UMinho
Mestrado Engenharia Informática
Requisitos e Arquiteturas de Software

Grupo 2, PL7 / Entrega 3
Relatório da solução arquitetural

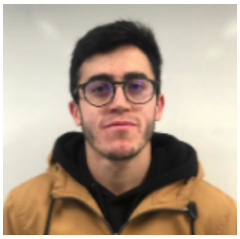
PG50222 André Manuel da Costa Ferreira,

PG50275 Bruno Ricardo Silva Campos,

PG50374 Francisco Alberto do Fundo Novo,

PG50384 Francisco Reis Izquierdo,

PG50779 Tiago Fernandes Ribeiro

17 de janeiro de 2023
2022/23

Conteúdo

1. Introdução e objetivos

A aplicação *RASBET* surge como um produto informático de suporte a apostas desportivas, no qual a mesma permite aos utilizadores realizarem apostas em diversos eventos desportivos. De entre as várias funcionalidades já implementadas, a *RASBET* suporta apostas desportivas no âmbito do futebol, no entanto existem diversos objetivos que se pretendem atingir futuramente com a implementação desta aplicação, tais como:

- O sistema deverá suportar vários tipos de desportos;
- O sistema deverá suportar vários utilizadores;
- O sistema deverá permitir realizar levantamentos bancários de forma segura;

1.1 Requisitos Principais

Requisito	Descrição
Registar	O sistema deve permitir o registo de um utilizador que ainda não se encontra no sistema.
Autenticar	O sistema deve permitir a autenticação de um utilizador que se encontra no sistema, sendo utilizados o email e a palavra-passe para este efeito.
Alterar informações do perfil	O sistema deve permitir a alteração dos parâmetros do perfil de um utilizador.
Consultar jogos	O sistema deve permitir ao utilizador selecionar um ou mais desportos e ver a lista de jogos desse(s) desporto(s).
Fazer aposta	O sistema deve permitir que os utilizadores realizem apostas.
Consultar histórico de apostas	O sistema deve permitir que os utilizadores tenham acesso ao seu histórico de apostas realizadas.
Depositar dinheiro	O sistema deve permitir que os utilizadores depositem dinheiro na sua carteira.
Levantar dinheiro	O sistema deve permitir que os utilizadores levantar dinheiro da sua carteira.
Inserir odd	O sistema deve permitir que os especialistas alterem a odd de jogos futuros.
Alterar estado da aposta	O sistema deve permitir que os administradores alterem o estado de uma aposta.

Tabela 1.1: Descrição dos requisitos principais

1.2 Objetivos de Qualidade

Além do supramencionado, convém mencionar os objetivos de qualidade mais importantes e que foram tidos em consideração no desenvolvimento da aplicação. Deste modo, todo o desenvolvimento desta fase, não só se focou nos requisitos acima descritos, como também nos seguintes objetivos de qualidade.

Qualidade	Âmbito
Performance	O sistema deve atender as funcionalidades requisitadas pelo cliente de forma eficiente, havendo um balanço dos recursos usados e disponíveis.
Segurança	O sistema deve estabelecer um ambiente fidedigno, por forma a assegurar as informações privilegiadas do cliente e do próprio sistema.
Manutenção e Suporte	O sistema deve estar apto à utilização por parte de diversos clientes que se inserem em condições distintas e capaz de ser melhorado/modificado.
Usabilidade	O sistema deve ser de fácil compreensão e rapidamente adaptável.
Aparência	O sistema deve ser fácil de navegar e deve captar a atenção do utilizador.

Tabela 1.2: Objetivos de qualidade a atingir

2. Restrições

No desenvolvimento da aplicação *RASBET*, existem diversas restrições que a equipa deve ter em conta aquando de tomar decisões tanto no âmbito da implementação como do design. Estas restrições, mostram ser de teor diversificado, no qual podemos listar as restrições mais fulcrais.

Restrição	**Descrição**
API externa *RASBET* API	A aplicação tem de fazer uso da *RASBET* API, que irá fornecer diversas informações sobre os jogos disponíveis, bem como os mesmos.
Tipo de aplicação	A aplicação tem de ser um web site.
Administradores e Especialistas	Os Administradores e Especialistas são previamente definidos não sendo possível registar novos Administradores e Especialistas.
Base de Dados	Haverá uma base de dados a guardar os dados relativos ao sistema.
Servidor	Haverá um servidor responsável pelo funcionamento da aplicação.
Internet	É necessário que o servidor tenha conectividade com a Internet, por forma a consultar e fazer uso da API externa *RASBET* API.
Linguagem da back-end	A linguagem da extensão da back-end desenvolvida previamente, deve ser mantida em Java.
Multi Cliente	O sistema deverá suportar o uso simultâneo por parte de diversos clientes.
Desportos	O sistema deverá suportar listar os jogos por desporto.
Prazos de entrega	4 de novembro: Requisitos mais implementação funcional. 5 de dezembro: Requisitos não funcionais da solução. 17 de janeiro: Alteração funcional e otimização do sistema.

Tabela 2.1: Restrições

3. Contexto e Alcance

Uma vez especificado os objetivos principais e de qualidade, bem como as restrições impostas no desenvolvimento da aplicação, convém mencionar o contexto e o alcance que é pretendido obter. Deste modo, teremos de abordar aspetos e especificá-los, nomeadamente os seus intervenientes (utilizadores) e os sistemas envolventes.

3.1 Perspetiva de Domínio

3.1.1 Utilizador

O utilizador é o interveniente responsável pela interação direta com a aplicação. Este interage com o pretexto de obter uma resposta da aplicação que satisfaça o seu pedido, no qual o mesmo pode ser prol de um dado contexto. Acima de tudo, um utilizador ao interagir com a aplicação, pretende conhecer ou aceder e usufruir do serviço disponível.

3.1.2 *RASBET*

Sistema principal que é responsável por receber e responder aos pedidos provenientes de um qualquer utilizador.

3.1.3 API

O sistema *RASBET* usufrui de informação disponibilizada pela API externa *RASBET API*, responsável por agregar informação relevante a apostas desportivas relativas a eventos de futebol.

3.2 Perspetiva Técnica

3.2.1 Utilizador

Podemos destacar 3 tipos de utilizador que interagem com o sistema:

- **Apostador**: O Apostador é o utilizador de maior prioridade e a quem pretendemos disponibilizar o serviço desenvolvido e todas as suas vertentes. Tem a capacidade de realizar apostas dos eventos desportivos do seu interesse, bem como aceder e alterar informações relativas ao seu perfil e seguir outros apostadores, entre outros.
- **Especialista**: O Especialista é o utilizador responsável por atualizar informações relativas aos jogos, nomeadamente as *odds* dos mesmos. É um interveniente intrínseco e necessário para o funcionamento da aplicação.
- **Administrador**: O Administrador é o utilizador responsável por abrir, fechar ou suspender as apostas, por gerir as notificações e criar promoções. É um interveniente intrínseco e necessário para o funcionamento da aplicação.

3.2.2 *RASBET*

A *RASBET* é o sistema principal que permite disponibilizar o serviço de apostas de eventos desportivos aos utilizadores, nomeadamente apostadores. Este sistema é o resultado da combinação e coletivo de outros sistemas que servem de suporte.

- ***Front-end***: Foi desenvolvido com vista a implementar o design da aplicação em formato *web*, bem como disponibilizar e puder atender os pedidos do utilizador.
- **Intermediário**: Sistema que permite criar a conexão e o envio e receção dos pedidos entre a *front-end* e a *back-end*.
- ***Back-end***: Foi desenvolvido com vista a responder aos pedidos provenientes da *front-end*, sendo responsável por atualizar e manter as informações do sistema atualizadas e coerentes, contactando com a API externa e a base de dados.
- **Base de dados**: Sistema que contacta com a *back-end*, sendo responsável por agregar toda a informação do sistema e atualizar a informação proveniente da *back-end*.

3.2.3 API

Sistemas externos responsáveis por disponibilizar informação útil à aplicação desenvolvida. Esta informação é relativa a eventos desportivos, sendo esta informação entregue através de um ficheiro *JSON*, sendo a própria API que atualiza as suas informações. É de salientar que o sistema *RASBET* apenas usufrui de uma API externa, a *RASBET API* que é responsável por disponibilizar informações relativas a jogos de futebol.

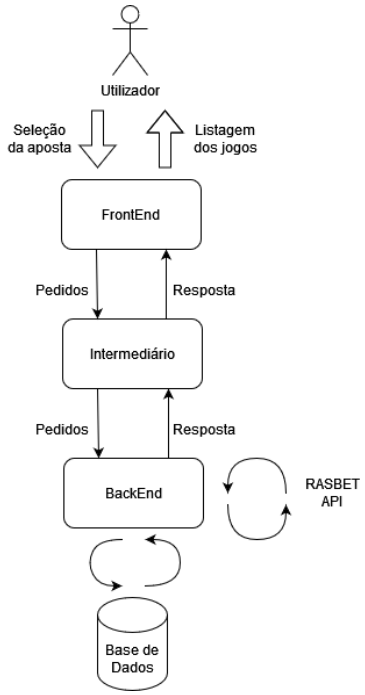


Figura 3.1: Diagrama da arquitetura da aplicação

4. Estratégia da solução

O projeto desenvolvido obrigou o grupo de trabalho a tomar decisões no que concerne aos vários tópicos abordados ao longo deste relatório. Neste âmbito, este capítulo relata todas as decisões e estratégias tomadas que definiram toda a estrutura da aplicação *RASBET*. Assim, toda a aplicação envolve um conjunto de padrões arquiteturais com vista a atingir o objetivo proposto, das quais as mesmas foram tidas em consideração na solução desenvolvida.

4.1 Padrões Arquiteturais

4.1.1 Model View Controler

Este padrão arquitetural foi abordado com o intuito de delinear e organizar todos os processos envolventes no sistema, uma vez que este padrão possui 3 componentes na qual cada uma tem uma definição específica. É de salientar que na solução desenvolvida não foi explicitamente definida cada componente, mas cada componente foi intrinsecamente abordada. Isto é, no que concerne à componente *View*, esta encontra-se definida pela *front-end*, que disponibiliza uma interface de utilizador que permite ao mesmo interagir com o sistema. No que concerne às componentes *Model* e *Controller* estas estão mutuamente definidas, uma vez que na *back-end* não há uma separação lógica entre o *Model* e *Controller*. Assim, estão definidas na *back-end* a componente *Model* que contém toda a informação lógica de como lidar com os pedidos provenientes da componente *View*, tendo como elo de ligação a componente *Controller* que é responsável por controlar como toda a informação é gerida.

4.1.2 Facade

Este padrão arquitetural foi implementado com vista a organizar e simplificar todo o sistema desenvolvido no que concerne à componente *back-end*. Deste modo, podemos retratar a *back-end* duma forma simples, dado que dispomos de uma interface provida deste padrão. Assim, no que diz respeito à *back-end*, o *Facade* é o objeto responsável por disponibilizar a interface referente à classe principal da componente. Deste modo, são acedidas todas as funcionalidades da *back-end* de forma simplificada. Esta interface denomina-se *IGestorFacade*, que disponibiliza todas as funcionalidades da classe *Gestor*. Para mais detalhe, aconselhamos a visualização da figura 5.3

4.1.3 Proxy

Este padrão arquitetural teve de ser implementado, dado que sem o mesmo, não seria possível tratar os pedidos de um qualquer utilizador. Isto deve-se ao facto de que, aquando da realização da primeira fase, os pedidos do utilizador eram recebidos diretamente na componente *back-end*, o que acabou por trazer adversidades quando foi implementada a componente *front-end* nesta segunda fase, sendo esta componente responsável por receber os pedidos do utilizador. Assim, o padrão foi implementado com vista a mitigar este problema, no qual foi criado um componente intermediário, tal como visto na figura 3.1, que serve de ponte de ligação entre os pedidos e respostas entre os dois principais componentes já mencionados. Assim, permite

redirecionar e encapsular do utilizador, todos os seus pedidos que são recebidos pela *front-end*, sendo posteriormente entregues à *back-end*, nomeadamente ao componente responsável por lidar com todos os pedidos recebidos no sistema.

4.1.4 Observer

Este padrão arquitetural foi utilizado para implementar as notificações do sistema. Este padrão foi utilizado em duas funcionalidades sendo um a de um Apostador seguir um jogo, onde o Apostador é notificados sempre que as odds são alteradas ou o score. A outra funcionalidade é dos seguidores de um apostador onde recebem uma notificação sempre que o apostador que segue realiza uma aposta, onde o Observer e o Observed são o Apostador.

4.1.5 Adapter

Utilizado para conseguir adaptar o resultado das chamadas à API caso ela tenha sido alterada.

4.2 Tecnologias

Além dos padrões arquiteturais, convém mencionar todas as tecnologias presentes e necessárias para o desenvolvimento dos mesmos. Assim, podemos referir e de forma breve explicar a motivação do uso de tais tecnologias.

- **Java**: Esta linguagem de programação foi utilizada no desenvolvimento da *back-end*, uma vez que foi a linguagem adotada e que melhor justificava o seu uso dada a experiência do grupo de trabalho, bem como o benefício de ser uma linguagem orientada aos objetos.
- **Python**: Esta linguagem de programação foi utilizada no desenvolvimento do componente *Proxy*, componente este que serve de intermediário e que já foi detalhado anteriormente.
- **React.js**: Esta biblioteca foi utilizada no desenvolvimento da *front-end*, com o intuito de poder manipular o DOM mais facilmente, uma vez que queríamos que o programa desenvolvido manipula-se de forma dinâmica a página *web*.
- **HTML**: Esta linguagem foi utilizado no desenvolvimento da *front-end* com vista a criar toda a estrutura de suporte da página *web*.
- **CSS**: Esta linguagem serviu de suporte à supramencionada tecnologia, por forma a criar o estilo especificado pelo grupo, com vista a formatar o documento que diz respeito à página *web*.

5. Vista de bloco de construção

Dada a extensão do projeto desenvolvido e tendo em conta todas as vertentes envolventes e já previamente abordadas de forma breve, podemos agora sumarizar o projeto tendo em conta a vista de bloco de construção. Tal como fora mencionado no capítulo 3, a aplicação envolve diversos sistemas que formam um todo e cooperam entre si por forma a atingir os diversos objetivos. Deste modo, por forma a organizar o projeto, iremos demonstrar os diversos níveis de blocos da implementação.

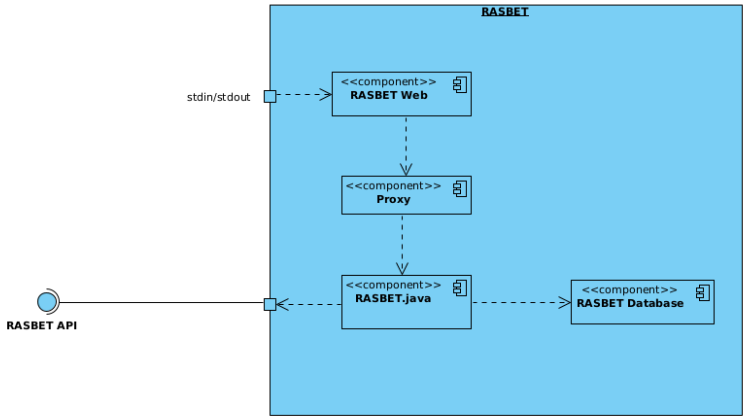


Figura 5.1: Diagrama de componentes (bloco de construção nível 1)

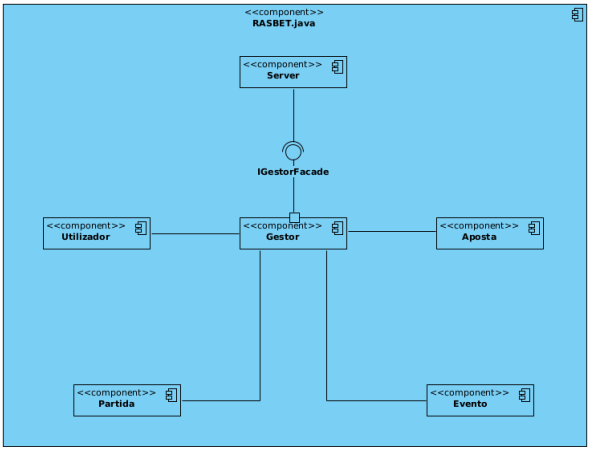


Figura 5.2: Diagrama de componentes (bloco de construção nível 2)

Uma vez abordados e detalhados todos os componentes que fazem parte da solução desenvolvida, no âmbito de complementar e implementar a aplicação já descrita, podemos dar ênfase a um dos sistemas, *RASBET.java*, sendo um dos componentes principais. Para tal, foi elaborado o seguinte diagrama de classes.

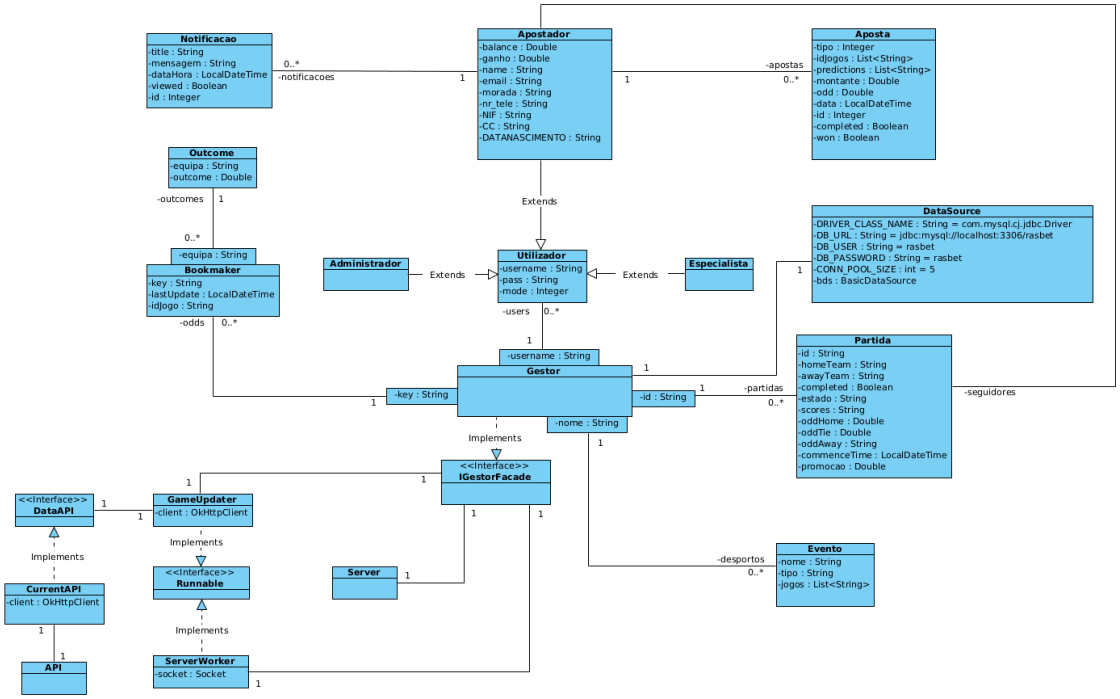


Figura 5.3: Diagrama de classes

No diagrama de classes supramencionado, é notório a presença de uma classe principal, a classe *Gestor* no qual a mesma possui a interface *iGestorFacade* responsável por contactar a classe *Server* que atende aos pedidos que vêm do componenete já mencionado, *Proxy.py* relativo ao pedidos do cliente. A classe *Gestor* agrega toda a informação necessária do sistema por forma a realizar as operações precisas para satisfazer o pedido obtido, através de estruturas de dados apropriadas, sob a forma de atributos. Esta classe alberga a informação sobre os utilizadores disponibilizada através da classe abstrata *Utilizador*, relativa ao tipo de utilizador existente no sistema. Assim, consoante o tipo de utilizador, existe uma classe apropriada, nomeadamente as classes *Administrador*, *Especialista* e *Apostador*. Convém salientar que juntamente à última classe mencionada, advém outras duas classes, *Notificacao* e *Aposta*, uma vez que este tipo de utilizador acresce de funcionalidades, requerendo a criação das classes mencionadas, agregando a informação relativa a notificações e apostas respetivamente. Além disto, a classe *Gestor* contém informação relativa a eventos e às partidas dos mesmos, através das classes *Evento* e *Partida* respetivamente. Convém mencionar também que a informação do sistema usada pela classe principal, é fornecida pela classe *DataSource* responsável por aceder ao componente *RASBET Database*. Por fim, temos também a classe *Bookmaker* no qual é acedida sempre que um utilizador, nomeadamente especialista, tem de atualizar/acrescentar informação relativa a um evento desportivo, por forma a puderem ser realizadas apostas, uma vez que agrega essas informações provenientes da API externa *RASBET API*.

6. Visualização em tempo de execução

Através do uso de diagramas de sequência vamos ilustrar como as diferentes partes do nosso sistema interagem entre si, percebendo a ordem em que as interações ocorrem para cada *use case* que é executado no nosso serviço. Assim, fica definida uma melhor visão da lógica de cada processo.

6.1 Registo

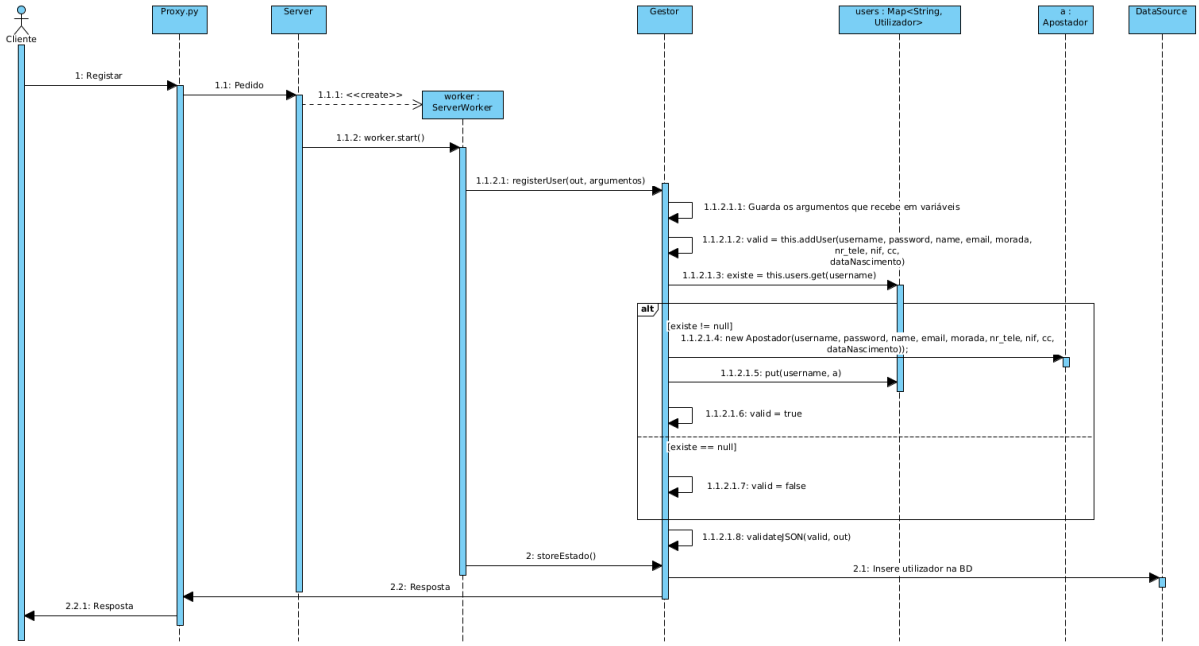


Figura 6.1: Diagrama de sequência do registo de um utilizador

Quando o utilizador se pretende registar tem de selecionar o botão *Sign up* presente na página inicial da aplicação. Após isto são lhe exigidos vários dados pessoais, como o username, nome, email, morada, número de telemóvel, data de nascimento, cartão de cidadão e o nif. Após a inserção dos dados é necessário que o cliente pressione o botão *Register* para que estes sejam recebidos pelo proxy que envia o pedido para o servidor, onde o seu funcionamento está explícito na figura 6.1.

6.2 Login

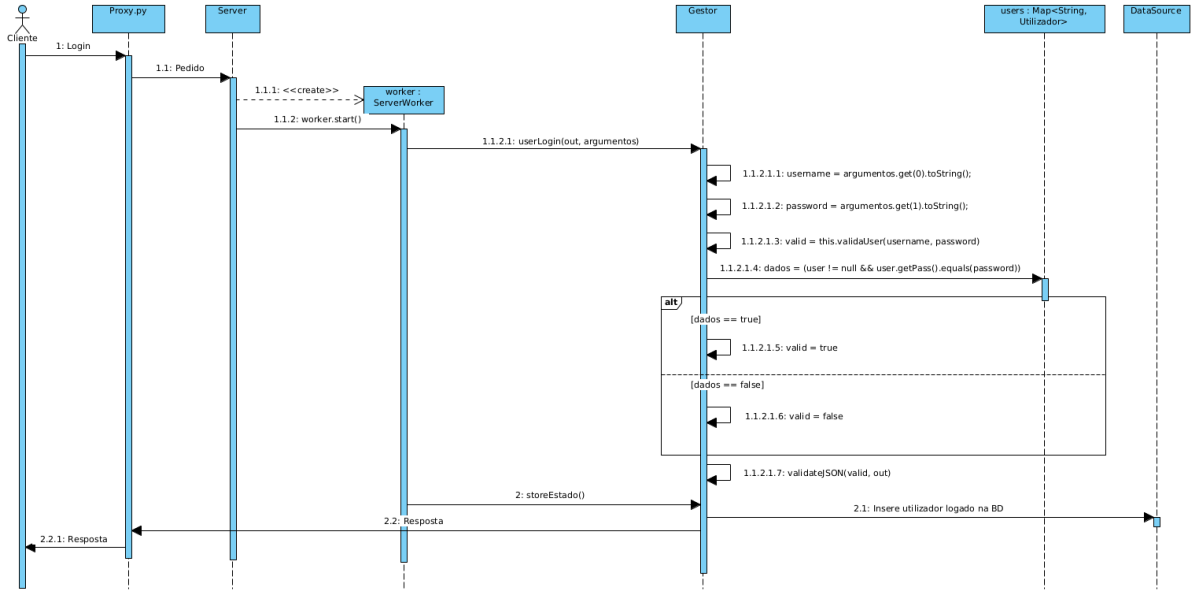


Figura 6.2: Diagrama de sequência do login de um utilizador

Na página inicial é necessário preencher os campos *Username* e *Password* e pressionar o botão *Login* que gera o pedido que o proxy envia para o servidor, explicado na figura 6.2. Caso a autenticação seja aceite pelo sistema, o utilizador é reencaminhado para a página principal onde poderá usufruir das funcionalidades do serviço.

6.3 Alterar informações do perfil

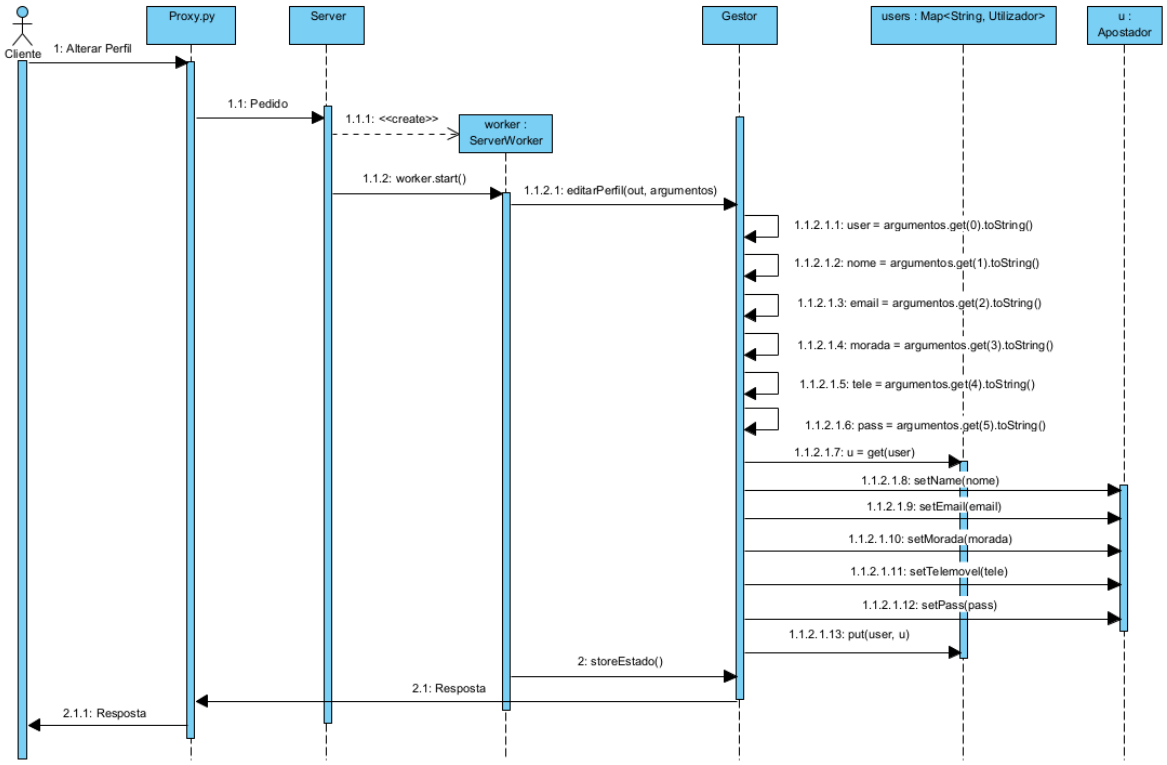


Figura 6.3: Diagrama de sequência de alterar informações do perfil

Na página principal é necessário selecionar o botão *Perfil* que listará várias opções das quais se opta pelo botão *Ver Informações*. Após isto são apresentadas todas as informações pessoais do utilizador bem como uma imagem que é possível selecionar para poder editar algumas das informações pessoais do utilizador. Sublinha-se que o username, data de nascimento, o cartão de cidadão e o nif são imutáveis. O pedido é gerado quando é selecionado o botão *Guardar*.

6.4 Consultar jogos

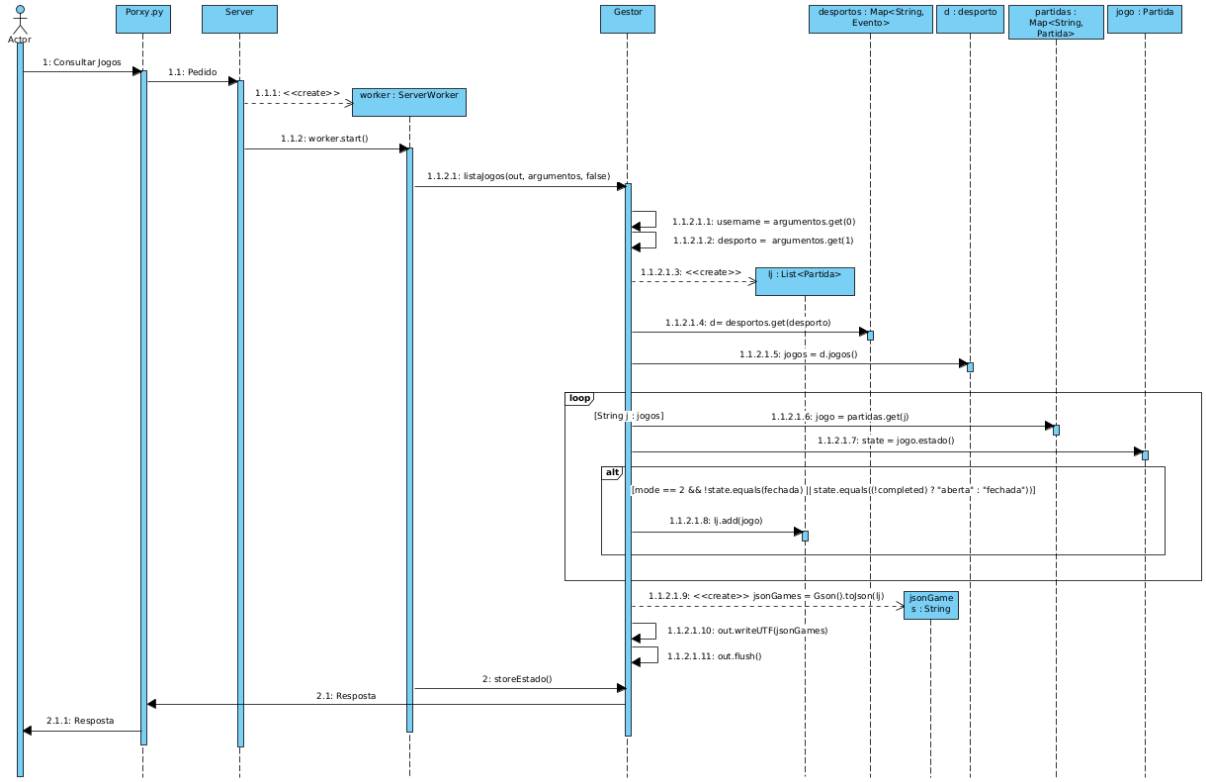


Figura 6.4: Diagrama de sequência de consultar jogos

Quando o utilizador pretende consultar o histórico de jogos terminados basta ir ao botão Perfil na sua página principal e selecionar a opção *Histórico de Jogos*. O proxy encaminhará este pedido para o servidor.

6.5 Fazer aposta

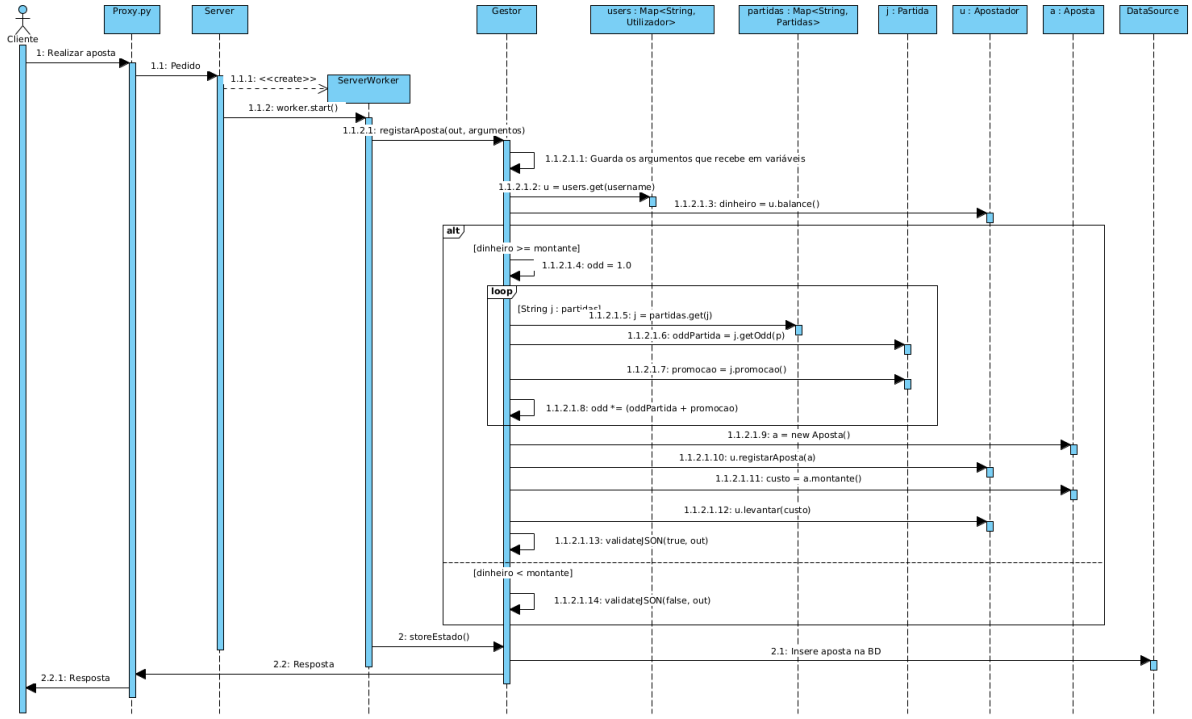


Figura 6.5: Diagrama de sequência de realizar uma aposta

Para efetuar uma aposta, o utilizador seleciona da página principal quais os jogos em que pretende apostar criando um boletim e em seguida é necessário introduzir o montante a apostar. O pedido é gerado após este selecionar o botão *Apostar*.

6.6 Consultar histórico de apostas

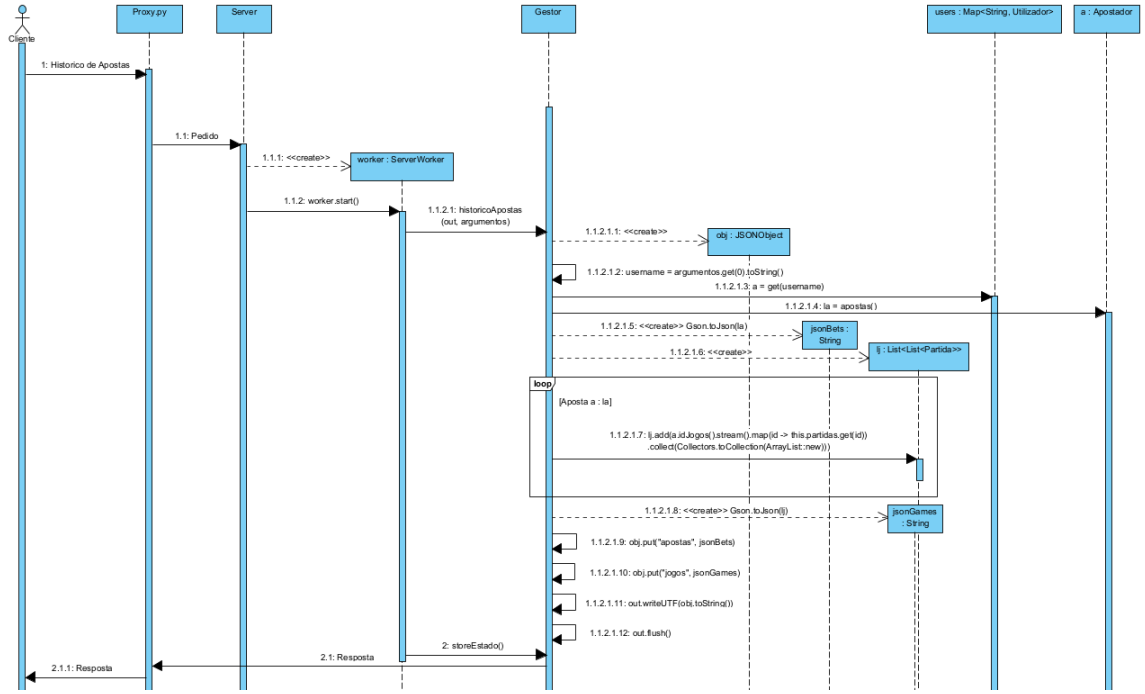


Figura 6.6: Diagrama de sequência de consultar histórico de apostas

Quando o utilizador pretende consultar o seu histórico de apostas basta ir ao botão *Perfil* na página inicial e selecionar a opção *Histórico de apostas*. O proxy encaminhará este pedido para o servidor.

6.7 Depositar dinheiro

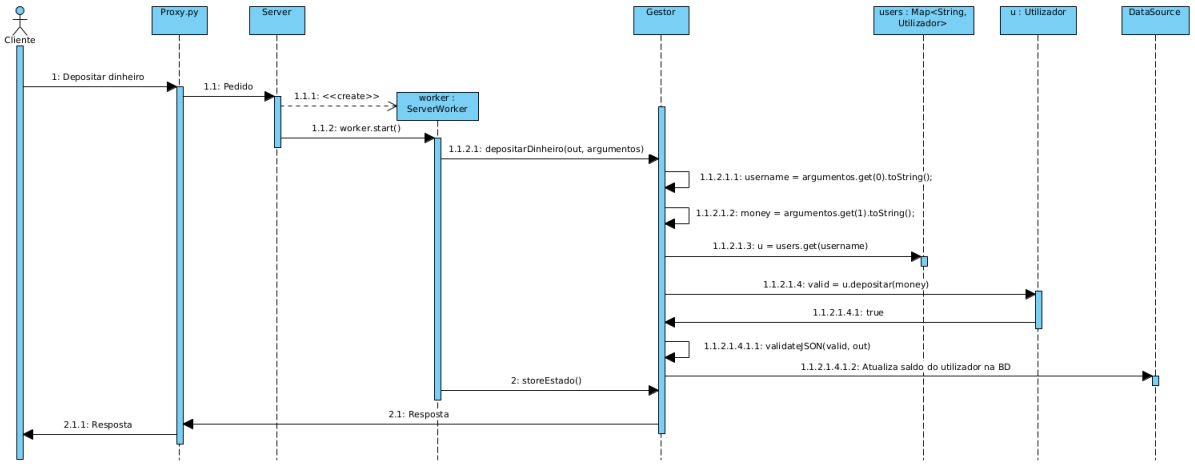


Figura 6.7: Diagrama de sequência de depositar dinheiro

Caso o utilizador pretenda depositar dinheiro é necessário selecionar o botão *Perfil* da página principal e de seguida selecionar *Despositar dinheiro* da lista que lhe é apresentada. Depois terá de optar por um dos três métodos disponíveis de depósito que são *MasterCard, MbWay e PayPal*. No primeiro caso, o utilizador tem de introduzir o nome do cartão, número,

cvv, data de validade. No segundo caso, é apenas exigido o número de telemóvel. No terceiro caso são pedidos o email, a password. É comum a todos os métodos introduzir a quantidade que o utilizador pretende depositar. O pedido é recebido pelo proxy quando é selecionado o botão confirmar que o redireciona para o servidor.

6.8 Levantar dinheiro

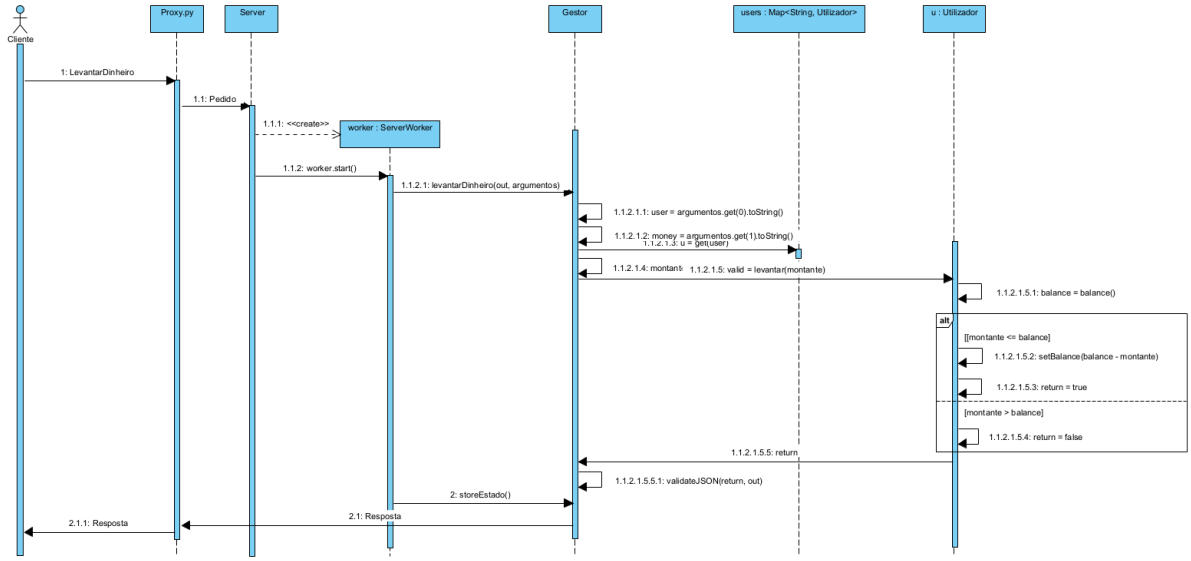


Figura 6.8: Diagrama de sequência de levantar dinheiro

Caso o utilizador pretenda levantar dinheiro é necessário selecionar o botão *Perfil* da página principal e de seguida selecionar *Levantar dinheiro* da lista que lhe é apresentada. O método de levantamento é igual ao de depósito de dinheiro que está explicado na secção anterior.

6.9 Inserir odd

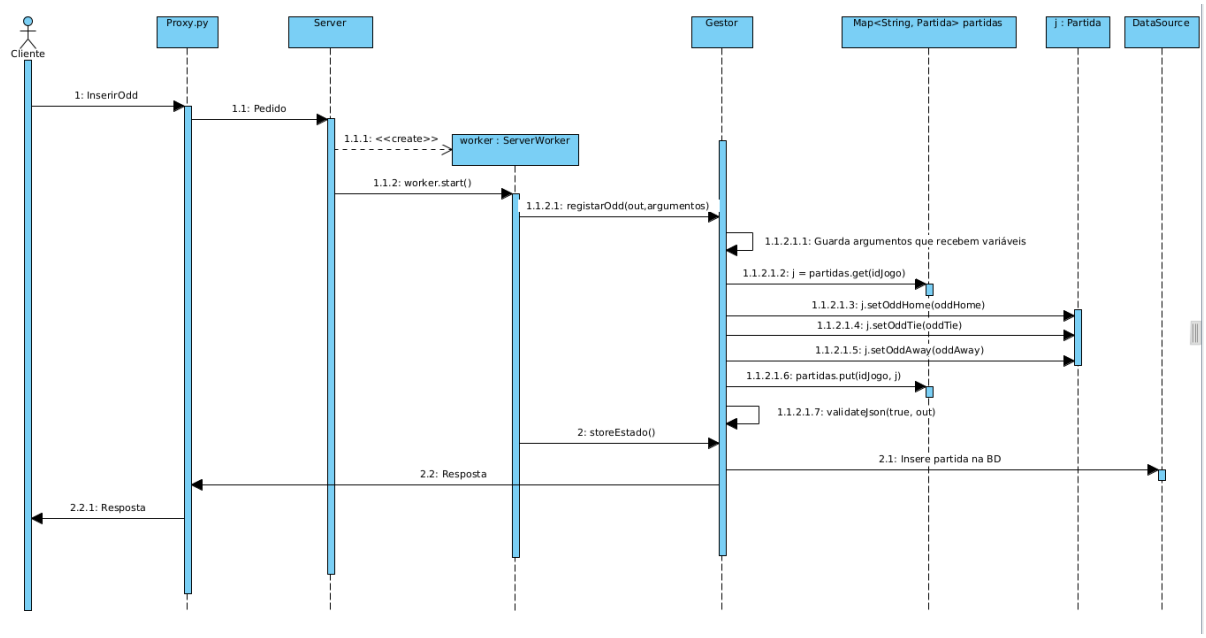


Figura 6.9: Diagrama de sequência de inserir odd

Neste caso, o cliente é definido como especialista. Quando este pretende definir odds de uma dada partida, na sua página principal basta escrever a odd que pretende atualizar para um dado jogo e selecionar o botão *Atualizar Odd* que irá dar origem ao pedido que irá ser reencaminhado para o servidor pelo proxy.

6.10 Alterar estado da aposta

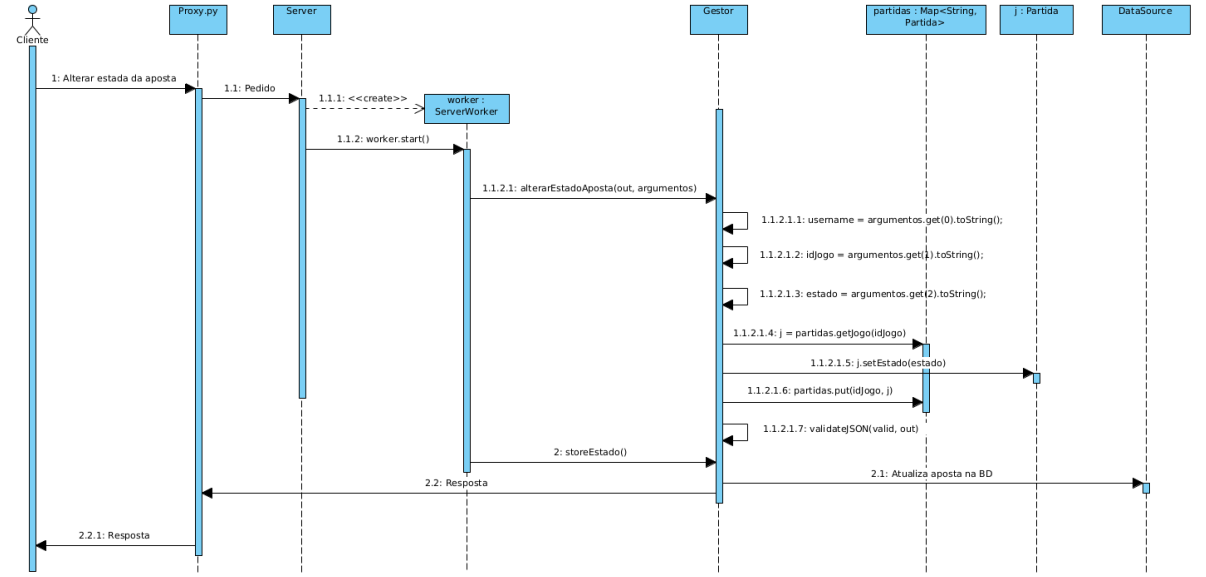


Figura 6.10: Diagrama de sequência de alterar estado da aposta

Em relação ao estado de uma aposta esta é tratada pelo administrador. Na sua página

principal basta ir ao respetivo jogo e selecionar um dos três estados disponíveis, abrir fechar ou suspender. Após isto é necessário confirmar o estado selecionando o botão *Confirmar estado* que irá ser responsável pela criação do pedido. Este pedido é enviado pelo proxy para o servidor.

7. Visualização de implementação

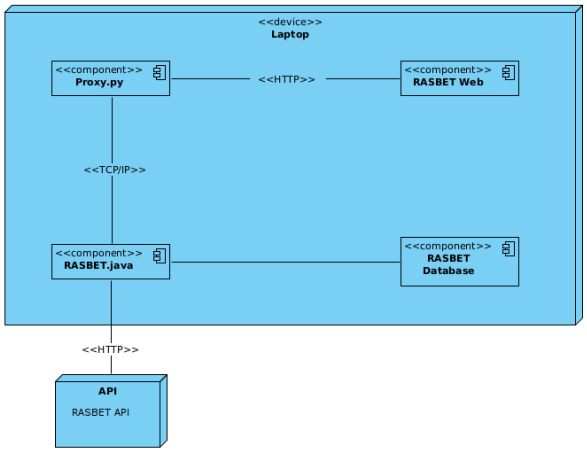


Figura 7.1: Diagrama de implementação

Através do diagrama descrito, percebemos que o nodo *RASBET Web* contacta através de *HTTP* com o nodo *Proxy.py* no qual este envia e recebe mensagens através de *TCP/IP*, com o nodo *RASBET.java*. Este conecta-se ao nodo *RASBET Database* que representa a base de dados *MYSQL* e contacta com o nodo *RASBET API* através de *HTTP*, que representa a API externa. Neste momento a aplicação está a correr apenas localmente.

Além disso, convém salientar para cada componente acima mencionado, as dependências necessárias por forma a todos os componentes envolvidos conseguirem realizar as tarefas necessárias e cooperarem entre si, de modo a compactuarem com o serviço proposto da aplicação.

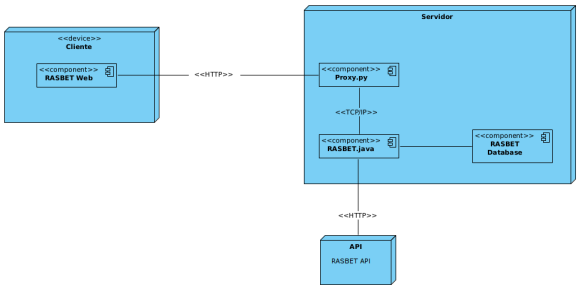


Figura 7.2: Diagrama de implementação

Numa versão futura da implementação, o grupo pretende criar um servidor, que iria tratar da base de dados e dos dados recebidos pela API. Os clientes entrariam em contacto com o servidor através do protocolo HTTP, sendo que o servidor trataria destes pedidos e respondia ao cliente, sendo este capaz de apresentar os conteúdos recebidos ao utilizador da aplicação.

Nome	Versão
MySQL	8.0.31
OkHttp	4.1.0
JSON	20190722
Gson	2.8.5
Apache Commons DBCP	2.7.0
Java	19.0.1

Tabela 7.1: Dependências necessárias para o componente *RASBET.java*

Nome	Versão
jpysocket	1.1.5
pip	20.0.2
pkg-resources	0.0.0
setuptools	44.0.0
websockets	10.4
Python	3.8.10

Tabela 7.2: Dependências necessárias para o componente *Proxy.py*

Nome	Versão
npm	9.1.3
Node js	18.12.1
React	18.2.0
CSS	3
HTML	5

Tabela 7.3: Dependências necessárias para o componentes *RASBET Web*

Para o componente *RASBET Database* é utilizada a versão 8.0.27 (MySQL Community Server - GPL) do MySQL.

Requisitos mínimos recomendados para suportar a aplicação no sistema operativo Linux (Ubuntu 20.04):

- Processador: Intel Core i5-8265U
- Placa gráfica (Integrada): Intel UHD Graphics 620
- Placa gráfica (Dedicada): NVIDIA GM108M [GeForce MX110]
- Memória RAM: 12GB

Requisitos mínimos recomendados para suportar a aplicação no sistema operativo Windows:

- Processador: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
- Placa gráfica (Integrada): Intel (R) HD Graphics 620
- Placa gráfica (Dedicada): NVIDIA GeForce 920MX
- Memória RAM: 12,0 GB

8. Alterações Realizadas

8.1 Base de Dados

Para a base de dados apenas foi criada uma nova tabela chamada "GameFollow" que serve como tabela de chaves estrangeiras para ligar as Partidas e os seus seguidores, seguindo a mesma lógica utilizada para a implementação dos seguidores de Apostadores.

8.2 Servidor

8.2.1 Seguir Partidas

Para poder suportar a nova funcionalidade pedida de um Apostador poder seguir Partidas foi modificada a Classe 'Partida' para conter uma Lista de Apostador que a seguem, notificando-os sempre que alguma alteração é realizada nas odds ou no resultado da partida. Foi também adicionado ao "Apostador" métodos para notificarem ao apostador sobre essas mudanças (*updateScore()* , *updateOdd()*).

Para conseguir executar estas novas funcionalidades foi acrescentado ao 'Gestor' e ao 'IGestorFacade' os metodos *getPartidasSeguidas()*,*seguirPartida()* e *naoseguirPartida()*.

8.2.2 Adapter

Foram também criadas 3 classes novas: a Interface "DataAPI", que contém apenas o método *fetchdata()*; a classe "CurrentAPI" que contém a API atual a ser utilizada, que implementa o *fetchdata()*, onde esta apenas faz um pedido à API atual e utiliza o método *fetch()* da Classe "API" onde transforma o que é obtido pela chamada a API para um formato que o sistema suporta.

8.3 Cliente Web

8.3.1 Apostas Múltiplas

Esta funcionalidade já estava presente na nossa implementação logo não foram necessárias novas mudanças, com exceção do limite de 20 partidas por aposta múltipla que foi implementada do lado do Cliente Web.

8.3.2 Seguir Partidas

Foi criado um novo ficheiro HTML *jogosSeguidos* para criar uma nova página que ilustre apenas os jogos atuais que um Apostador segue. Foi alterado o *page_apostador* para conter agora um icon para que um apostador possa seguir ou deixar de seguir um jogo.

8.4 Refactoring

Aquando a realização da fase anterior foi já efetuada o refactoring do código, em paralelo com cada passo da implementação. O que tornou a introdução da novas funcionalidades pedidas nesta fase uma tarefa muito mais acessível e como o código já seguia uma estrutura relativamente limpa facilitou no refactoring do novo código adicionado.