

# UMinho

Mestrado em Engenharia Informática

Requisitos e Arquiteturas de Software

Grupo 4, PL6 / Entrega 2

PG50327 Diogo Rebelo  
PG50310 Daniel Xavier  
PG50537 Júlio Gonçalves  
PG50414 Henrique Alvelos  
PG50469 João Cerquido



20 de junho de 2023

2022/23

# Prefacio

Ao longo do desenvolvimento do projeto, foram encontradas várias dificuldades, quer em termos de toma de decisões, quer em termos de implementação da solução para o produto em questão. Primeiro, importava compreender quais os diagramas realmente necessários para modelar com sucesso a arquitetura da solução, posteriormente, era necessário perceber de que forma esta arquitetura espelhava a fase de especificação anterior. Para além disso, a principal dificuldade prendeu-se com a conexão entre as entidades principais do sistema, o *Back-end* e o *Front-end*, através da utilização das tecnologias mais adequadas.

Urge, agora, destacar alguns aspectos relativos à organização do presente documento. Então, este introduz-se pelos objetivos que definem o projeto, dando especial foco ao seu contexto de desenvolvimento; passa-se para a sua modelação a nível de abstração decrescente, até que se faz referência a requisitos de qualidade e alguns riscos associados ao desenvolvimento do projeto.

Finalmente, mostra-se relevante especificar o contributo acrescido (+), decrescido (-) ou equilibrado (0), de cada elemento do grupo, para o desenvolvimento do projeto:

<b>Diogo Rebelo</b>	0	<b>Duarte Cerquido</b>	0	<b>Júlio Beites</b>	0
<b>Henrique Alvelos</b>	0	<b>Daniel Xavier</b>	0		

# Conteúdo

<b>1</b>	<b>Introdução e Objetivos</b>	<b>6</b>
1.1	Visão Geral dos Requisitos . . . . .	6
1.2	Metas de Qualidade . . . . .	8
<b>2</b>	<b>Restrições de Arquitetura</b>	<b>9</b>
<b>3</b>	<b>Contexto e Âmbito</b>	<b>10</b>
3.1	Contexto de Negócio . . . . .	10
3.2	Contexto Técnico . . . . .	11
<b>4</b>	<b>Estratégias de Solução</b>	<b>12</b>
4.1	Objetivos de Qualidade . . . . .	12
4.2	Estruturação do RASBet . . . . .	13
<b>5</b>	<b>Modelação do Bloco de Construção</b>	<b>14</b>
5.1	Diagrama de Classes - Nível 2 . . . . .	14
5.2	Divisão das Responsabilidades . . . . .	15
5.3	Modelo Lógico . . . . .	16
<b>6</b>	<b>Modelação da Execução</b>	<b>17</b>
6.1	Gestão de Conta . . . . .	17
6.1.1	Iniciar sessão . . . . .	17
6.1.2	Alterar Informações de Perfil . . . . .	18
6.2	Registros . . . . .	19
6.2.1	Registrar . . . . .	19
6.3	Gestão Bancária . . . . .	19
6.3.1	Depositar Dinheiro . . . . .	19
6.4	Gestão de Jogos, Odds, Apostas . . . . .	20
6.4.1	Fazer Aposta . . . . .	20
6.4.2	Criar Promoções . . . . .	20
6.4.3	Alterar Estado da Aposta . . . . .	21
6.4.4	Alterar Odd . . . . .	21
6.5	Consultas . . . . .	21
6.5.1	Consultar Jogos . . . . .	21
6.5.2	Consultar Histórico de Transações . . . . .	22
6.5.3	Consultar Histórico de Apostas . . . . .	22
6.5.4	Consultar Promoções . . . . .	22
6.5.5	Consultar Notificações . . . . .	23
6.6	Supporte Técnico . . . . .	23
6.6.1	Fazer Pedidos de Ajuda . . . . .	23
6.6.2	Responder a Pedidos . . . . .	23
6.7	Alertas e Notificações . . . . .	24

6.7.1	Enviar Notificações . . . . .	24
<b>7</b>	<b>Modelação de <i>Deployment</i></b>	<b>25</b>
<b>8</b>	<b>Requisitos de Qualidade</b>	<b>26</b>
<b>9</b>	<b>Riscos e Dívida Teórica</b>	<b>29</b>
<b>10</b>	<b>Glossário</b>	<b>31</b>

# Listas de Figuras

3.1	Diagrama genérico com os vários <i>stakeholders</i> em cada contexto. . . . .	10
3.2	Diagrama de Contexto do Sistema. . . . .	11
5.1	Diagrama de Classes . . . . .	14
5.2	Modelo lógico . . . . .	16
6.1	Diagrama de Sequência do Início de Sessão. . . . .	17
6.2	Diagrama de Sequência da Alteração de informações de perfil. . . . .	18
6.3	Diagrama de Sequência do Registo. . . . .	19
6.4	Diagrama de Sequência do Depósito de Dinheiro. . . . .	19
6.5	Diagrama de Sequência de Fazer uma Aposte. . . . .	20
6.6	Diagrama de Sequência da Criação de Promoções. . . . .	20
6.7	Diagrama de Sequência da Alteração do Estado do Boletim de Jogo. . . . .	21
6.8	Diagrama de Sequência da Atualização da <i>Odd</i> . . . . .	21
6.9	Diagrama de Sequência da Consulta de Jogos. . . . .	21
6.10	Diagrama de Sequência da Consulta do Histórico de Transações. . . . .	22
6.11	Diagrama de Sequência da Consulta do Histórico de Apostas. . . . .	22
6.12	Diagrama de Sequência da Consulta de Promoções. . . . .	22
6.13	Diagrama de Sequência da Consulta de Notificações. . . . .	23
6.14	Diagrama de Sequência do Pedido de Ajuda. . . . .	23
6.15	Diagrama de Sequência da Resposta ao Pedido de Ajuda. . . . .	23
6.16	Diagrama de Sequência do Envio de Notificações pelo Administrador. . . . .	24
7.1	Diagrama de Implantação para cada cenário. . . . .	25
8.1	Árvore de Qualidade. . . . .	26

# Listas de Tabelas

1.1	Síntese de Requisitos.	7
1.2	Metas de qualidade especificadas.	8
2.1	Resumo das restrições de arquitetura.	9
4.1	Algumas estratégias de solução para os objetivos especificados.	12
5.1	Divisão de Responsabilidades.	15

# 1. Introdução e Objetivos

O presente documento contempla a segunda fase do trabalho prático elaborado. É, mais propriamente um documento de arquitetura para a aplicação RASBet, especificada na fase anterior. Globalmente, o documento procura elucidar aspectos importantes concretos sobre a implementação da aplicação em questão, começando pelas restrições de arquitetura, passando pelas estratégias de solução, respetivas modelações e conceitos, decisões de arquitetura, e acabando nos requisitos de qualidade. Alguns capítulos foram adicionados para maior coesão no documento.

Tendo sempre presente a funcionalidade principal da efetuação de apostas pelos utilizadores, o nosso público-alvo, é importante mencionar, nesse sentido, os objetivos que se pretendem ver cumpridos, no final da realização deste documento, em termos aplicacionais, tais como:

- Existência de uma arquitetura que facilite a adição de novos métodos, em caso de modificações futuras;
- Existência de uma divisão implícita dos vários componentes da aplicação que evidencie um parcelamento entre as várias funcionalidades do sistema;
- Existência de um método de atualização constante no sistema, em termos de armazenamento, gestão, apresentação e notificação dos respetivos utilizadores;
- Destaque à utilização de *design patterns* relevantes (nomeadamente, padrão *observer*, desde logo presente no término de um jogo e posterior notificação dos apostadores);
- Documentação que reflita de um modo geral a aplicação a ser futuramente implementada.

## 1.1 Visão Geral dos Requisitos

Os requisitos foram identificados e criteriosamente explicados no relatório de especificação da primeira fase, mais propriamente no capítulo de *Requisitos Funcionais* e *Requisitos Não Funcionais*. Este documento da primeira fase consistiu num documento de especificação de requisitos e, por isso, analisa todo o âmbito do projeto, seus intervenientes, as funcionalidades do sistema e o modo como este se relaciona com o exterior, as restrições que limitam este sistema enquanto requisitos não funcionais e toda a sua projeção mesmo em termos de interfaces. Neste contexto, tem-se uma secção que resume cada tipo de requisitos especificados, já que foram o foco da fase anterior. Trata-se de uma síntese, já que o resumo dos requisitos já foi acrescentado ao documento de requisitos.

<b>Tipo de Requisito</b>	<b>Count</b>	<b>Secção</b>	<b>Descrição</b>
<b>Funcional</b>	23	Utilizadores	O utilizador deve conseguir registar-se na aplicação O Utilizador deve conseguir iniciar sessão. O Utilizador deve conseguir terminar sessão. O Utilizador deve conseguir consultar a lista dos jogos.
			O Apostador deve conseguir editar o seu perfil. O Apostador deve conseguir apostar em qualquer jogo de qualquer desporto O Apostador deve conseguir consultar o seu histórico de apostas realizadas. O Apostador deve conseguir consultar o histórico de transações monetárias da sua conta.
			O apostador deve poder depositar dinheiro na sua carteira. O apostador deve poder levantar dinheiro da sua carteira. O apostador deve poder consultar as promoções existentes. O apostador deve poder consultar as notificações recebidas. O apostador deve poder pedir ajuda de um técnico.
			O Especialista deve poder inserir a Odd para um jogo. O Especialista deve poder atualizar a Odd de um jogo.
		Técnico	O Técnico deve poder responder a pedidos de ajuda dos Apostadores.
		Administrador	O Administrador deve poder alterar o estado de uma aposta. O Administrador deve poder enviar notificações a qualquer Utilizador. O Administrador deve poder criar promoções.
			O sistema deve garantir que nenhum utilizador tem acesso a outros perfis de utilizador. A aplicação subtraí o valor apostado do saldo da carteira. Um jogo tem obrigatoriamente um de 3 estados, Aberto, Fechado, Suspenso. O sistema tem de notificar os apostadores dos resultados das suas apostas.
			O sistema deve possuir, nas suas múltiplas plataformas, o seu logótipo representado. O sistema deve cativar visualmente fãs de desporto. O sistema deve distinguir-se visualmente dos produtos no mesmo nicho.
		Usabilidade	O sistema deve suportar várias línguas. O sistema deve suportar várias moedas O sistema deve ser capaz de converter o horário de um determinado jogo, de acordo com a time-zone do apostador. O sistema deve ser user-friendly, sendo intuitivo e prático, especialmente para apostadores que não estejam muito familiarizados com sites de apostas.
			O sistema deve possuir um uptime alto. O sistema deve atualizar rapidamente odds e resultados. O sistema deve guardar e processar os dados de (de)crescentes quantidades de utilizadores sem comprometer o funcionamento normal do sistema. O sistema deve apresentar uma taxa de utilização crescente nos 2 anos seguintes.
			O sistema deve ser uma plataforma web (site). O sistema deverá guardar os dados dos utilizadores numa Base de Dados. O sistema deverá respeitar o modelo arquitetural MVC (Modelo Vista-Controlador)
			O sistema deve ser capaz de prestar suporte técnico ao utilizador, em caso de necessidade O sistema deve permitir uma correção eficaz e prática de bugs O sistema deve permitir atualizações regulares em alturas de pouca utilização.
<b>Não Funcional</b>	26	Performance	O sistema deve validar a informação submetida, impedindo-a de ser armazenada, caso esteja incorreta. O sistema deve garantir que apenas os especialistas registados geram as odds de um resultado. O sistema deve informar os utilizadores sobre a alteração das políticas de privacidade.
			O sistema deve estar registrado como uma casa de apostas legal na entidade local responsável. O sistema deve ser independente no que diz respeito à religião e política. O sistema deve estar em conformidade com as leis definidas regras de um jogo de fortuna ou azar.
			O sistema deve ser apenas utilizado por apostadores maiores de idade, ou seja, com idade igual ou superior a 18 anos. O sistema deve possuir apostas que funcionem de acordo com os requisitos legais definidos pelas entidades responsáveis.

**Tabela 1.1:** Síntese de Requisitos.

## 1.2 Metas de Qualidade

Apresentaremos as cinco métricas de qualidade com maior prioridade, estando esta explicitamente identificada e em sintonia com as prioridades também identificadas no documento da fase anterior. A identificação das principais metas foi feita através da análise da prioridade do requisito. Então, de acordo com os requisitos não funcionais, destaca-se:

Prioridade	Meta	Descrição
Must	Atualização rápida de informações	O sistema deve atualizar rapidamente odds e resultados.
Must	Escalabilidade do sistema perante /diminuição/aumento de utilizadores	O sistema deve guardar e processar os dados de (de)crescentes quantidades de utilizadores sem comprometer o funcionamento normal do sistema.
Must	Formato web	O sistema deve ser uma plataforma web (site).
Must	Gestão responsável da informação	O sistema deve garantir que apenas os especialistas registados geram as odds de um resultado.
Must	Maior de Idade	O sistema deve ser apenas utilizado por apostadores maiores de idade, ou seja, com idade igual ou superior a 18 anos.

Tabela 1.2: Metas de qualidade especificadas.

Consideram-se estas metas indispensáveis para a implementação do sistema. Então, no sentido de justificar a nossa escolha, tenha-se em conta o seguinte:

- É fundamental que apenas o especialista seja capaz de alterar as *odds* de apostas, de modo a garantir o correto e seguro funcionamento do programa;
- Por outro lado, é também de grande importância que as *odds* sejam atualizadas de forma rápida e eficaz pelo sistema, quando submetidas pelo especialista, de forma a apresentar a informação aos apostadores em tempo próximo do real;
- O formato *web* é super relevante já que é o que permite aos utilizadores acederem à aplicação, em qualquer dispositivo conectado à *internet*;
- Finalmente, os utilizadores devem ser maior de idade de modo a cumprir a restrição legal a que o nicho das apostas se limita.

## 2. Restrições de Arquitetura

As restrições de arquitetura, tal como as restrições de limite técnico e limite organizacional foram identificadas e descritas no relatório da primeira fase. As restrições obrigatórias mais próximas do domínio da solução surgem no capítulo de *Restrições do Projeto*, na secção de *Restrições Obrigatórias*, já as restrições que se aproximam mais da gestão do projeto, como prazos e orçamentos, constam no capítulo de *Project Issues*, mais propriamente nas secções de *Tasks* e *Costs*.

<b>Restrições Obrigatórias</b>
<b>Ambiente de Implementação de Solução</b>
RO#1 - O sistema deve possuir um repositório no Github
RO#2 - O sistema deve estar conectado à Internet
<b>Gestão do Projeto - Orçamento</b>
RO#3 - Os custos associados ao projeto não devem ser superiores a 20 000 €
<b>Gestão do Projeto - Prazo</b>
RO#4 - Os prazos de desenvolvimento e entrega do projeto devem ser respeitados, de acordo com as três fases.

**Tabela 2.1:** Resumo das restrições de arquitetura.

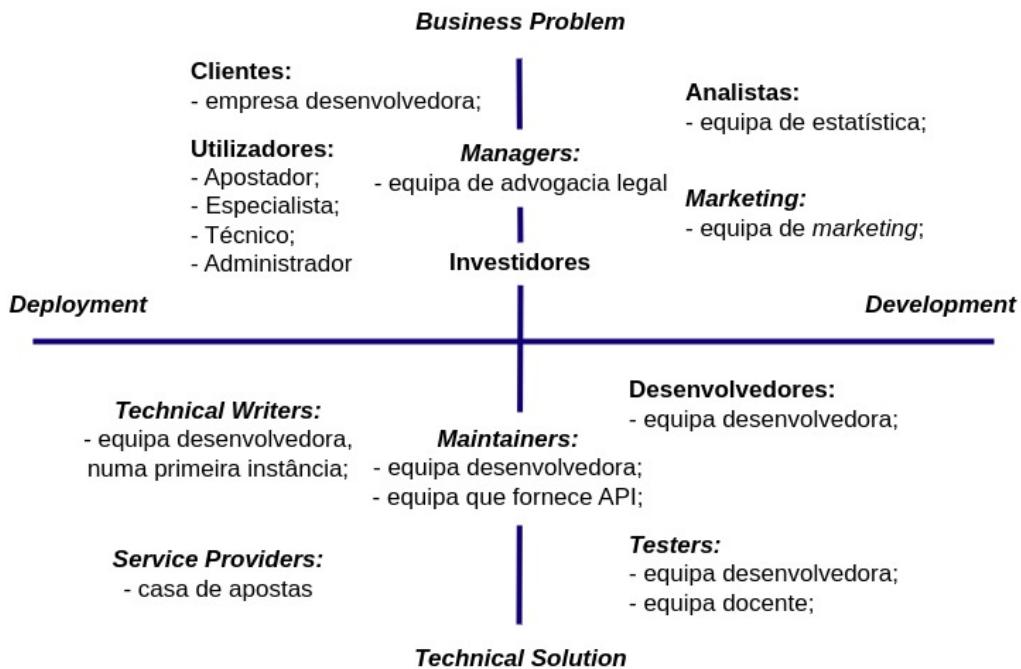
Para além disso, também era obrigatória a recorrência à API disponibilizada pela equipa docente, para a obtenção dos dados dos Jogos e Desportos. Trata-se igualmente de uma restrição obrigatória que diz respeito ao Ambiente de Implementação da Solução.

### 3. Contexto e Âmbito

Neste capítulo, abordaremos o âmbito e o contexto do sistema, permitindo assim delimitar o sistema ao nível do seu âmbito e de todos os seus sistemas vizinhos (externos) e utilizadores (parceiros de comunicação). Para melhor compreensão desta secção, fornecemos um diagrama de contexto do sistema, após descrição dos contextos que fazem sentido, no nosso caso. As seguintes secções foram ambas abordadas na fase anterior: na *Contextualização*, já se referência o contexto, nos *Instigadores* e *Utilizadores de Produto*, faz-se alusão a interações importantes que, depois, são exploradas um pouco mais concretamente no capítulo de *Âmbito do Produto*, através do modelo de domínio, onde de forma genérica e ainda não muito limitativa se contextualiza o sistema.

#### 3.1 Contexto de Negócio

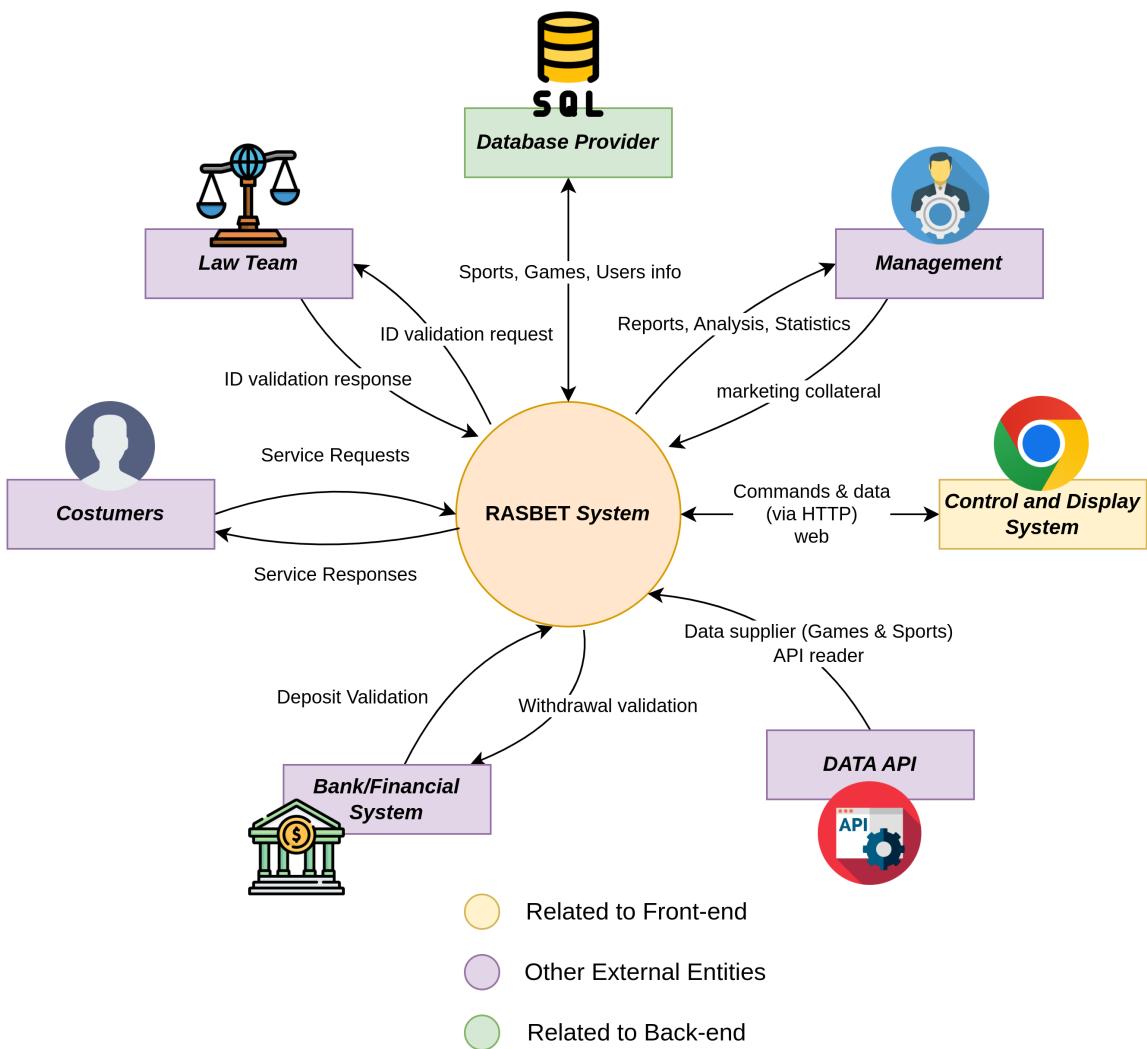
Nesta secção, abordamos o contexto do nosso sistema, delimitando o domínio do nosso sistema, as casas de apostas, a um conjunto limitado de funcionalidades, como por exemplo a existência de jogos para um determinado número de desportos ou o tipo de apostas a implementar. É sabido que o contexto de negócio se deve focar nas necessidades e expectativas da audiência em questão, logo, torna-se importante referenciar claramente e de forma genérica os *stakeholders*:



**Figura 3.1:** Diagrama genérico com os vários *stakeholders* em cada contexto.

## 3.2 Contexto Técnico

Nesta secção, abordamos o contexto técnico no nosso sistema, ou seja, os limites do sistema que evidenciam as fontes de informação vizinhas e externas à aplicação. Neste sentido, é importante realçar a existência de uma API externa providenciada pela equipa docente e por ela mantida, daí a referência no diagrama acima. Neste contexto, a equipa que fornece a API acaba por ser um dos importantes *maintainers* do bom funcionamento da aplicação, garantindo que ela se comporta de acordo com o previsto. Para além disso, outras entidades podem ser identificadas e, como forma de evidenciar ambos os contextos (de negócio e técnico), realizou-se o seguinte diagrama de contexto do sistema:



**Figura 3.2:** Diagrama de Contexto do Sistema.

É, então, possível identificar o conjunto das entidades externas ao sistema e que com este interagem, formando relacionamentos. Assim sendo, cada bloco retangular evidencia uma entidade externa e o círculo central evidencia o sistema. Os relacionamentos são representados por setas e clarificam o tipo de relação entre o sistema e essa entidade.

## 4. Estratégias de Solução

Neste capítulo, é relevante fazer um resumo das estratégias de solução que moldam a arquitetura. Consideramos relevante começar por expor as abordagens seguidas para atingir as metas de alta qualidade enunciadas anteriormente e o conjunto de decisões organizacionais relevantes.

### 4.1 Objetivos de Qualidade

Objetivo de qualidade	Cenário	Forma de Solucionar
<b>O sistema deve apenas ser utilizado por utilizadores maiores de idade.</b>	- Menor de idade não deve conseguir efetuar o seu registo no sistema, não deve criar uma conta no sistema.	- Sistema de registo que apenas possibilita a criação de contas a maiores de idade, através do confronto com a data de nascimento submetida. - Sistema de validação que averigue se o indivíduo tem mesmo 18 anos, através do confronto com o CC
<b>O sistema deve funcionar em formato web e em vários browsers.</b>	- Utilizador utiliza diferentes browsers para aceder à aplicação de apostas.	- Equipa de testes deve testar a aplicação em diferentes browsers. - Análise e Verificação do código HTML produzido pela equipa desenvolvedora.
<b>O sistema deve garantir a gestão responsável e segura da informação.</b>	- O Administrador altera o estado do jogo. - O Especialista altera as odds de um resultado. - O Apostador altera apenas as suas informações.	- Utilização de um sistema de logs que contenhas as alterações no sistema e os respetivos acessos de cada utilizador. - Utilização de sistemas de registo e utilização de credenciais, garantindo a integridade no acesso ao sistema.
<b>O sistema deve garantir a existência de atualizações rápidas da informação.</b>	- O Apostador efetua uma aposta, que é depois verificada. - O Administrador encerra um jogo, os apostadores desse jogo são notificados de imediato.	- Sistema que permite enviar notificações ao utilizadores. - Front-end que reflete este sistema de notificação. - Desenvolvimento de um Back-end eficaz que consiga rapidamente efetuar estas verificações.
<b>O sistema deve ser escalável em ambos os sentidos.</b>	- Aumento do número de utilizadores simultâneos, durante um jogo importante.	- Sistema que possua uma adaptação dinâmica dos recursos alocados.
<b>Existência de informação atualizada.</b>	- Atualização de odds. - Atualização de jogos. - Atualização de resultados.	- Sistema que seja capaz de ler, de entre um intervalo de tempo definido, a API, possuindo a informação atualizada de acordo com esta. - Sistema que atualize, dentro de um intervalo de tempo definido (e curto), as respetivas odds, caso estas tenham sofrido alterações. - Sistema que atualize, dentro de um intervalo de tempo definido (e curto), os respetivos jogos existentes (API).
<b>Existência de Interfaces simples, práticas e intuitivas na aplicação.</b>	- Utilizador navega facilmente pela lista de desportos/jogos.	- Separação de informação que é diferente por diferentes estruturas (classes, dados, etc.). - Front-end que possibilite esta fácil navegação, através de listas ou paginação.
<b>O Sistema deve estar preparado para sofrer alterações (expansões).</b>	- Administrador decide a adição de novos desportos ou tipos de apostas.	- Uso de mecanismos que contribuam para fácil expansão, como interfaces para as componentes principais do sistema.
<b>O sistema deve possuir acessibilidade.</b>	- Responsável futura do sistema consegue facilmente perceber, analisar e interpretar a arquitetura do sistema.	- Providenciar a solução arquitetural. - Utilização de modelos orientados a objetos e que facilitem essa interpretação. - Providenciar instruções de análise e interpretação na própria solução (documentação).

**Tabela 4.1:** Algumas estratégias de solução para os objetivos especificados.

## 4.2 Estruturação do RASBet

Nesta secção, contemplam-se aspectos estruturais importantes relativos ao sistema implementado. O sistema desenvolvido baseia-se num diagrama de classes não muito complexo, mas que acaba por ser representativo do programa a desenvolver e que, por isso, surge a seguir. Neste mesmo diagrama, é possível observar as classes mais importantes do sistema, das quais se destaca as que dizem respeito aos vários utilizadores do sistema, as responsáveis pela leitura da API e as que efetuam a persistência de dados.

Para além disso, a aplicação deverá, como sugerem também os diagramas de sequência a seguir, ser implementada em Java. Existem várias vantagens na utilização de um paradigma orientado aos objetos para a construção da solução, nomeadamente:

- **Integração:** A facilidade de integração de interfaces gráficas;
- **Familiarização:** Implementação torna-se mais fácil devido à familiarização da equipa desenvolvedora;
- **Transformação:** A facilidade de implementar através dos modelos e diagramas apresentados em todo este processo;
- **Portabilidade:** Permitindo a utilização do programa em várias plataformas e dispositivos;
- **Garbage Collector:** gestão automática de memória, permitindo uma alocação mais eficiente.
- **Documentação:** é extensa, por ser uma linguagem muito utilizada, permitindo resolver *bugs* mais rapidamente, com esse auxílio.

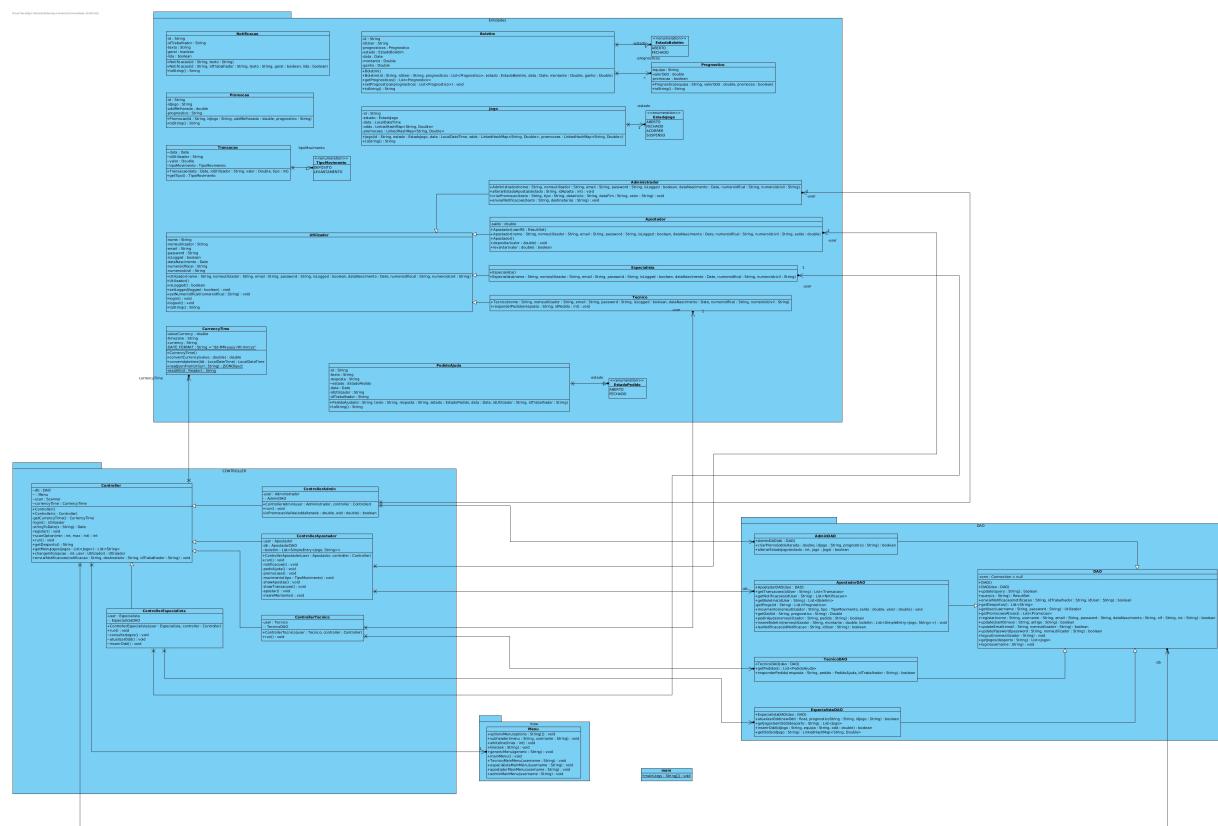
Considerando o nicho da aplicação a desenvolver, o mundo das apostas desportivas, é de esperar que a maioria das funcionalidades do próprio sistema estejam relacionadas com a visualização, alteração/modificação de dados, e sua respetiva gestão. Então, tendo esta linha bem presente entre estes processos principais, surge o modelo MVC como uma alternativa natural para a base da arquitetura do sistema. Este modelo disponibiliza, assim, uma solução simples e prática, perante a interação entre o utilizador e os dados do sistema, distinguindo também os vários tipos de utilizadores que o sistema possui e as funcionalidades de cada um.

Em relação ao armazenamento da informação, a sua persistência é assegurada por uma base de dados em *mySQL*, por se tratar de sistema de gerenciamento de Bases de Dados intuitivo e que suporta às funcionalidades específicas do sistema. Perante a existência de uma API externa fornecida (que fornece os dados dos vários desportos e jogos ao sistema), idealiza-se também a existência de mecanismos capazes de ler desta API e efetuar as transformações necessárias para a visualização da informação de um modo correto, seguro e prático. Para além disso, destaca-se a escalabilidade inerente, a qual poderá aumentar a complexidade da solução.

## 5. Modelação do Bloco de Construção

Com o objetivo de modelar o sistema e os diferentes blocos de construção que o compõe, foi realizado o diagrama de classes, que surge de seguida.

## 5.1 Diagrama de Classes - Nível 2



**Figura 5.1:** Diagrama de Classes

## 5.2 Divisão das Responsabilidades

Através das funcionalidades idealizadas para a aplicação, no decorrer da Fase 1, de especificação, é possível efetuar uma divisão das responsabilidades concreta, muito relacionada com aquilo que cada tipo de Utilizador pode/consegue fazer na aplicação. Deste modo, urge apresentar as responsabilidades atribuídas a cada componente, ou seja, os métodos associados a cada uma das componentes do sistema, efetuando a ligação com cada use case.

Observe-se o esquema abaixo, que relaciona, para cada tipo de ator, o *use case* em questão e a componente responsável:

Use Case	Grupo de Funcionalidade	Componente Responsável	
Iniciar Sessão	Gestão de conta	Utilizador Genérico	
Terminar Sessão			
Consultar Jogos			
Enviar Automaticamente Notificação	Alertas, Notificações	Notificador do Sistema	
Enviar Notificações			
Alterar Estado da Aposta	Gestão de jogos, odds, apostas	Administrador	
Registar Trabalhadores	Registros		
Responder a Pedidos	Suporte Técnico	Técnico	
Inserir Odd	Gestão de jogos, odds, apostas		
Atualizar Odd			
Consultar Notificações	Consultas	Apostador	
Consultar Promoções			
Consultar histórico de apostas			
Consultar histórico de transações			
Alterar informações de perfil	Gestão de conta		
Registar	Registros		
Fazer pedidos de ajuda	Suporte Técnico		
Fazer aposta	Gestão de jogos, odds, apostas		
Levantar Dinheiro	Gestão Bancária		
Depositar Dinheiro			

**Tabela 5.1:** Divisão de Responsabilidades.

### 5.3 Modelo Lógico

Segue-se o modelo lógico da base de dados *projeto RASBet*.

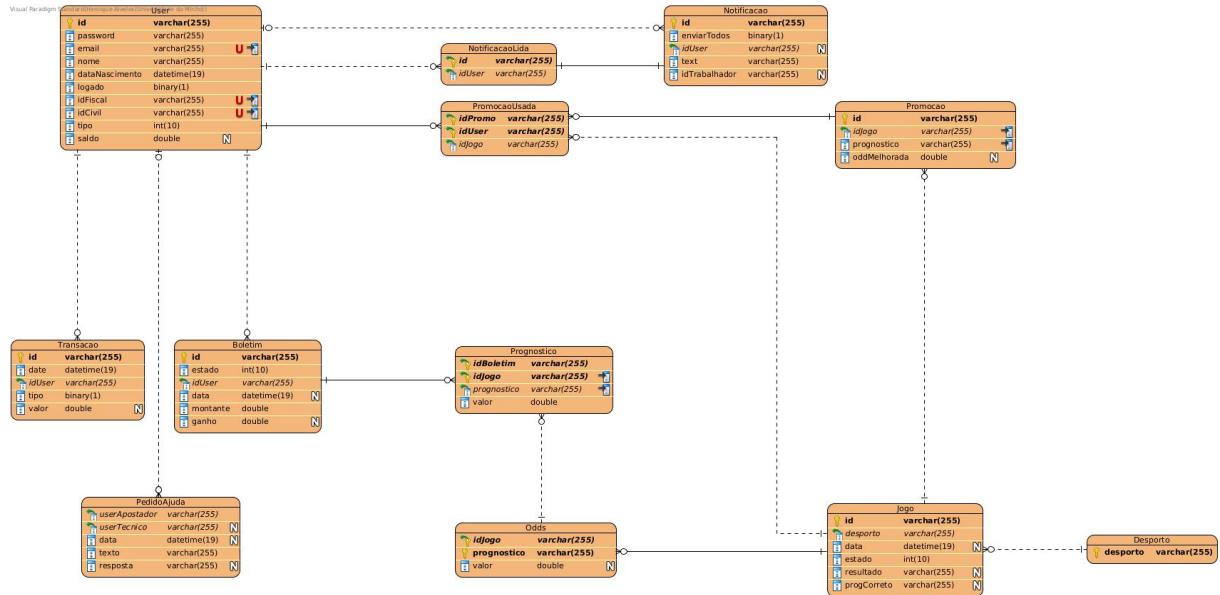


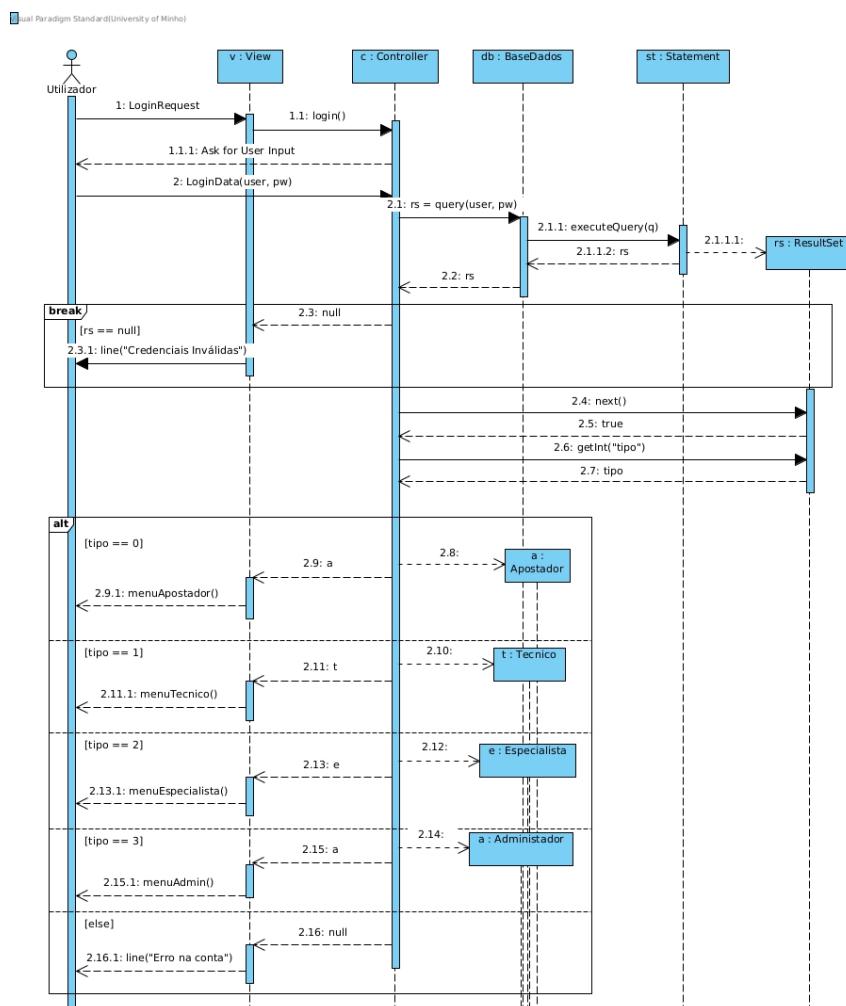
Figura 5.2: Modelo lógico

# 6. Modelação da Execução

Nesta secção, apresenta-se o conjunto dos vários diagramas de sequência construídos. Importa, agora, destacar a construção dos diagramas focada num paradigma orientado aos objetos. Estes diagramas vão permitir modelar o comportamento dos principais métodos da arquitetura, descrevendo a relação entre as várias entidades e objetos, ao longo do seu ciclo de vida.

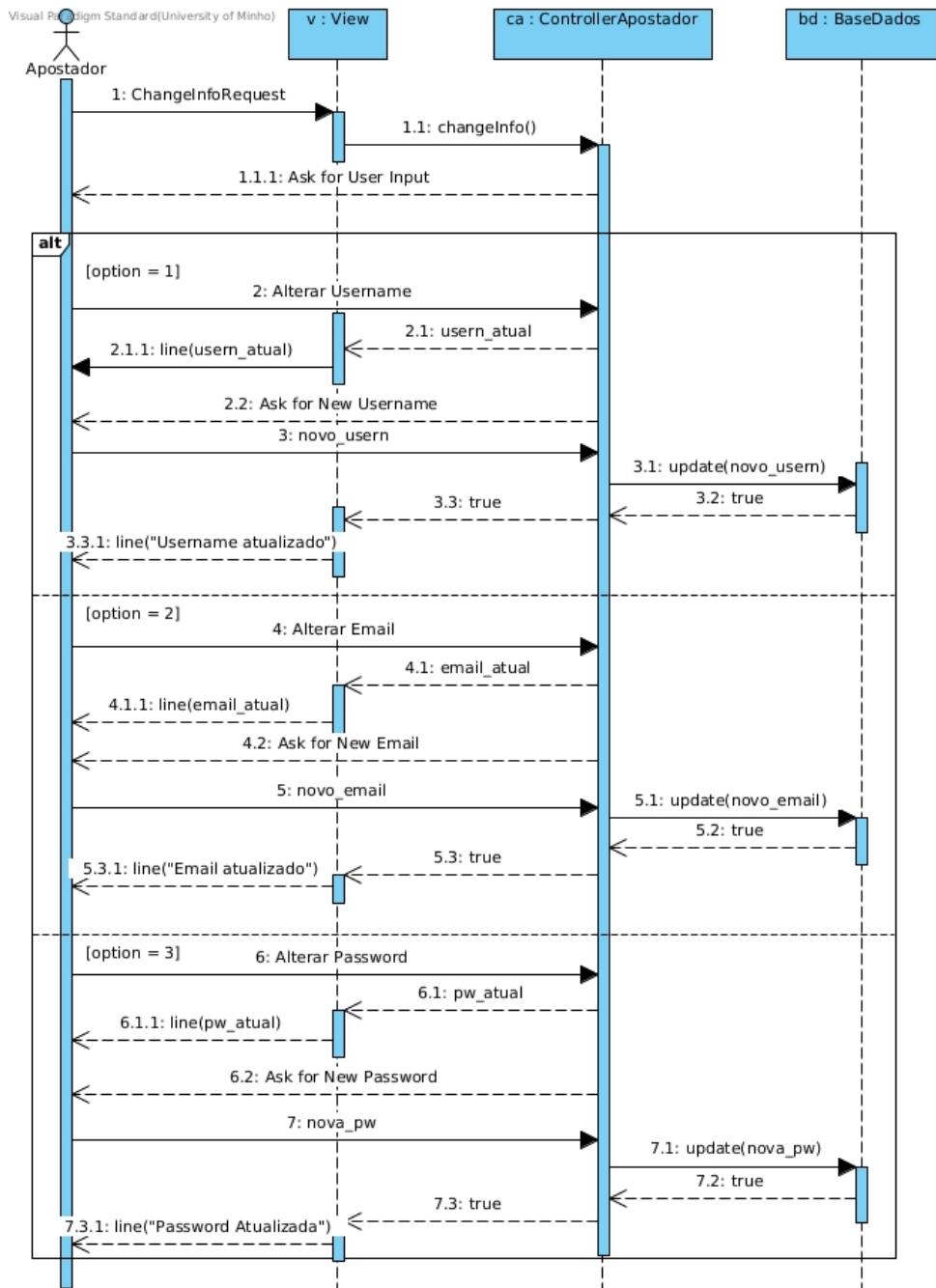
## 6.1 Gestão de Conta

### 6.1.1 Iniciar sessão



**Figura 6.1:** Diagrama de Sequência do Início de Sessão.

### 6.1.2 Alterar Informações de Perfil



**Figura 6.2:** Diagrama de Sequência da Alteração de informações de perfil.

## 6.2 Registros

### 6.2.1 Registrar

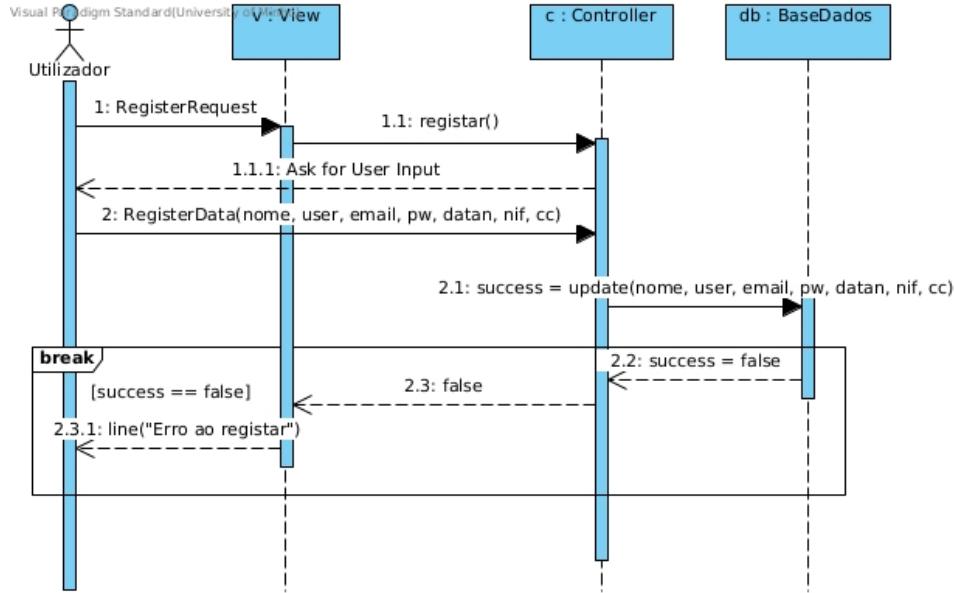


Figura 6.3: Diagrama de Sequência do Registo.

## 6.3 Gestão Bancária

### 6.3.1 Depositar Dinheiro

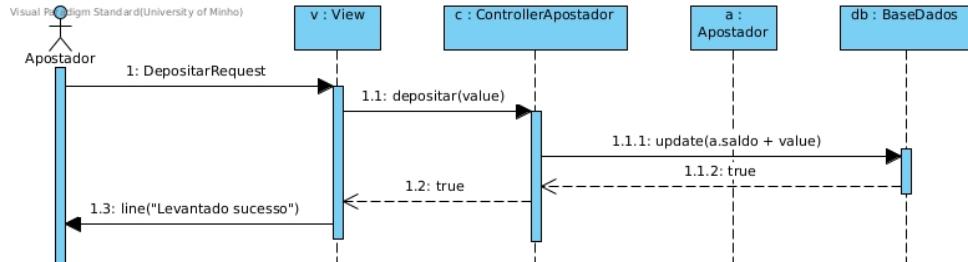


Figura 6.4: Diagrama de Sequência do Depósito de Dinheiro.

## 6.4 Gestão de Jogos, Odds, Apostas

### 6.4.1 Fazer Aposta

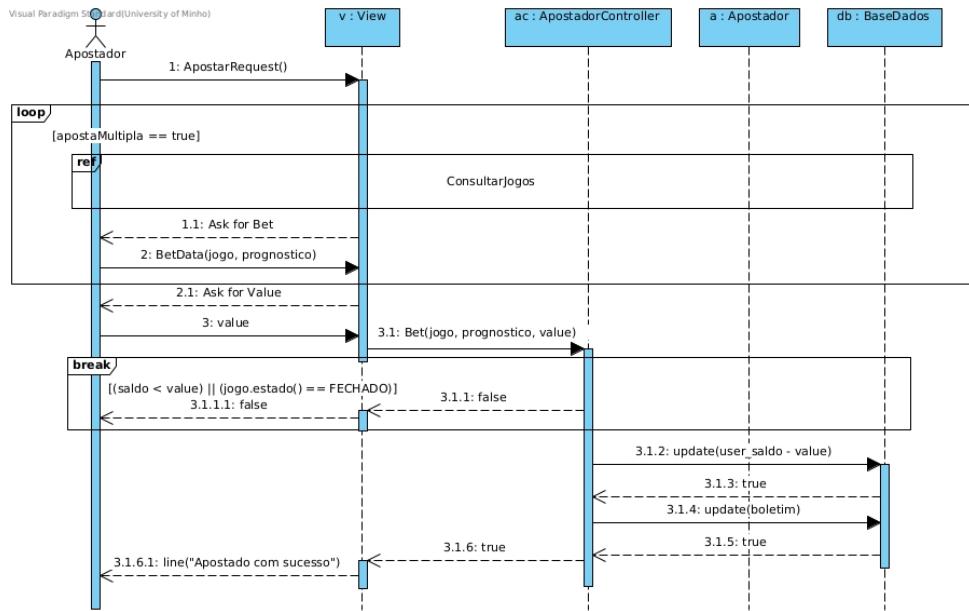


Figura 6.5: Diagrama de Sequência de Fazer uma Aposta.

### 6.4.2 Criar Promoções

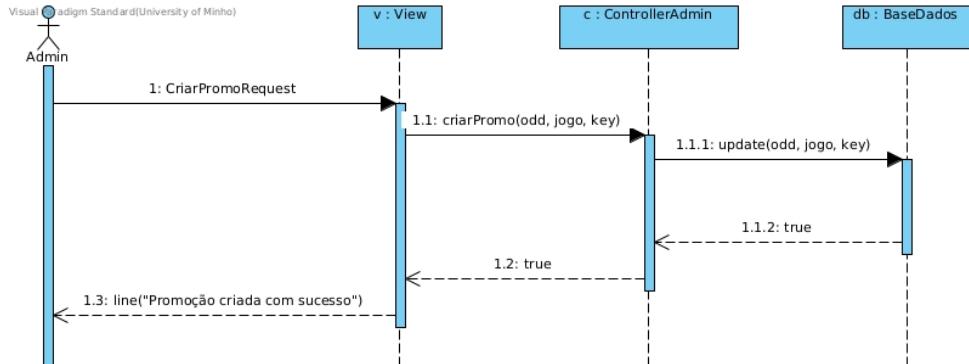
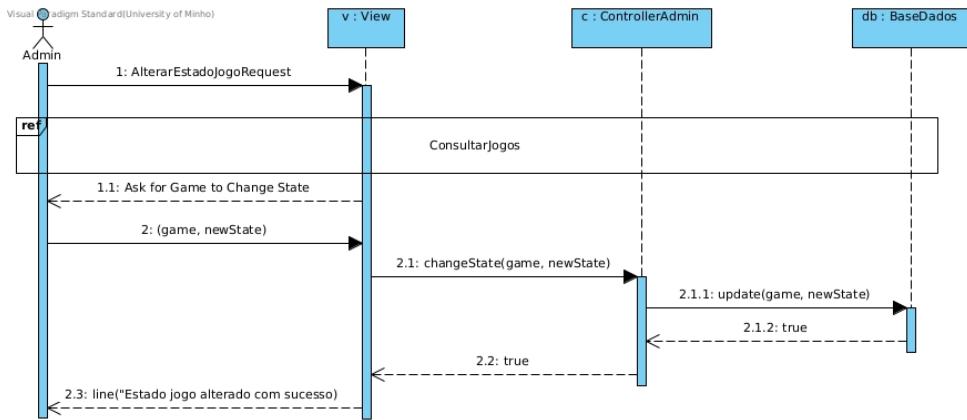


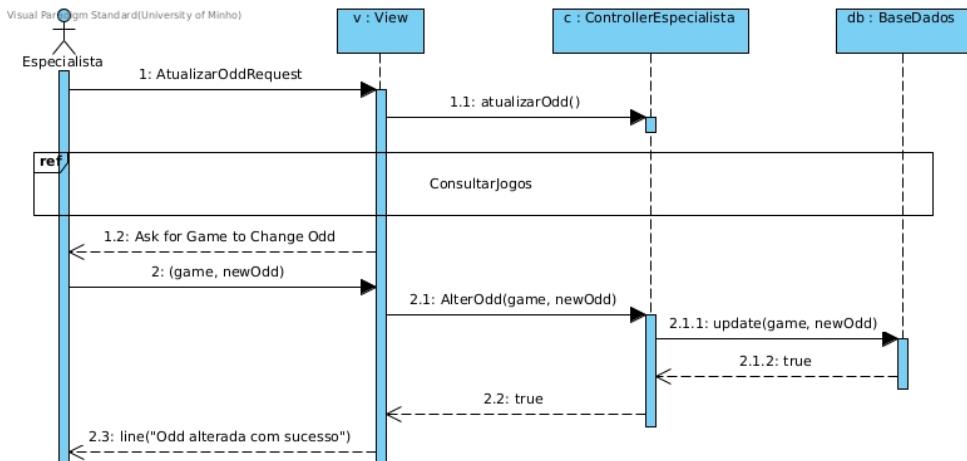
Figura 6.6: Diagrama de Sequência da Criação de Promoções.

### 6.4.3 Alterar Estado da Aposta



**Figura 6.7:** Diagrama de Sequência da Alteração do Estado do Boletim de Jogo.

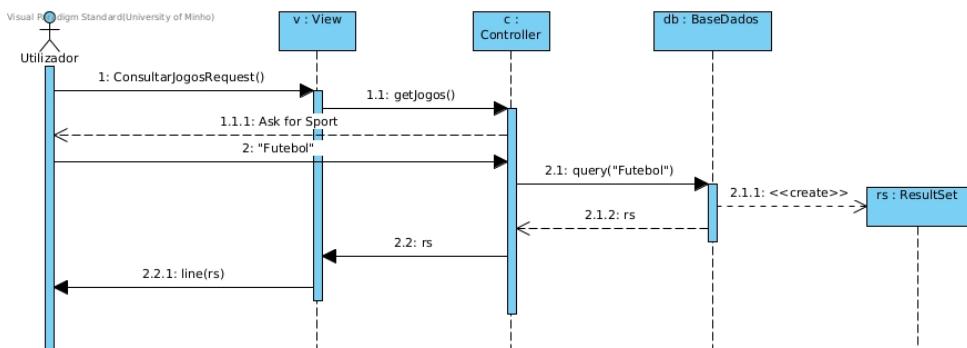
### 6.4.4 Alterar Odd



**Figura 6.8:** Diagrama de Sequência da Atualização da Odd.

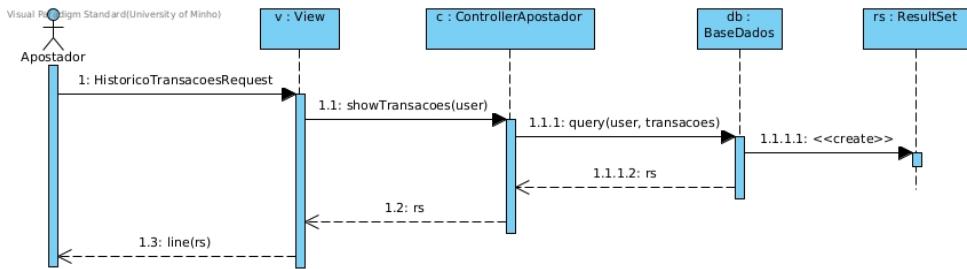
## 6.5 Consultas

### 6.5.1 Consultar Jogos



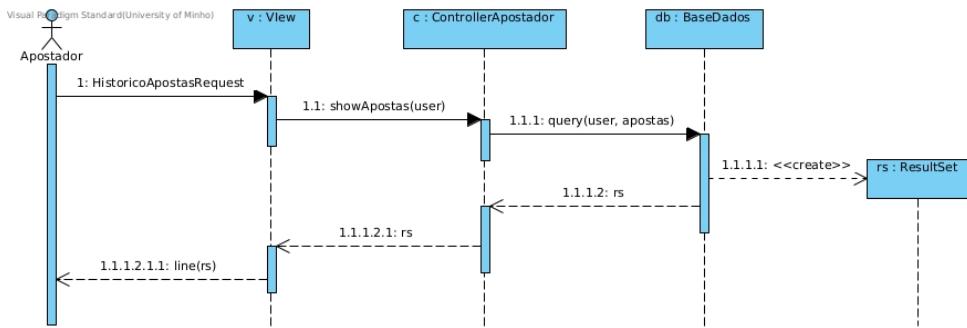
**Figura 6.9:** Diagrama de Sequência da Consulta de Jogos.

### 6.5.2 Consultar Histórico de Transações



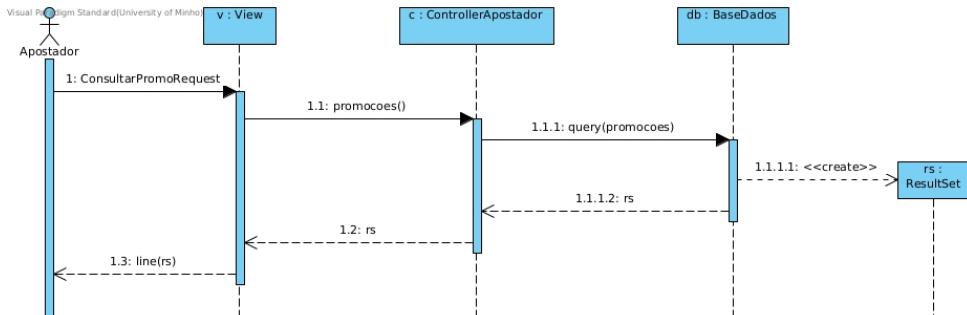
**Figura 6.10:** Diagrama de Sequência da Consulta do Histórico de Transações.

### 6.5.3 Consultar Histórico de Apostas



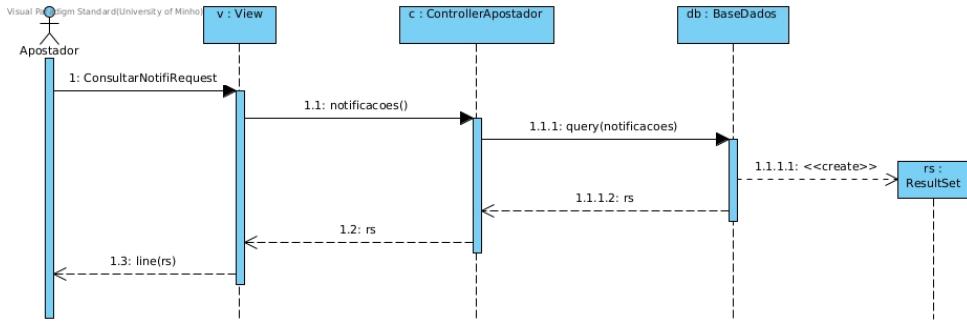
**Figura 6.11:** Diagrama de Sequência da Consulta do Histórico de Apostas.

### 6.5.4 Consultar Promoções



**Figura 6.12:** Diagrama de Sequência da Consulta de Promoções.

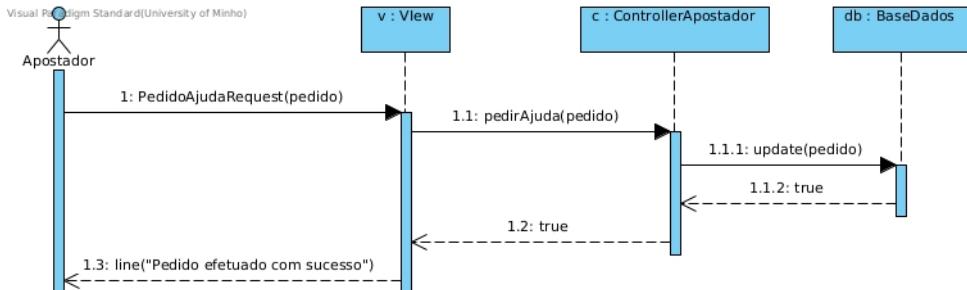
### 6.5.5 Consultar Notificações



**Figura 6.13:** Diagrama de Sequência da Consulta de Notificações.

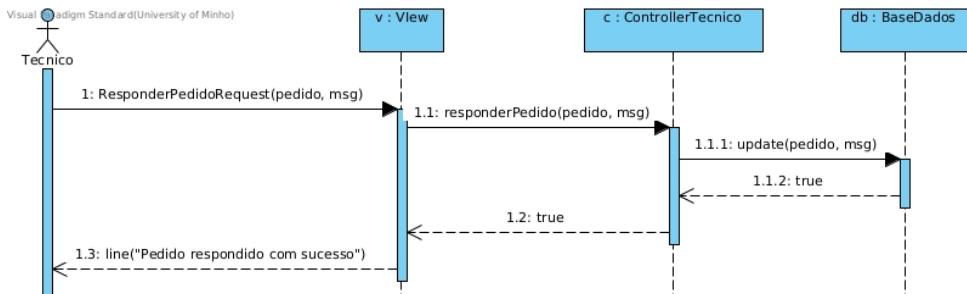
## 6.6 Suporte Técnico

### 6.6.1 Fazer Pedidos de Ajuda



**Figura 6.14:** Diagrama de Sequência do Pedido de Ajuda.

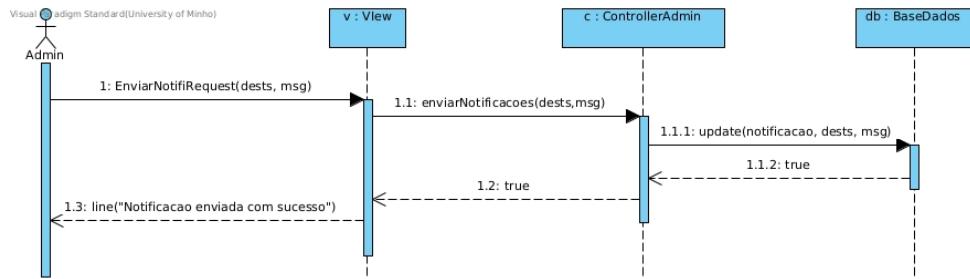
### 6.6.2 Responder a Pedidos



**Figura 6.15:** Diagrama de Sequência da Resposta ao Pedido de Ajuda.

## 6.7 Alertas e Notificações

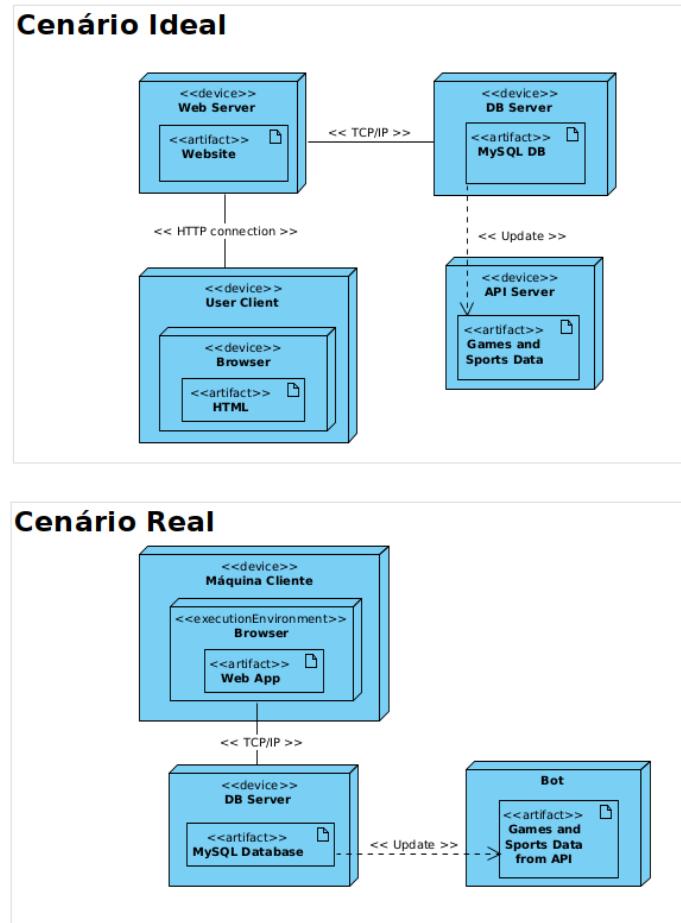
### 6.7.1 Enviar Notificações



**Figura 6.16:** Diagrama de Sequência do Envio de Notificações pelo Administrador.

## 7. Modelação de *Deployment*

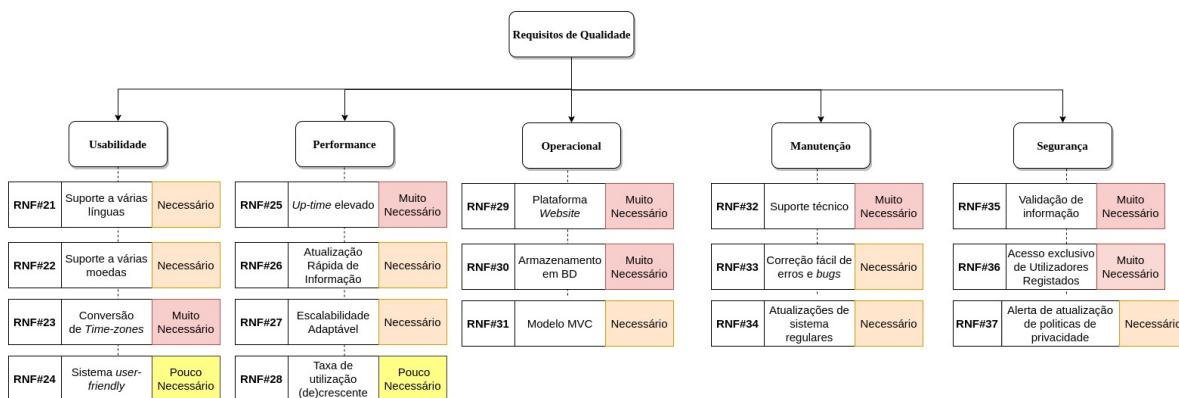
O diagrama de *deployment* vai permitir visualizar qual o hardware onde o programa implementado será executado. Numa primeira instância do sistema, todos os componentes do programa (sistema, base de dados e API externa) serão executados na mesma máquina. Desde logo, a utilização de uma API externa leva à existência de uma dependência dos dados da API para o correto funcionamento da aplicação, por isso, decidimos inseri-la também neste diagrama. A informação da BD existirá na nossa máquina. Idealmente, existiria um servidor que permitia a atualização da informação em qualquer dispositivo, garantindo a sua persistência, o que levaria à adição de um novo componente de *hardware*, um servidor, para a BD. Além disso, tendo em conta a execução da aplicação no formato *web*, também poderíamos representar um servidor *web*, onde se executaria o *website*. Neste caso, seria o servidor responsável pelo arquivo e gestão de informação do *site*, tendo-se assim, um outro componente de *hardware*.



**Figura 7.1:** Diagrama de Implantação para cada cenário.

## 8. Requisitos de Qualidade

O conjunto de requisitos de qualidade já foi, implicitamente, especificado na Fase 1, no capítulo que diz respeito aos requisitos não funcionais. Apesar disso, serve esta secção para descrever de forma genérica os requisitos de maior importância, focando a análise através de cenários específicos para cada um. É muito comum, neste âmbito, representar o conjunto de requisitos por tipo, acompanhando-os das respectivas tolerâncias (desde nada necessário até muito necessário). Contudo, estas tolerâncias acabam por se traduzir através das várias prioridades. Assim sendo, a árvore de qualidade seguinte aglomera este conjunto de requisitos, sendo o requisito não funcional que lhe deu origem especificado. Do conjunto de requisitos não funcionais, identificamos os principais cinco tipos, e para cada um, colocamos as suas tolerâncias, surgindo os seguintes requisitos de qualidade:



**Figura 8.1:** Árvore de Qualidade.

Tendo em conta o conjunto de requisitos anterior, no contexto de desenvolvimento da aplicação, repara-se que os requisitos de aparência/aspeto, legais e culturais e políticos não constam. Decidiu-se não incluir os de aparência, já que, não são fundamentais ou indispensáveis para a existência da aplicação, ou seja, olhando para as suas prioridades, decidiu-se que não eram os mais relevantes. Não incluímos os requisitos legais e culturais e políticos, porque os primeiros estão muito dependentes do contexto legal de cada local, e os segundos não só dependem muito não do local, mas também têm uma subjetividade maior. Os requisitos legais são de extrema importância, contudo não são controláveis no que diz respeito à solução implementada.

Seguem-se, agora, os cenários para cada requisito identificado anteriormente.

## Cenários

Estes cenários têm o objetivo de descrever o comportamento do serviço.

- **Suporte a várias línguas:** O sistema suporta várias línguas, de modo a ampliar o número de utilizadores que podem utilizar a aplicação, permitindo suportar utilizadores de diferentes regiões;
- **Suporte a várias moedas:** O sistema suporta vários tipos de moedas, de modo a ampliar o número de utilizadores que podem utilizar a aplicação, permitindo suportar utilizadores de diferentes regiões;
- **Conversão de *Time-zones*:** O sistema é capaz de converter o horário de um determinado jogo, de acordo com a time-zone do apostador, de modo a garantir a sincronização entre os tempos dos vários apostadores dos diferentes locais;
- **Sistema *user-friendly*:** O sistema é intuitivo e prático, especialmente para apostadores que não estejam muito familiarizados com sites de apostas, garantindo uma utilização fácil e cómoda por parte de cada apostador;
- **Up-time elevado:** Perante uma queda do sistema, o administrador deve ser informado e o estado do sistema deve ser guardado, garantindo que os dados dos apostadores não são postos em causa. Então, o sistema deve estar ativo 99% do tempo.
- **Atualização Rápida de informação:** As atualizações de dados relativos ao serviço por parte do especialista estão disponíveis ao utilizador em tempo-real, em menos de 1 segundo.
- **Escalabilidade Adaptável:** O sistema adapta-se ao número (de)crescente de utilizadores, fazendo a gestão de recursos necessária, pelo que: não deverá existir um aumento superior a 40% nos tempos de resposta, quando existem até 400 utilizadores ativos no sistema;
- **Plataforma *Web-site*:** O sistema deve ser apresentado numa plataforma web, nos 3 browsers especificados mais utilizados (Google Chrome, Microsoft Edge e Safari), garantindo que está disponível em vários em dispositivos variados e levando a um maior número de apostadores que utilizam a aplicação.
- **Armazenamento em BD:** O sistema guarda os dados dos utilizadores numa Base de Dados, que armazena toda a sua informação, contribuindo para a persistência dos dados da aplicação.
- **Modelo MVC:** O sistema respeita o modelo arquitetural MVC (Modelo Vista-Controlador), permitindo uma fácil integração das camadas visuais e de interação do utilizador com a lógica de negócio.
- **Suporte técnico:** O sistema é capaz de prestar suporte técnico ao utilizador, em caso de necessidade, fornecendo um meio de contacto com um técnico adequado e disponibilização de uma aba de FAQs.
- **Correção fácil de erros e bugs:** O sistema permite uma correção eficaz e prática de bugs, alertando para a existência deste e para a necessidade de um *check-up*, caso o mesmo não tenha sido feito nas 4h semanais dedicadas para o mesmo;

- **Atualizações de sistema regulares:** O sistema permite atualizações regulares em alturas de pouca utilização. Através de uma análise cuidada dos relatórios estatísticos do sistema, observam-se quais são as 10 horas de menor movimento/utilização mensal, efetuando-se as atualizações necessárias, durante esse intervalo de tempo;
- **Validação de informação:** O sistema valida a informação submetida, impedindo-a de ser armazenada, caso esteja incorreta (e alerta o utilizador). Isso permite garantir a integridade dos dados.
- **Acesso exclusivo de Utilizadores Registados:** O sistema deve garantir a gestão responsável da informação, garantindo o acesso exclusivo de informação( apenas os especialistas registados geram as *odds* de um resultado e só administradores devem conseguir registar trabalhadores, por exemplo);
- **Alerta de atualização de políticas de privacidade:** O sistema informa os utilizadores sobre a alteração das políticas de privacidade, garantindo que o utilizador está em conformidade com as novas políticas.

## 9. Riscos e Dívida Teórica

Neste capítulo, é importante apresentar o conjunto de sistemas que se envolvem com a aplicação a desenvolver, especificando a necessidade de existência de novos sistemas vizinhos que tornem a nossa aplicação mais robusta e melhor implementada no futuro. Logo no início do documento, no diagrama de contexto, foi mostrado um cenário idealizado da relação do sistema RASBet com outros sistemas que seriam importantes para garantir o melhor funcionamento do sistema, seja em termos de funcionalidades, seja em termos de gestão e manutenção da aplicação a longo prazo. Sendo assim, vamos focar em análise no contexto real. Então, considerem-se as seguintes observações:

- **Sistema (bancário) de métodos de pagamento:** No capítulo relativo a *Factos e Presupostos* (Fase 1) já se fazia referência ao facto de, por se tratar de um contexto académico, se ter assumido que não se implementaria um sistema de pagamento externo, por questão de simplicidade. Porém, num contexto real, fazia todo o sentido implementar este sistema de pagamento, que fosse responsável por gerir o dinheiro envolvido na aplicação, mais propriamente, aquando dos levantamentos e depósitos. Deveria assim, ser possível depositar dinheiro na carteira, através da comunicação com a conta bancária do cliente, o banco autorizaria ou não o débito do montante especificado da conta do apostador. Da mesma forma, aquando do levantamento, o banco validaria ou não a transferência para a conta do apostador. Estes depósitos/levantamentos seriam também feitos de acordo com o método mais cômodo para o apostador, seja por multibanco, *mbway*, *paypal*, etc. Seria interessante desenvolver adequadamente estas funcionalidades. Associado a esta técnica, temos o risco de que a disponibilidade do sistema fica associada à disponibilidade das interfaces.
- **Bot de Suporte:** Seria igualmente interessante a existência de um *Bot*, no que diz respeito ao suporte técnico. Então, um apostador, em caso de dúvidas, falaria com este Bot, que seria responsável por entender a dúvida do apostador e responder da forma mais adequada, como acontece na maioria dos casos do contexto real. Funcionaria com um apoio ao cliente: atualmente, já existem robôs que efetuam até chamadas telefónicas, obtendo e produzindo informação do/para o cliente.
- **Equipa de gestão, legalidade e manutenção:** Já num contexto mais de gestão e manutenção da aplicação, também mencionado no diagrama de contexto inicial, seria relevante existir todo um conjunto de equipas, quer responsáveis por questões legais (garantir que todos os clientes cumpririam os requisitos legais através da comparação da data de nascimento colocada no registo e a data de nascimento que surge no CC - o utilizador deveria dar *upload* da foto do CC aquando do registo), quer por questões de gestão estatística e *marketing* (elaboração de relatórios, interessantes para o melhoramento da aplicação de acordo com o *feedback* dos utilizadores). Esta ideia está relacionada com um risco da dependência das interfaces externas com as quais o sistema comunica.
- **Tiers:** No sentido de tornar o sistema mais exclusivo, poderia-se adotar um sistema de *Tiers*, onde, pagando um certo valor mensal ou anual, se disponibilizaria um serviço mais refinado e com mais possibilidades de apostas, por exemplo, através da apresentação de

informações mais pormenorizadas sobre os eventos desportivos, seja em termos de dados atualizados ao segundo, como ocorre em contexto real. A existência de dinheiro em jogo, torna o sistema inevitavelmente mais potencial ataques, existindo também aqui um risco de segurança associado.

## 10. Glossário

<b>RASBet</b>	Nome do sistema de apostas a implementar
<b>RAS</b>	Requisitos e Arquiteturas de <i>Software</i>
<b>odd</b>	Métrica da probabilidade de um resultado
<b>FAQs</b>	<i>Frequently Asked Questions</i>
<b>BD</b>	Base de Dados
<b>DB</b>	<i>DataBase</i>
<b>CC</b>	Cartão de Cidadão
<b>UI</b>	<i>User Interface</i>
<b>HTML</b>	<i>HyperText Markup Language</i>
<b>API</b>	<i>Application Programming Interface</i>

# Conclusão

Na sua globalidade, estamos contentes com o trabalho realizado uma vez que todos os requisitos funcionais foram implementados contudo apesar de tentarmos ligar o *front-end* ao *back-end* tal não foi possível e ficou abaixo dos nossos objetivos. contudo

Nesta segunda fase do projeto, seguimos de igual forma as técnicas de análise de requisitos e os padrões de arquitetura e desenvolvimento neste que foi um dos nossos primeiros contactos com a Engenharia de *Software*, no que diz respeito a uma análise muito mais profunda dos requisitos de *Software*. Consideramos este tema primordial à nossa formação enquanto Engenheiros Informáticos.

Esta fase de construção permitiu-nos "meter as mãos na massa" e dados os seus requisitos operacionais e funcionais construir um sólido sistema de *Software*, operando no domínio de uma base de dados relacional.

Embora pensemos ter cumprido todos os objetivos propostos, em todos os projetos de grande dimensão e que envolvem bastante trabalho e rigor, há sempre pontos em que apresentamos mais dificuldades, e este projeto não foi exceção.

Neste contexto, os nossos principais pontos fracos e dificuldades recaíram sobre o domínio das nossas linguagens de programação e como referenciado a conexão do *front-end* com o *back-end*. Apesar disso, apresentamos maior facilidade a trabalhar como grupo, discutindo profusas vezes as ideias apresentadas mas mantendo o espírito de equipa como base do progresso do trabalho.

Esperamos deste modo, na fase posterior, estender o trabalho, de modo a torná-lo mais completo e funcional.