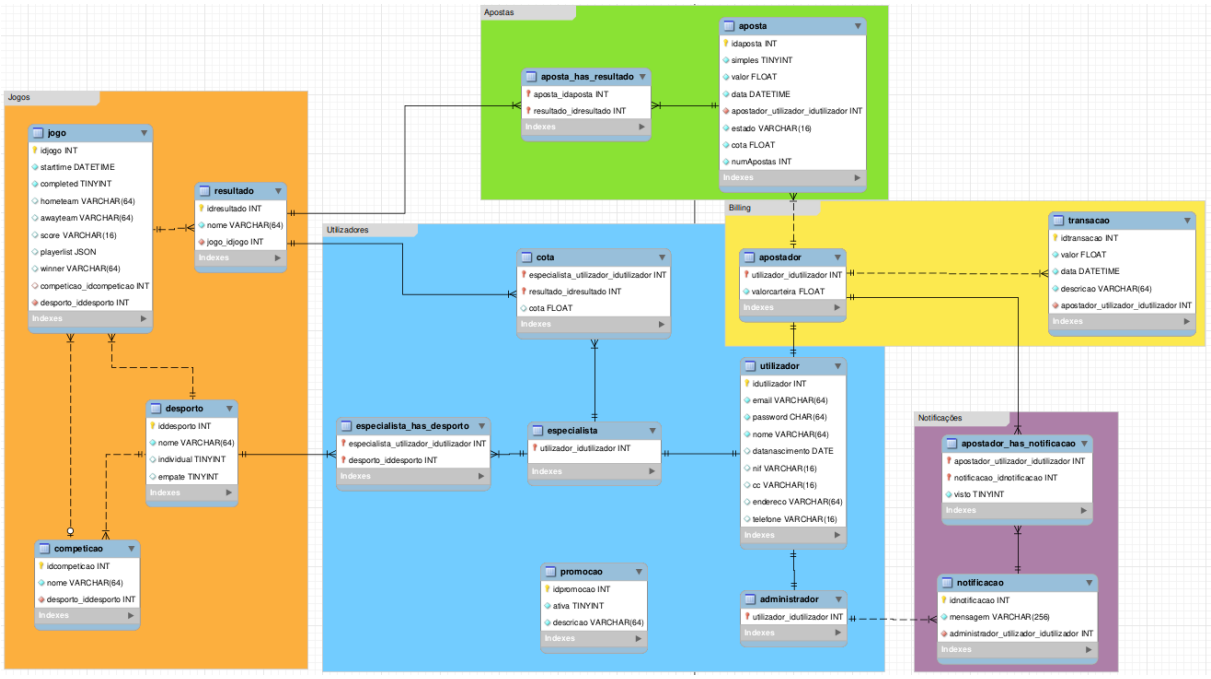


Figura 5.4: Modelo Lógico.

5.4.2 Utilizadores

Esta secção contém toda a informação acerca dos utilizadores do sistema, assim como as características que os definem. Tanto como os Apostadores, Especialistas e Administradores partilham a mesma tabela *utilizador*, visto que esta contém os atributos comuns entre todos como email e password.
De seguida os utilizadores são tipificados entre as três categorias de utilizadores presentes no sistema, diferenciando assim os seus papéis. Nos casos simples como o do Apostador, é apenas necessário incluir o valor atual da sua carteira. Já no Especialista, necessita de informação adicional sobre os desportos nos quais está qualificado a dar cotas, assim como as cotas que já deu aos vários jogos/resultados no sistema.

5.4.3 Jogos

A tabela *jogo* contém todos os jogos passados e futuros do sistema RasBet, estes são obrigatoriamente categorizados por desporto e, opcionalmente, por competição. Os jogos também possuem uma lista de resultados que poderá ter qualquer dimensão.

5.4.4 Apostas

As Apostas são caracterizadas pelo seus vários atributos como o tipo de aposta, a sua cota e o seu valor, assim como uma lista de resultados que essa aposta contém. No caso de Apostas de tipo simples, esta lista só deverá conter apenas um elemento, ao invés de Apostas múltiplas, que poderão ter N resultados, que serão controlados através do atributo *numApostas*.

5.4.5 Billing

A secção de *Billing* da Base de Dados é composta principalmente pela tabela *transacao*, esta contém as várias transações efetuadas pelos utilizadores, assim como os vários pagamentos e entrega de prémios geradas pelas Apostas. Para uma maior simplicidade, o valor das transações poderá ser negativo, representando assim um levantamento de dinheiro ou pagamento de uma aposta.

5.4.6 Notificações

As notificações são armazenadas entre duas tabelas, uma que representa a informação de uma notificação, e outra tabela que representa que Apostadores ainda não viram essa notificação. Quando vista, a linha correspondente deverá ser marcada como vista ou eliminada.

5.4.7 Procedures

A Base de Dados contém várias *Procedures*, sendo estas o método principal de acesso e modificação dos dados no sistema. No caso de operações com modificações múltiplas, onde será necessário garantir que estas ocorrem como um todo (atomicidade) é utilizado, como dito anteriormente, o comando *TRANSACTION*. Este comando garante que, caso algum erro ocorra entre uma operação mais complexa, nenhum dos comandos já executados terá efeito no estado final da base de dados.

6. Runtime View

Nesta secção apresentamos o comportamento da solução através de diagramas de sequência dos *use cases* mais importantes do sistema. Estes diagramas cobrem vários cenários apresentando as interações entre os diversos componentes do sistema, assim como situações de exceção ou erro e a forma como o sistema lida com estas.

6.1 Diagramas de Sequência

Estes diagramas de sequência contêm vários atores, sendo os principais os componentes do sistema referidos nas secções anteriores: *RasBetUI*, *JavaBackendApplication* e *RasbetDB*. Para além destas, são também incluídas classes utilizadas em passos intermédios como *IRasbetUtilizadores* e *RasbetUtils*. Relativamente aos *use cases* do sistema, irão ser apresentados os seguintes: Registo Apostador, *Log In*, Alterar Informações de Perfil, Consultar Jogos, Fazer Aposta, Criar Transação e, por fim, *Log Out*.

6.1.1 Registo Apostador

Este *use case* trata o registo de novos utilizadores no sistema, adicionando-os à base de dados do sistema. Os dados pessoais que o utilizador insere são verificados pelo RasBetUI em termos de tipo de dados e conteúdo através de expressões regulares, por isso o *back-end* apenas se tem de preocupar em adicionar o utilizador, caso este não exista já no sistema.

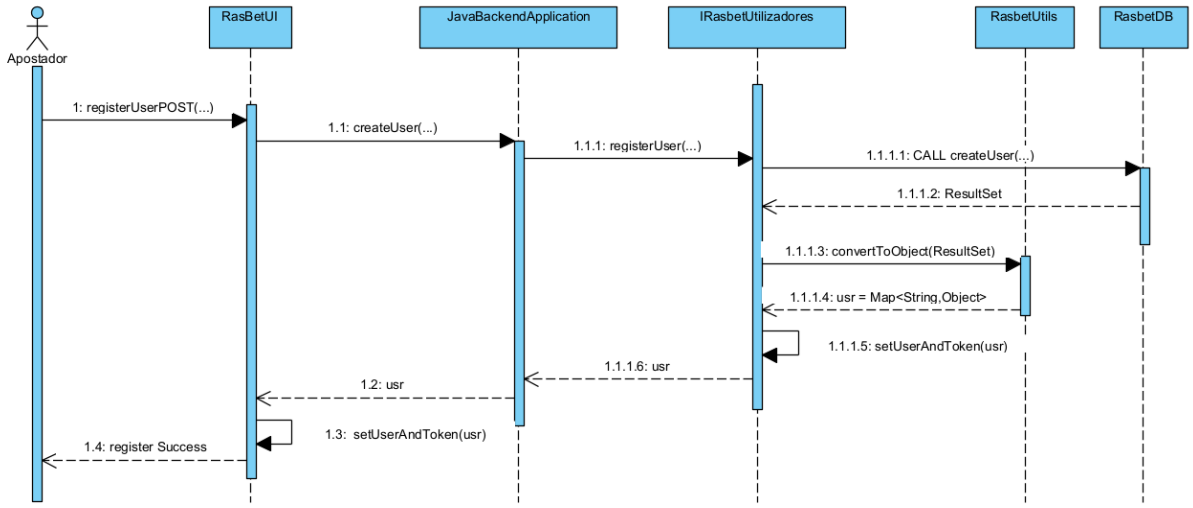


Figura 6.1: Diagrama de Sequência Registar Apostador

6.1.2 *Log In*

Este *use case* trata a autenticação do utilizador no sistema, quer seja apostador, especialista ou administrador. Existem dois tipos de *log in* no sistema, o *log in* por *email* ou por *token*. No caso do *email*, o *log in* é feito manualmente pelo utilizador através do formulário de início de sessão, onde preenche o mesmo com o seu *email* e palavra-chave. No caso do *token*, o pedido é feito automaticamente pelo componente do *RasBetUI* quando a página sofre um *refresh*, provocando uma verificação se o utilizador estava com sessão iniciada ou não, acedendo à informação do *token* guardado na *local storage* do *browser*.

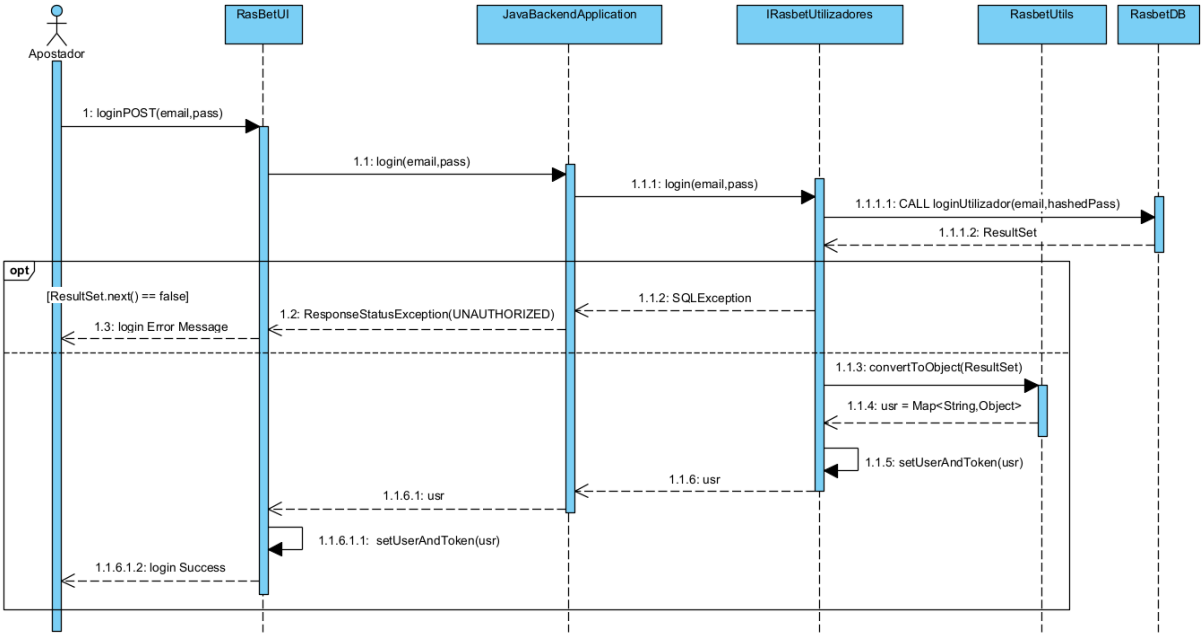


Figura 6.2: Diagrama de Sequência *Log In* Email

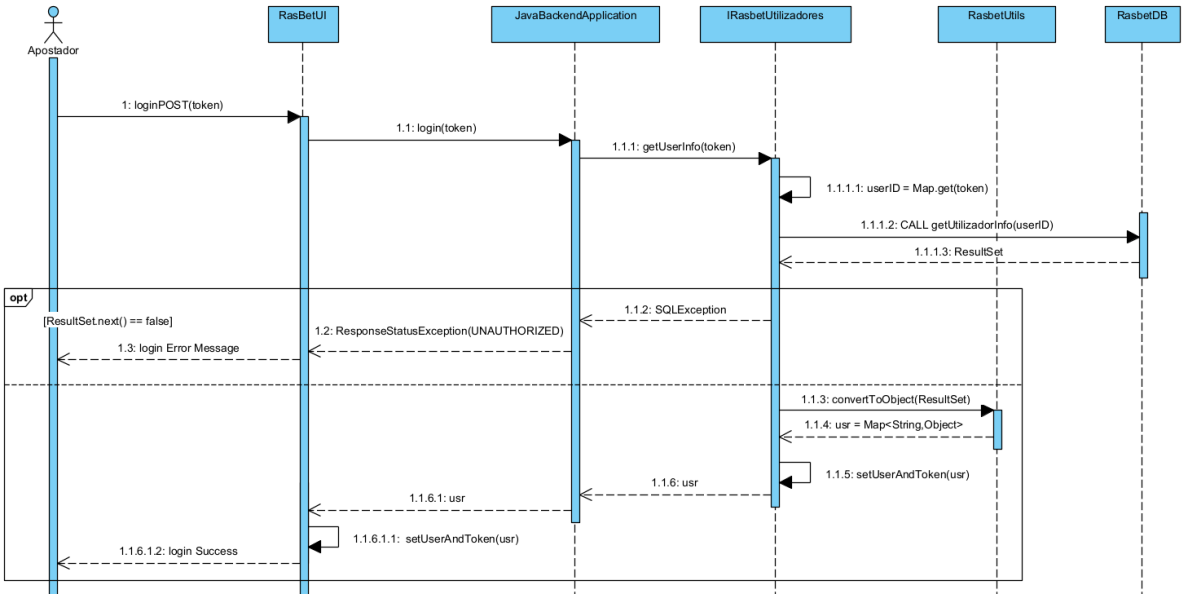


Figura 6.3: Diagrama de Sequência *Log In Token*

6.1.3 Alterar Informações de Perfil

O seguinte *use case* é relativo à modificação das informações pessoais de um apostador com registo. Este insere os novos dados que pretende atualizar no seu perfil e submete os mesmos, enviando toda a informação relativa ao apostador para o *back-end* e esperando pela resposta. Para aceder a esta página, o apostador necessita de estar autenticado.

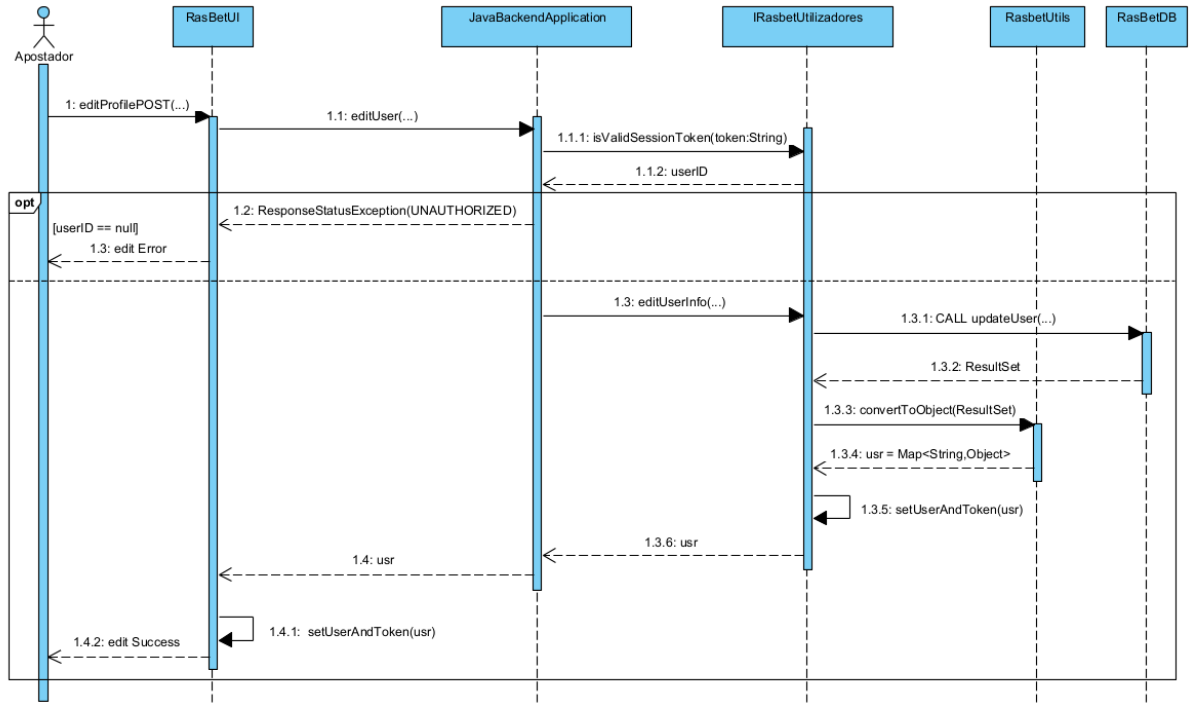


Figura 6.4: Diagrama de Sequência Alterar Informações Perfil

6.1.4 Consultar Jogos

Este *use case* consiste no pedido de todos os jogos presentes no sistema a partir o *RasBetUI*. Assim, sempre que qualquer utilizador atualiza o *browser* ou acede à página inicial da aplicação, é feito um pedido GET ao *back-end* para obter a lista com todos os jogos para, assim, apresentá-los na página. Essa informação é organizada ao nível do *RasbetDB* e enviada de volta para o *front-end* para ser exibido ao utilizador.

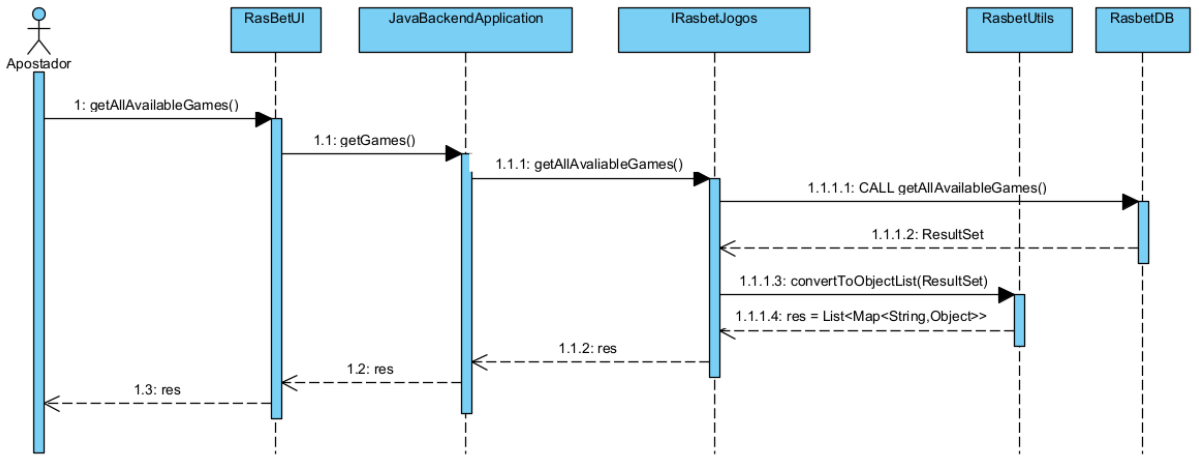


Figura 6.5: Diagrama de Sequência Consultar Jogos

6.1.5 Fazer aposta

Este *use case* consiste no envio de uma aposta realizada por um apostador com sessão iniciada, uma vez que não é possível fazer apostas sem estar autenticado na aplicação. Assim, o apostador seleciona os jogos e o resultado dos mesmos ao qual pretende apostar. Esta informação é organizada num boletim onde o apostador poderá adicionar o valor que pretende apostar tal como selecionar se é uma aposta simples ou múltipla. Se o apostador selecionar a carteira como método de pagamento, o *RasBetUI* verifica se existem fundos suficientes para fazer a aposta.

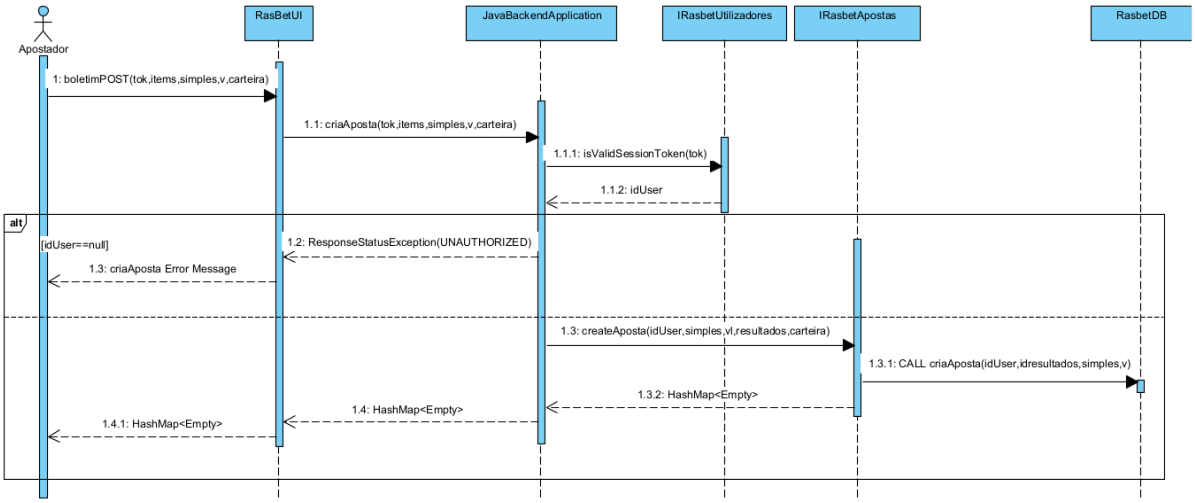


Figura 6.6: Diagrama de Sequência Fazer Aposta

6.1.6 Criar uma transação

Este *use case* ilustra o depósito ou levantamento de fundos na conta de um apostador, tendo, mais uma vez, este de estar autenticado no sistema.

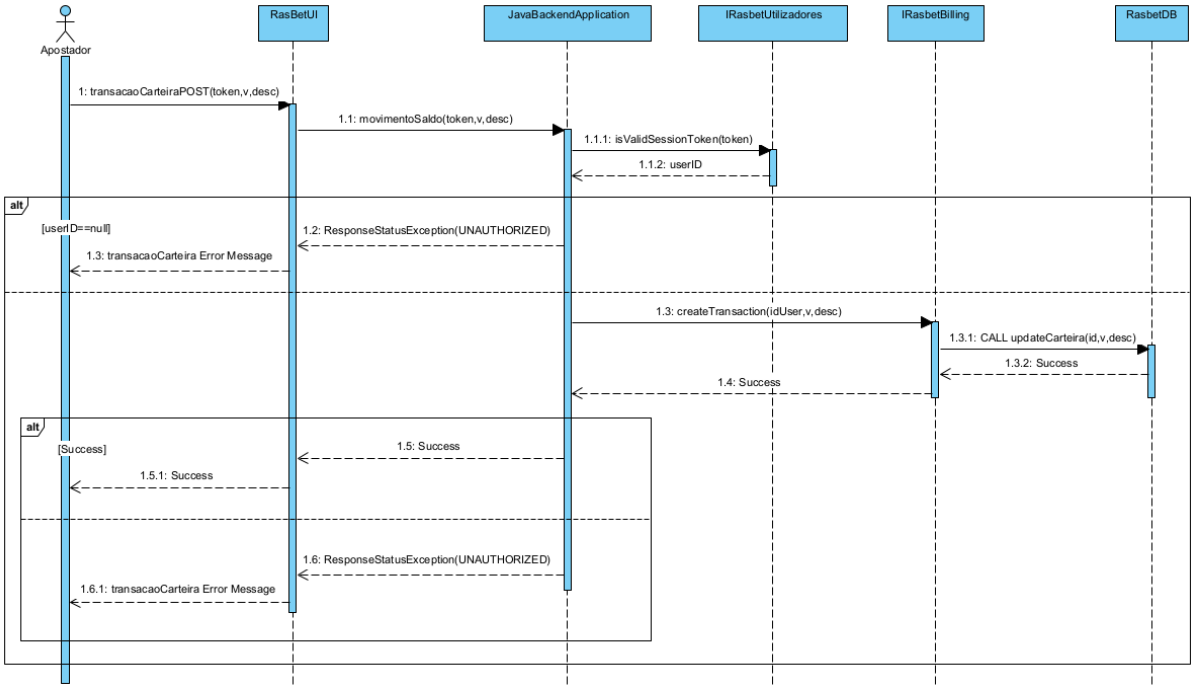


Figura 6.7: Diagrama de Sequência Criar Transação

O método utilizado, quer para depósitos, quer para levantamentos, é o mesmo visto que o sistema apenas soma o valor enviado ao valor presente na carteira do apostador, sendo este positivo no depósito e negativo no levantamento. O componente *RasBetUI*, no caso do levantamento, verifica se o valor a ser retirado não é maior do que o valor presente na carteira do apostador.

6.1.7 *Log Out*

Derivado do *use case Log In* resulta o *Log Out* e, por isso, é necessário que o utilizador esteja autenticado para conseguir executá-lo. Assim, quando o utilizador seleciona o *log out* no *front-end*, é enviada uma mensagem ao *back-end* contendo o *token* do utilizador, de modo a atualizar a lista de utilizadores *online* no sistema.

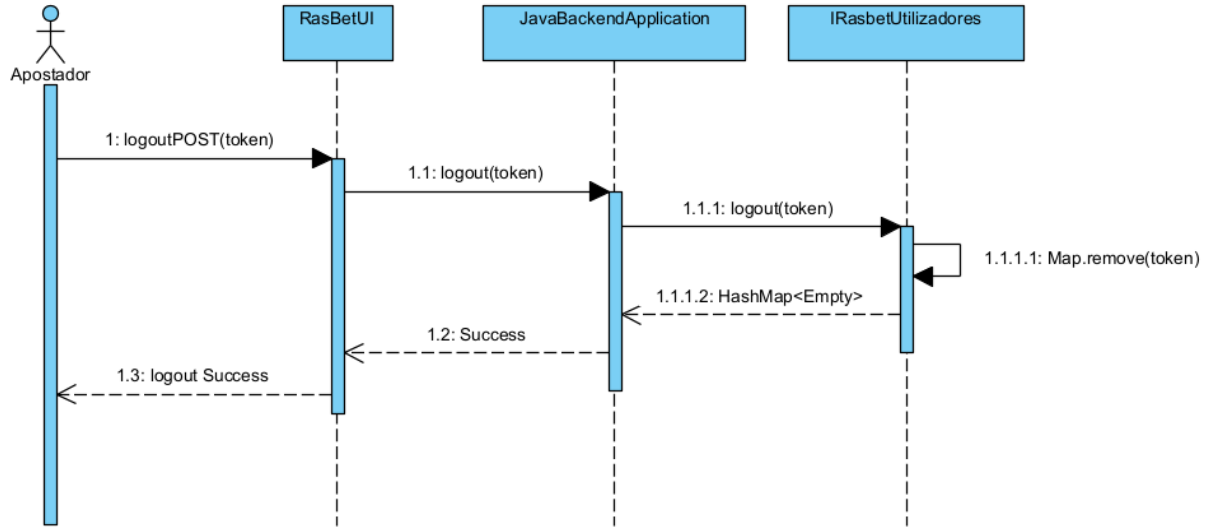


Figura 6.8: Diagrama de Sequência *Log Out*

7. Deployment View

Esta secção trata a infraestrutura técnica do sistema com ambientes, computadores, processadores e/ou topologias, representando os elementos da infraestrutura através de blocos.

7.1 Condições mínimas

Para o correto *deployment* do sistema, é necessário verificar várias condições mínimas tanto ao nível de *hardware* como de *software*. Assim, passamos a enumerar as diversas condições, notando que, sendo que havia limitações relativas à conexão estabelecida com a API, proveniente de instalações em serviços como *Google Cloud* devido ao uso obrigatório do VPN, optou-se por uma implementação numa máquina local:

- ***Laptop*** com 4Gb de RAM, 16Gb de armazenamento e com Ubunto LTS 20.04 como sistema operativo onde existem três componentes, onde dois são necessários para o funcionamento do *back-end* e outro para o *front-end*.
- Relativamente ao ***back-end***, é necessário o *MySql-Server 8.0.31*, necessário para o suporte e funcionamento da base de dados, para o *JavaBackend* o *Openjdk-17-jre-headless*, e, para a conexão com a API externa, é necessário instalar e configurar o programa "eduVPN" de forma a ligar-se à rede da 'Universidade do Minho'.
- Como requisitos do ***front-end*** encontram-se o *Nodejs 16* e *NPM 8.8.5*.

Assim, apresentamos o diagrama de *deployment* do sistema:

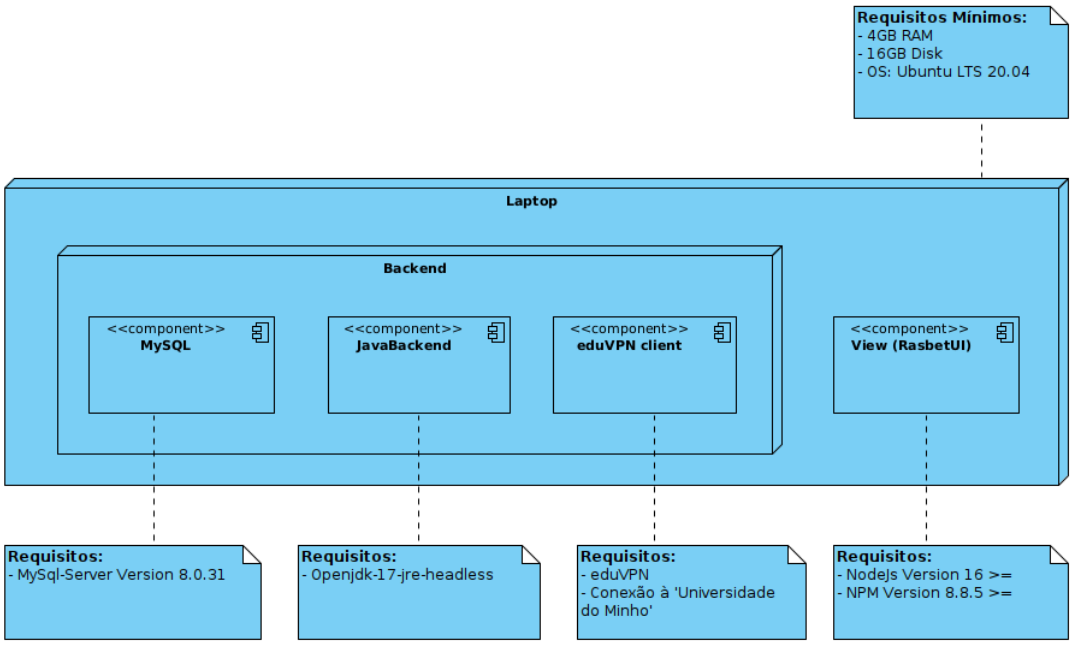


Figura 7.1: Diagrama de Instalação.