Universidade do Minho

UMinho

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

PL6 Grupo: 63 / Entrega 2

pg50633 Mariana Rocha Marques
pg50229 António Luís de Macedo Fernandes
pg50483 João Paulo Sousa Mendes
pg50518 José Diogo Martins Vieira
pg50499 João Silva Torres




Braga, 9 de agosto de 2023

Conteúdo

Lista de Figuras

1. Introdution and Goals

No âmbito da continuação do desenvolvimento do projeto de Requisitos e Arquiteturas de Software tínhamos como objetivo terminar toda a implementação do rasbet. Dito isto, foram tidos em conta todos os requisitos funcionais e não funcionais definidos na primeira fase, para que o *banckend* e o *frontend* da aplicação fossem bem desenvolvidos.

O principal fator que nos levou a escolher todos os requisitos foi a análise de casas de apostas online, bem como a experiência de alguns membros do grupo com o funcionamento das mesmas. Assim sendo, a escolha de todos os requisitos teve sempre como objetivo a simplicidade e objetividade da aplicação.

Numa fase incial preocupamo-nos com a implementação dos essenciais para o bom funcionamento do rasbet, tais como: *login*, *signin*, favoritos, calendrário de jogos, entre outros. Numa fase mais avançada decidimos introduzir requisitos que consideramos apelativos aos usuários, como por exemplo, 5 euros em freebets a cada utilizador no seu registo, quando ganha 5 apostas seguidas recebe 5% do valor apostado, *cashout* de 50% do valor apostado até a hora do evento começar, etc.

Quanto aos requisitos não funcionais, sobressaem-se 3 como sendo essenciais. Primeiro, o sistema só pode ser utilizado por maiores de 18 (idade legal para fazer apostas). Em segundo, a aplicação tem de ser fácil de utilizar para utilizadores não acostumados a sites de apostas desportivas. Por último, só é permitido aos especialistas alterarem informação relativa aos jogos e odds.

2. Constraints

Um dos requisitos principais para o desenvolvimento da aplicação foi a incorporação de uma API externa (API-rasbet) fornecida pelos docentes e obrigatória para a obtenção da informação. Para além disso, e já aboradado anteriormente definimos uma série de requisitos funcionais e não funcionais necessários e quase obrigatórios para o correto funcionamento da aplicação. Descrevendo os de maior prioridade de seguida:

- **Requisitos Funcionais**
  - O Apostador tem de se registar na aplicação para a poder utilizar;
  - O Utilizador valida os seus dados para submeter uma aposta;
  - O Especialista pode alterar e/ou adicionar eventos desportivos, bem como as odds associadas aos mesmos;
  - A aplicação deve ter um calendário dos jogos;
  - A aplicação deve permitir ao utilizador realizar diversas apostas em eventos desportivos;
  - A aplicação permite o registo de novos utilizadores;
- **Requisitos não Funcionais**
  - A aplicação permite o carregamento/levantamento de saldo da sua carteira através de 3 métodos de pagamento distintos;
  - O Sistema deverá ter alta disponibilidade;
  - O Sistema deverá ter rigor ao calcular os ganhos de apostas conforme as odds e uma rápida atualização das mesma;
  - É só permitido aos especialistas alterarem informação relativa aos jogos e odds;
  - O sistema só pode ser utilizado por maiores de 18 (idade legal para fazer apostas);

Apesar de termos definido mais requisitos, estes foram os que achamos que tinham maior relevância para o nosso sistemas e que foram a nossa prioridade ao implementar a aplicação. Para alem disto, também nos foi imposto uma restrição temporal de entrega até dia 5 de dezembro de 2022, o que nos fez ignorar alguns requisitos mais superficiais.

3. Context and Scope

Em termos de contexto e âmbito de sistema, como ator principal iremos ter o utilizador que tanto pode ser Apostador, Administrador e Especialista. Cada um destes irá ter funcionalidades diferentes mas todas elas irão, através da interface gráfica, comunicar com o *backend* do nosso sistema. Isto é, para cada pedido do utilizador realizado no *frontend* este irá responder ao mesmo no *backend*. Por sua vez, este realiza os pedidos necessários na Base de Dados a partir da informação da API fornecida pelos docentes. Estes dois últimos, funcionam como principais fontes de armazenamento e obtenção de informação, quer relativos aos jogos e suas odds (RASbet API), quer a dados pessoais do utilizador e seus histórico de apostas (BD).

4. Solution Strategy

Nesta secção iremos abordar quais as estratégias que foram implementadas para dar vida à nossa aplicação. Podemos afirmar que tal segue um padrão MVC, Model View Controller. Ao utilizar este modelo, podemos isolar o nosso código em camadas, nomeadamente, a da lógica de negócio com a interface gráfica da aplicação.

Inicialmente começamos por desenvolver o nosso model (back-end) em *Java*. É nesta camada que foi definida a lógica da aplicação e dividida em 3 subdivisões: lógica de negócios das apostas, utilizadores e jogos. Em cada uma delas definimos as principais entidades do nosso projeto (Jogo, Utilizador e Aposta) e os diferentes métodos necessários para realizar os diferentes requisitos anteriormente referidos. Como banco de dados utilizamos uma base de dados (MySQL) que irá armazenar as informações necessárias de cada utilizador assim como das suas apostas. Sempre que inserimos, atualizamos ou removemos uma determinada informação da nossa base de dados esta é feita através de queries que se encontram definidas na pasta relativa à BD.

De seguida, vamos abordar a camada View. Esta foi desenvolvida em React. React trata-se de uma biblioteca front-end em *JavaScript* em que o principal intuito é desenvolver uma interface em página web. Utilizando como referência as mock ups dadas, fomos dando vida a nossa aplicação web. Também através do react somos capazes de fazer pedidos ao nosso back-end e à API fornecida pelos docentes. Assim sendo, funciona assim como Controller da nossa aplicação, uma vez que faz a comunicação entre a nossa lógica de negócios com a interface gráfica da aplicação.

5. Building Block View

Para melhor visualização da modelação das classes do nosso projeto e os respetivos metodos das mesmas, segue-se em seguida o diagrama de classes da nossa solução:

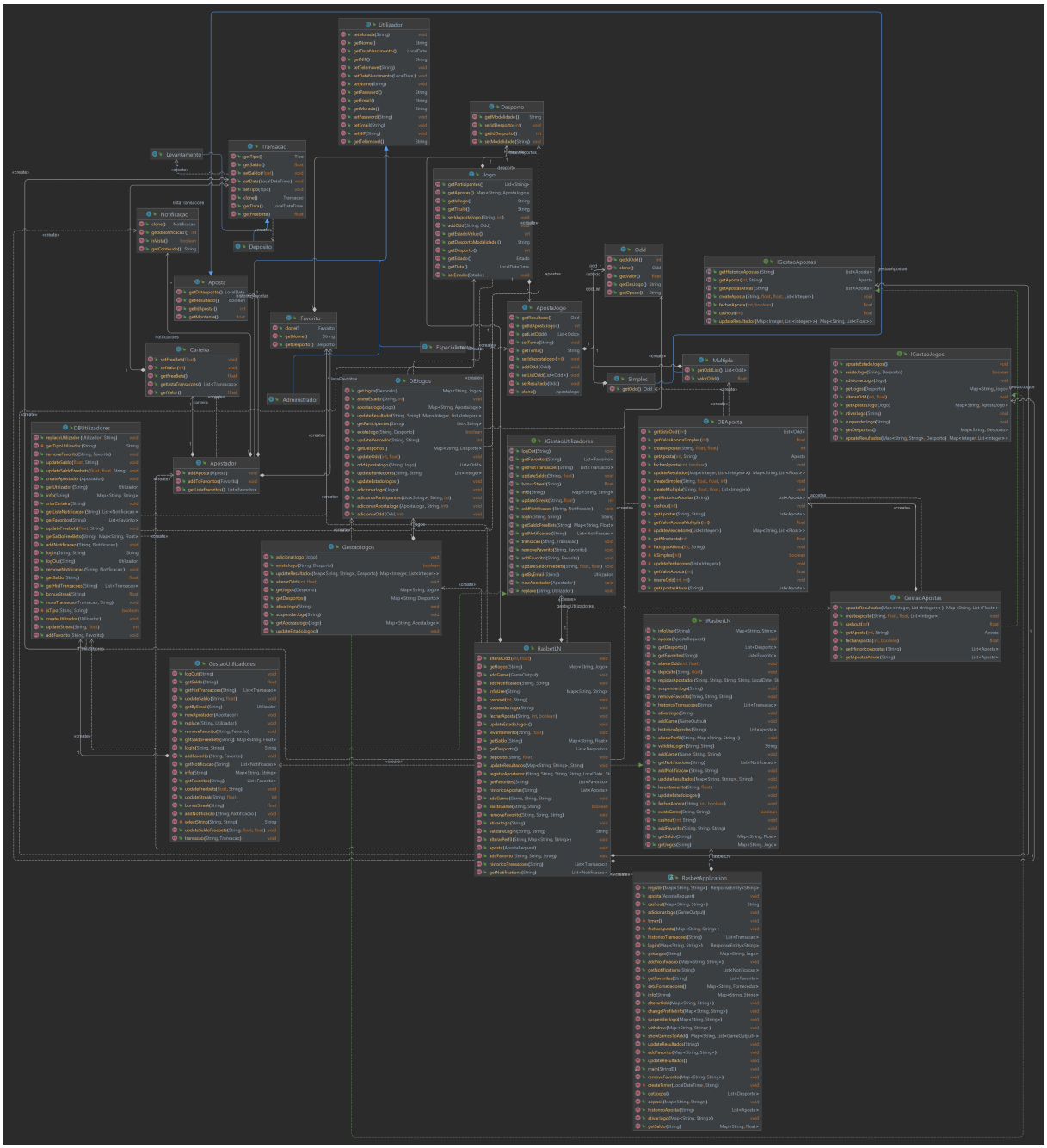

Figura 5.1: Diagrama de Classes

(A figura encontra-se em anexo para melhor visualização da mesma)
No seguinte diagrama de componentes conseguimos demonstrar os relacionamentos entre os diferentes componentes do nosso sistema e ver com mais claridade a distinção do nosso código em duas camadas, o frontend e o backend.

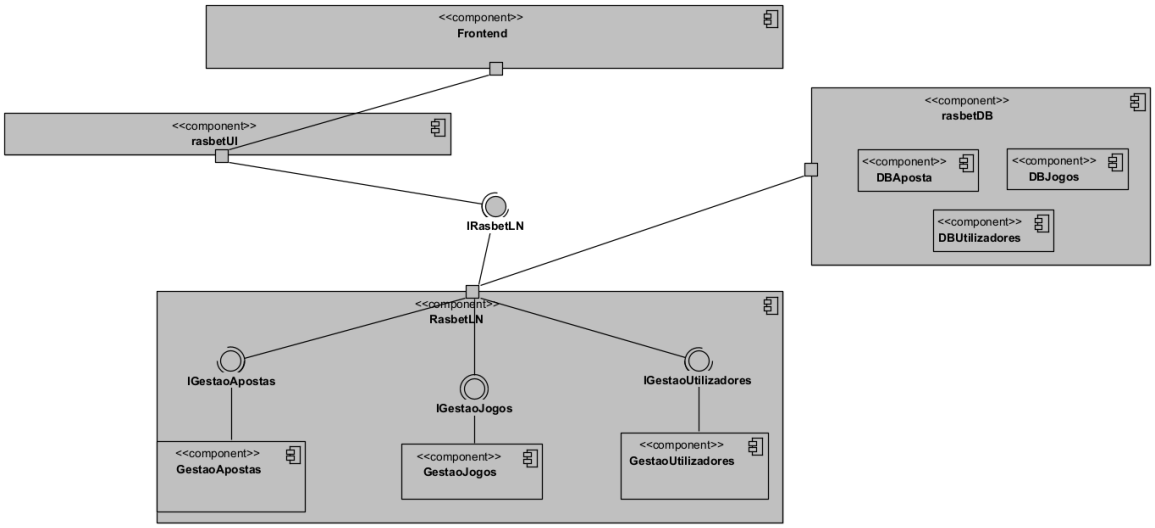

Figura 5.2: Diagrama de Componentes

6. Runtime View

De forma a compreender melhor como são realizadas algumas das funcionalidades do nosso sistema, fizemos uma série de diagramas de sequência para melhor ilustrar a troca de mensagens entre os vários intervenientes do nosso código conforme os diferentes pedidos. Uma grande parte dos diagramas já se encontram respresentados no documento de requisitos, iremos apenas ilustar aqui os seguintes diagramas de sequência relativos a use cases extras implementados pelo grupo. Nomeadamente:

6.1 Consultar Favorito

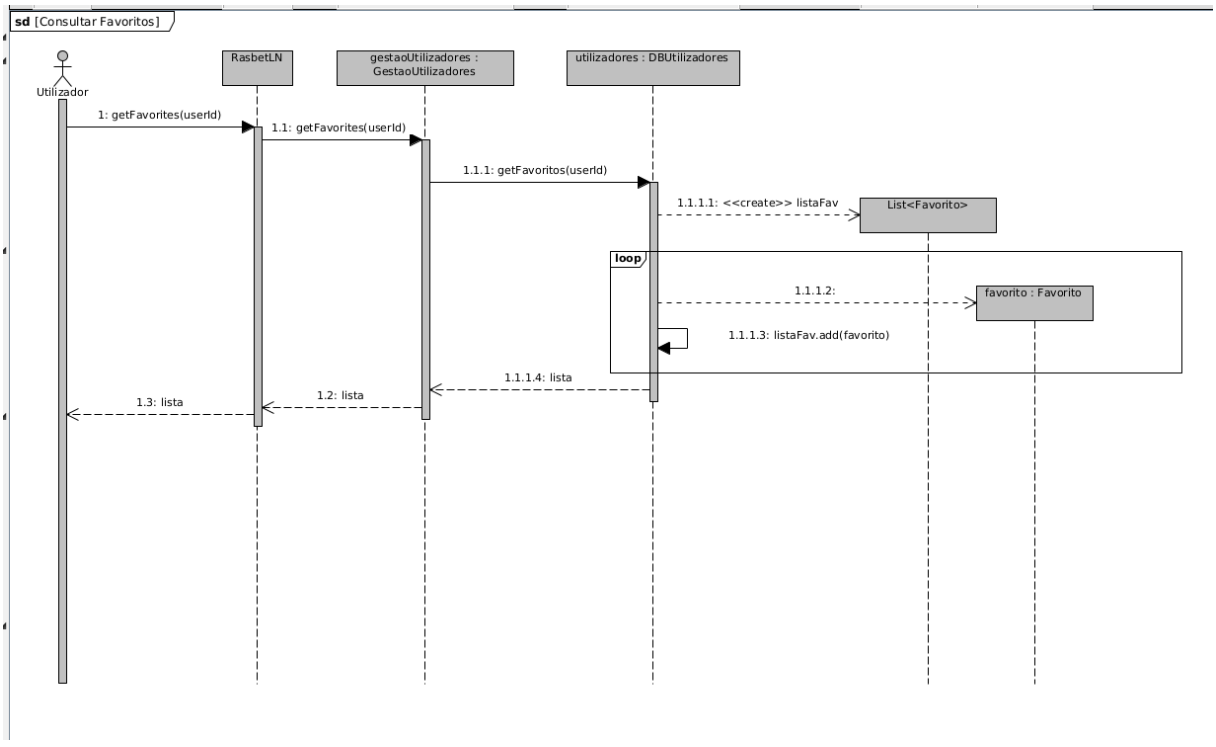

Figura 6.1: Diagrama de Sequência Consultar Favorito

6.2 Cash-out

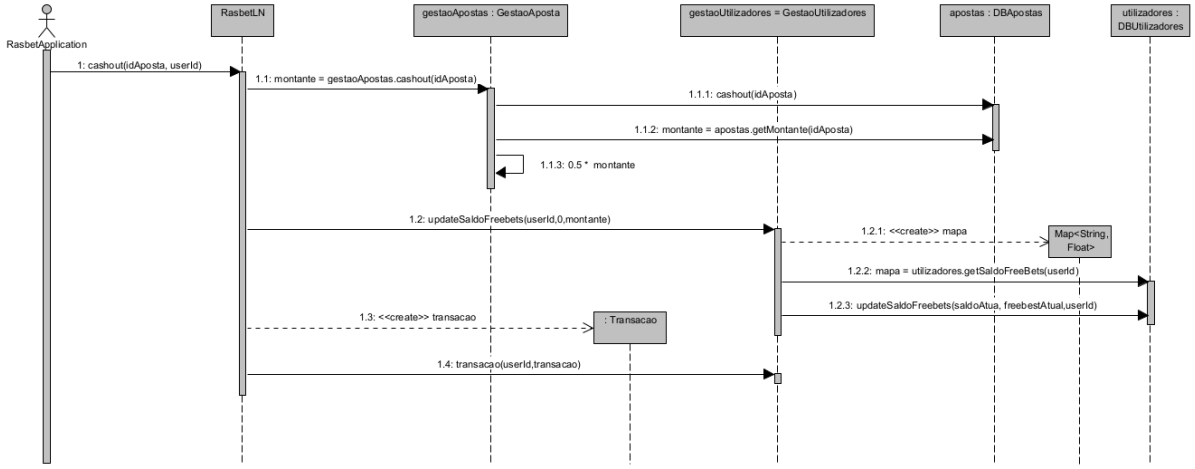

Figura 6.2: Diagrama de Sequência de Cashout

7. Deployment View

Através do seguinte diagrama de implementação conseguimos entender melhor a lógica do nosso website que após receber os diferentes pedidos do utilizador este irá comunicar com o nosso backend através de pedidos HTTP. Este por sua vez, responde aos pedidos comunicando com a API e/ou Base de Dados do sistema.

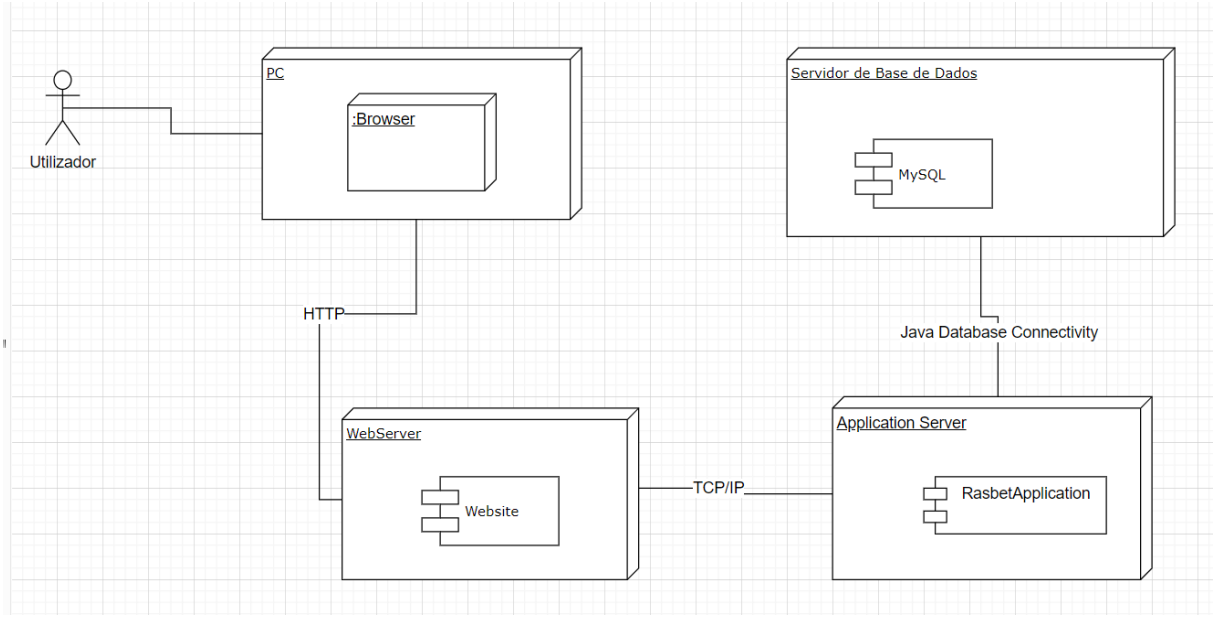

Figura 7.1: Deployment diagram