Universidade do Minho

Departamento de Informática

RASBet

Requisitos e Arquiteturas de Software
(1.º Semestre/2021-2022)

PL2 / Entrega 2

PG47138 Dinis Lourenço Gomes

PG47347 Joel Salgueiro Martins

PG47087 Carlos Filipe Coelho Ferreira

PG47627 Rúben Daniel Almeida Adão


Dinis Gomes


Joel Martins


Carlos Ferreira


Rúben Adão

19 de novembro de 2021
2021/2022

Tabela de Conteúdos

Lista de Figuras

Prefácio

Ao longo desta segunda fase do trabalho prático será demonstrada uma reavaliação da parte da empresa, nomeadamente, uma ideologia de como o software será desenvolvido. Para tal serão mencionados e expostos vários diagramas que auxiliam a formação deste documento.

Para o documento gerado foi utilizado o template *arc42* um template para arquiteturas de comunicação e documentação, sem exceção de nenhum tópico.

A equipa sente-se de forma geral satisfeita com o trabalho realizado até este ponto, e considera que esta segunda fase está bem formulada para um suave desenvolvimento futuro das próximas fases.

Contribuição dos Elementos

- PG47138 Dinis Lourenço Gomes: 0
- PG47347 Joel Salgueiro Martins: 0
- PG47087 Carlos Filipe Coelho Ferreira: 0
- PG47627 Rúben Daniel Almeida Adão: 0

1. Introdução e objetivos

1.1 - Requirements Overview

Podemos referir alguns requisitos cruciais ao funcionamento sistema:

Requisito	**Descrição**
Requisito 5	Os utilizadores podem efetivamente fazer apostas para um jogo, sendo o saldo retirado e a aposta registada.
Requisito 3	Os utilizadores devem receber notificações sobre jogos aos quais apostaram se estes acabaram, assim como o seu resultado.
Requisito 16	Um gerente ou API, deve conseguir alterar os rates das apostas, fechar as apostas ou alterar o estado de um jogo, como por exemplo, dar um jogo como terminado e apresentar o resultado.
Requisito 6	Os utilizadores podem depositar ou receber dinheiro na sua conta do web site para a sua conta multibanco.

Tabela 1: Requisitos Cruciais

Os requisitos restantes e os previamente apresentados apresentam-se disponibilizados no capítulo 9 do relatório RASBet-1.

1.2 - Quality Goals

Segurança

A prioridade mais alta do software será garantir um ambiente no qual os dados e o dinheiro dos utilizadores e da própria empresa estejam seguros.

Pode-se apresentar a situação em que os utilizadores não ganhem mais dinheiro do que o previamente calculado ou que não percam menos dinheiro do que aquilo que decidiram apostar, ou seja, garantir que não haja alterações ilegítimas à carteira dos utilizadores.

Outra secção pertinente, os dados pessoais dos utilizadores, devem ser protegidos de modo a evitar *leaks*.

Capacidade de Manutenção

O sistema deverá ter a habilidade de ser modificado e/ou adaptado, de acordo com as necessidades actuais da empresa, sejam estas novos desportos que esta queira incluir ou modificações na parte visual da aplicação.

Operabilidade

O sistema terá de ser de fácil utilização de modo a permitir que utilizadores que não tenham grande conhecimento técnico da tecnologia associada à aplicação, ou seja, um telemóvel, computador, etc….

Compatibilidade

A compatibilidade entre dispositivos é também fulcral. O utilizador pode querer visualizar ou modificar, as apostas feitas quando se encontra longe do seu dispositivo habitual.

1.3 - Stakeholders

Os *stakeholders* associados ao desenvolvimento deste sistema mantiveram-se na maioria.

Nome	***Roles/Job***	***Degree of involvement***	***Degree of influence***	***Knowledge needed***
Dinis Lourenço Gomes	Tester, SD, User	Máximo	Médio	Máximo
Joel Salgueiro Martins	Tester, SD, User	Máximo	Médio	Máximo
Carlos Filipe Coelho Ferreira	Tester, SD, User, GP	Máximo	Médio	Máximo
Rúben Daniel Almeida Adão	Tester, SD, User	Máximo	Médio	Máximo
ERA	GCLR	Máximo	Máximo	Médio
IAJ	———	Mínimo	Mínimo	Médio
Administradores (Empresa)	Admin	Máximo	Máximo	Máximo
Departamento Legal	Advogado	Máximo	Máximo	Máximo
Patrocinadores	Sponsor	Médio	Máximo	Mínimo
Utilizadores	Tester, User	———	Mínimo	Mínimo

Tabela 2: *Stakeholders* de RASBet

2. Constraints

Nesta secção do relatório seriam referidos algumas restrições que influenciaram o desenvolvimento da aplicação. No entanto estes já foram abordados no capítulo 4 - Restrições Obrigatórias do relatório RASBet-1.

3. Context and Scope

Neste tópico iriam ser referidas as delimitações do sistema e as suas respectivas comunicações com terceiros, no entanto este tópico ja foi abordado no capítulo 7 do relatório RASBet-1.

Business Context

Este tópico já foi coberto no capítulo 8.1 do relatório RASBet-1.

Technical Context

Este tópico já foi coberto no capítulo 8.2 do relatório RASBet-1.

4. Solution Strategy

Conteúdo

De seguida iremos apresentar um conjunto de decisões tomadas relativamente ao desenvolvimento do software em questão. Começando por:

- **Decisões de tecnologia**: A equipa decidiu que todo o backend do software irá ser desenvolvido na linguagem Java. Já o frontend seria principalmente com uso das linguagens HTML, CSS e JavaScript.
- **Decisões sobre a decomposição de nível superior do sistema**: Ficou decidido que a componente do software que interage com o cliente, portanto o frontend, irá ser decomposta em várias páginas que apenas diferem no seu conteúdo central, mantendo o conteúdo do topo da página e conteúdo do fim da página inalterados.
- **Decisões sobre como atingir os principais objetivos de qualidade**: Para atingir os respetivos objetivos de qualidade a equipa decidiu marcar diversas reúniões para avaliar o estado do desenvolvimento do software.
- **Decisões organizacionais relevantes**: A equipa optou por dividir o desenvolvimento em partes para que podessem ser adotados métodos de delegação de tarefas a terceiros.

Motivação

Uma vez que estas decisões são os pilares da arquitetura da aplicação, devem ser tomadas com especial cuidado, assim como prevendo os efeitos colaterais que irão tem no desenvolvimento posterior.

Forma

A equipa decidiu desenvolver o backend da aplicação usando a linguagem Java devido ao seu enorme poder e também devido ao facto de todos os membros da equipa se sentirem à vontade com ela. Para o frontend, apesar da equipa não ter conhecimentos sobre as linguagens, ficou decidido usar as linguagens HTML, CSS e JavaScript, uma vez que estas são muitíssimo utilizadas no mercado de trabalho.

Objetivo	**Abordagem Arquitetônica**
Base de dados	Todo o código da base de dados foi gerado a partir do modelo lógico construído para a base de dados
Interface das páginas	Toda a interface das páginas foi cuidadosamente desenhada, antes de se idealizar.
Manipulação dos dados sensíveis	Todas as funções que manipulam dados sensíveis, como credências e dinheiro, foram devidamente pensadas e planeadas com diversos diagramas

Tabela 3: Abordagens estratégicas gerais

5. Building Block View

Whitebox Overall System

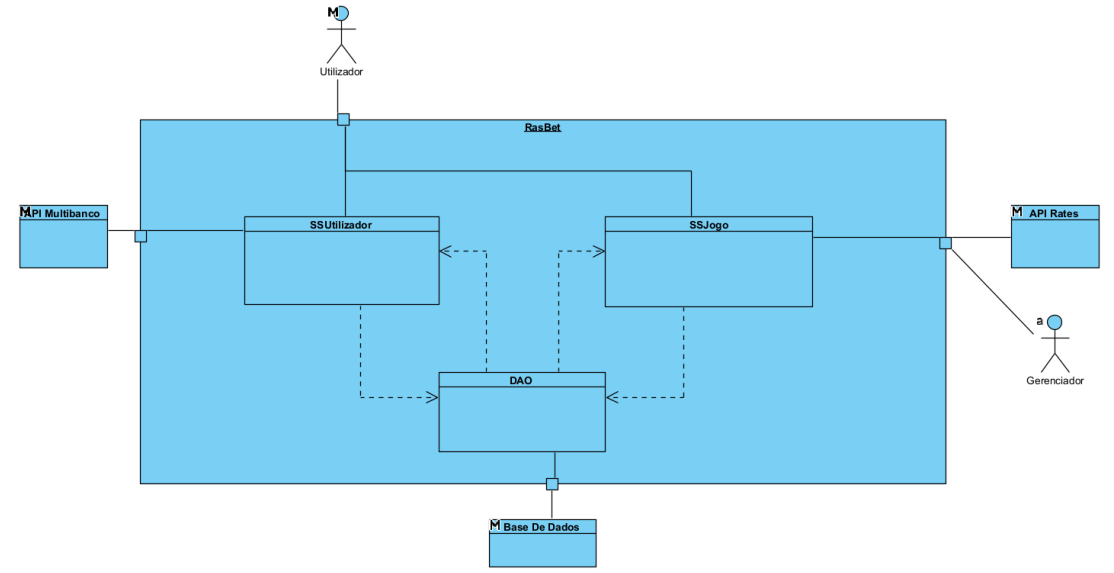

Figura 1: Building Block View - Nivel 1

Raciocínio: Com o objetivo de desenvolver um sistema organizado, claro e eficiente, a equipa decidiu dividir este em diferentes subsistemas dependendo das funcionalidades gerais que devem ser implementadas.

- **Sub-Sistemas:** Deve encapsular as respetivas funcionalidades do seu domínio.
- **Data Acess Object:**Deve permitir acesso a todos os dados do sistema

Building Block	Descrição
SSUtilizador	Responsável por encapsular as funcionalidades relacionadas aos utilizadores do sistema, como a sua informação pessoal, bancária e notificações.
SSJogos	Responsável por encapsular as funcionalidades relacionadas a realização de apostas, como jogos, equipas e registo de cada aposta realizada.
DAO	Responsável por manter o acesso entre o nosso sistema e a sua base de dados, permitindo a leitura e escrita da última.

Tabela 4: WhiteBox RasBet

Data Acess Object:
A classe **DAO** responsável por aceder a base de dados é um classe com uma extrema importância no desenvolvimento do sistema, isto porque, é nesta classe que se prevê um maior impedimento para atingir os requisitos de qualidade e segurança desejados, visto que aceder a base de dados para além de ser um processo demoroso é também necessário ter em atenção o cenário ao qual várias entidades o tentam fazer em simultâneo.

Building Blocks - Nivel 2

SSUtilizador (WhiteBox)

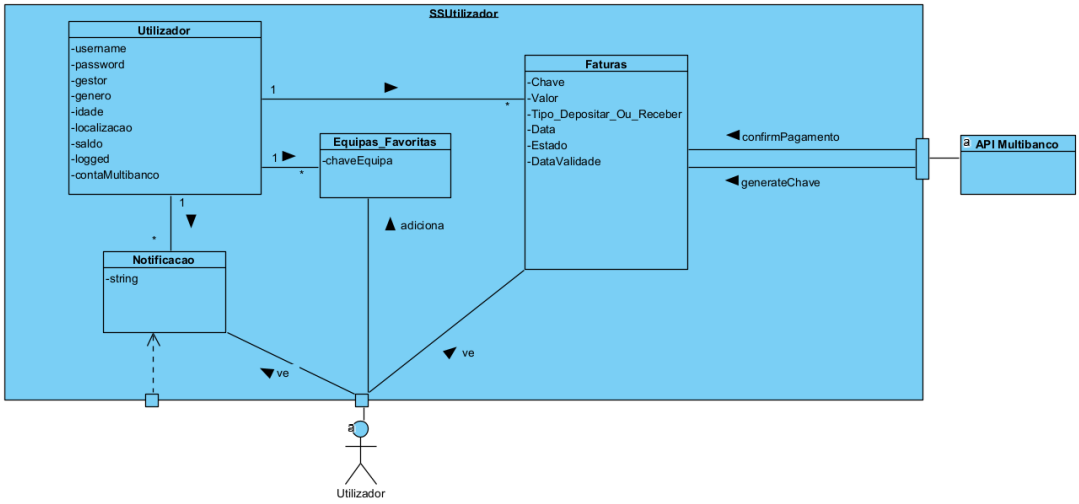

Figura 2: Building Block View - Nivel 2 - SSUtilizador

Raciocínio:O sub-sistema do âmbito utilizadores deve então conter todas as classes necessárias para guardar e gerir a informação sobre utilizadores e os seus atributos, seguindo a seguinte composição:

- **Utilizador**
- **Notificações de cada utilizador**
- **Equipas favoritas de cada utilizador**
- **Faturas registadas de cada utilizador**

Building Block	Descrição
Utilizador	Responsável por guardar os atributos pessoais de cada utilizador, assim como as suas credenciais para fazer login no sistema e até indicar se está online ou não.
Notificacoes	Várias mensagens que cada utilizador têm sobre o estado das suas apostas, novos jogos disponíveis das suas equipas preferidas, entre outros possíveis.
Faturas	Registo e estado de todas as transferências bancarias entre o nosso utilizador e a aplicação.
Equipas_Favoritas	Registo de todas as equipas cujo utilizador indicou como favoritas.

Tabela 5: WhiteBox SSUtilizador

Faturas:
A classe **Faturas** possui uma relação com a API externa Multibanco que vale a pena salientar. A API acede a faturas para realizar duas operações diferentes.

- Originar a chave da fatura na sua criação que será depois utilizada como referência para realizar os pagamentos multibanco.
- Alterar o estado da fatura quando o pagamento tiver sido confirmado.Somente quando tal acontece é que o sistema deve atualizar o saldo do utilizador.

SSJogo (WhiteBox)

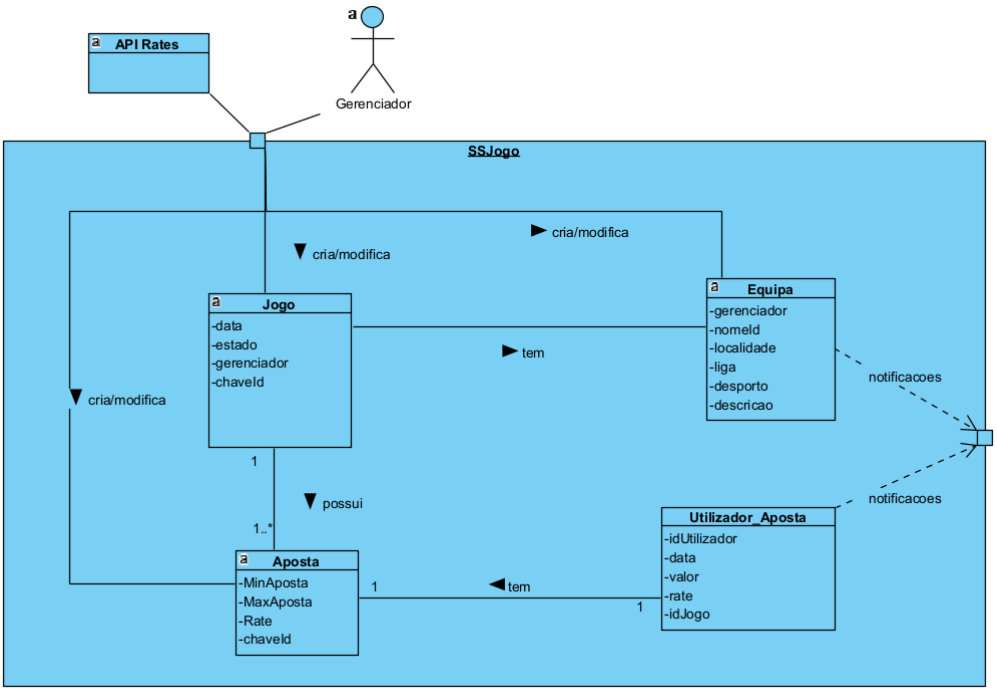

Figura 3: Building Block View - Nivel 2 - SSJogo

Raciocínio:O sub-sistema do âmbito jogo deve então conter todas as classes necessárias para guardar e gerir a informação sobre jogos, equipas, apostas e os seus atributos, seguindo a seguinte composição:

- **Jogo**
- **Aposta**
- **Equipa**
- **Apostas realizadas por cada utilizador**

Building Block	**Descrição**
Jogo	Responsável por guardar as informações gerais de cada jogo.
Aposta	Responsável por guardar as informações gerais de cada aposta, relacionando esta com o respetivo jogo.
Equipa	Responsável por guardar as informações gerais de cada equipa.
Utilizador_Aposta	Registo de todas as apostas efetuadas por cada utilizador, assim como o seu estado e o rate na altura da realização da aposta.

Tabela 6: WhiteBox SSJogo

Ao efetuar algumas operações neste sub-sistema é necessário gerar algumas notificações que serão direcionadas aos nossos utilizadores assim como:

- Na criação de um jogo, os utilizadores cujas equipas favoritas participarão deverão ser notificados.
- Ao alterar o estado de um jogo ou as suas apostas os utilizadores que no passado realizaram apostas sobre este jogo devem ser notificados.

DAO (WhiteBox)

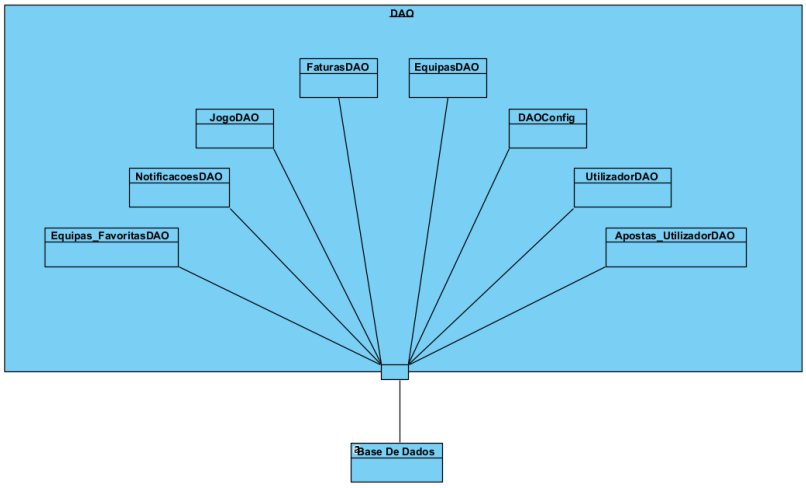

Figura 4: Building Block View - Nivel 2 - DAO

Raciocínio:A classe responsável por acessos a base de dados foi por motivos de organização e gestão multi-thread dividida em várias classes dependendo do tipo de informação ao qual se deseja aceder.

- **Classe de acesso a faturas**
- **Classe de acesso a jogos**
- **Classe de acesso a notificações**
- **Classe de acesso a equipas**
- **Classe de acesso a equipas favoritas**
- **Classe de acesso a utilizador**
- **Classe de acesso a apostas realizadas**
- **Classe de configuração da base de dados**

Building Block	**Descrição**
JogoDao	Classe responsável por criar, ler e modificar jogos e as suas respetivas apostas na base de dados.
UtilizadorDao	Classe responsável por criar e ler a informação relativa a cada utilizador na base de dados.
EquipasDao	Classe responsável por criar e ler a informação relativa a cada equipa na base de dados.
FaturasDao	Classe responsável por criar, modificar e ler a informação relativa a cada fatura na base de dados.
NotificacoesDao	Classe responsável por criar, ler e remover a informação relativa a cada notificação na base de dados.
Apostas_UtilizadorDao	Classe responsável por criar e ler a informação relativa as apostas realizadas por cada utilizador.
Equipas_FavoritasDao	Classe responsável por adicionar ou ler a informação relativa as equipas favoritas de cada utilizador.
DAO_Config	Classe responsável por configurar a ligação enter o nosso sistema e a base de dados.

Tabela 7: WhiteBox DAO

A classe **DAOConfig** é uma classe de releva importância pois esta, como se encarrega da relação entre a base de dados e o sistema deve ser modificada dependendo da maquina nativa onde se esta a correr o programa.

Building Blocks - Nivel 3

Aposta (WhiteBox)

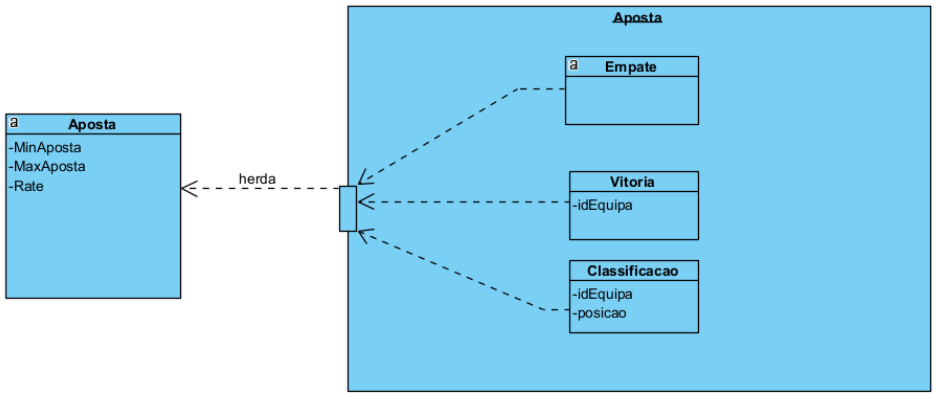

Figura 5: Building Block View - Nivel 2 - Aposta

Raciocínio: De forma a permitir a possibilidade de adição de novos desportos no sistema, a equipa achou necessário generalizar a classe aposta, visto que, diferentes desportos envolvem um ou mais diferentes tipos de apostas.As classes do tipo aposta neste momento são:

- **Vitória**
- **Empate**
- **Classificação**

Building Block	Descrição
Vitoria	Classe que indica apostar na vitória de uma equipa no final de um determinado jogo.
Empate	Classe que indica apostar no empate entre 2 ou mais equipas no final de um determinado jogo.
Classificação	Classe que indica apostar na posição que um classificado ou equipa irá terminar no final de um jogo.

Tabela 8: WhiteBox Aposta

O desporto Futebol terá então a possibilidade de três apostas diferentes, duas apostas do tipo vitória para cada uma das equipas e uma aposta do tipo empate.

No futuro, se decidido fazer apostas sobre o desporto basquetebol este terá então duas apostas disponíveis, uma vitória para cada equipa.

6. Runtime View

Registar

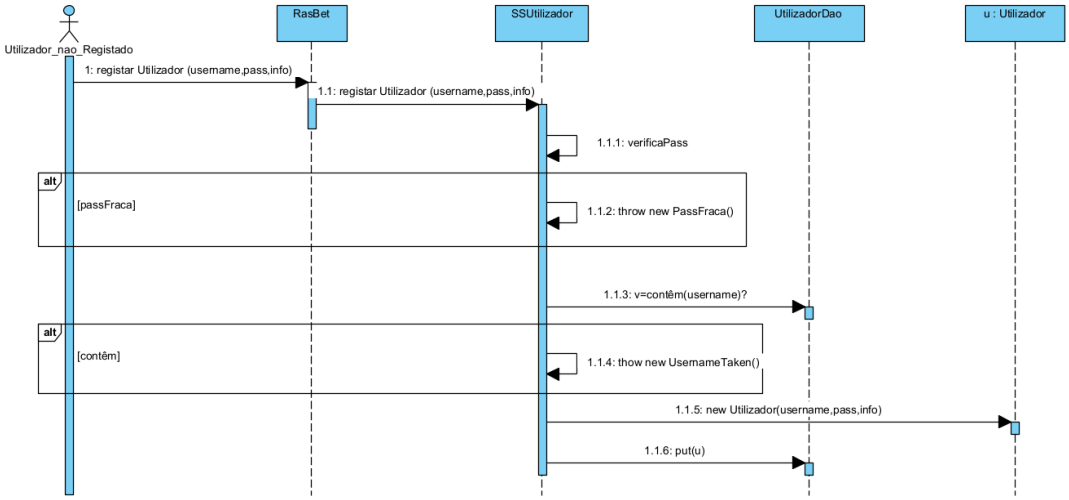

Figura 6: Diagrama de sequência - Registar

Explicação:

1. Utilizador insere credenciais.
2. Ver se a palavra-passe é forte o suficiente.
3. Ver se o username já existe.
4. Criar o utilizador.
5. Adicionar o utilizador a base de dados .

Login

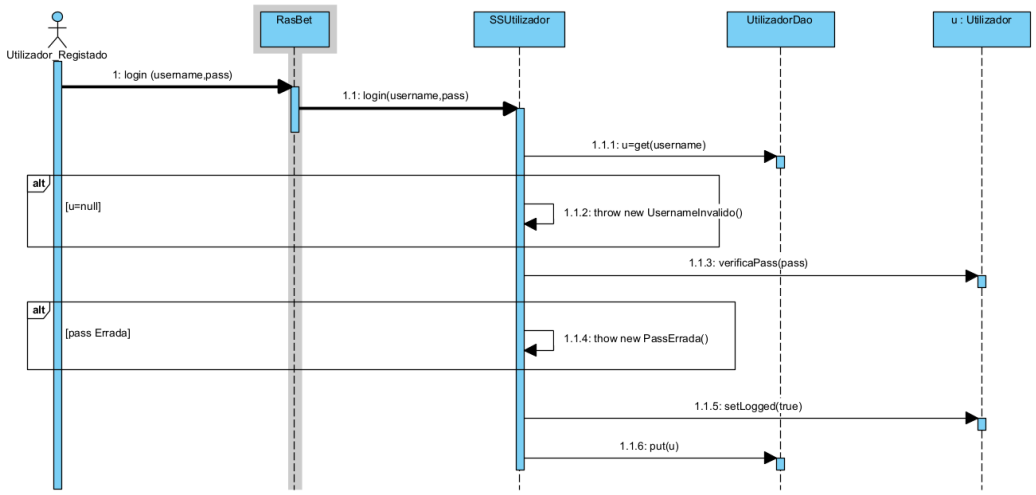

Figura 7: Diagrama de sequência - Login

Explicação:

1. Utilizador insere credenciais.
2. Sistema vê se o utilizador existe na base de dados.
3. Sistema verifica se a palavra passe é correta.
4. Sistema atualiza o utilizador para logged.
5. Sistema adiciona o novo utilizador na base de dados.

Filtrar Jogo

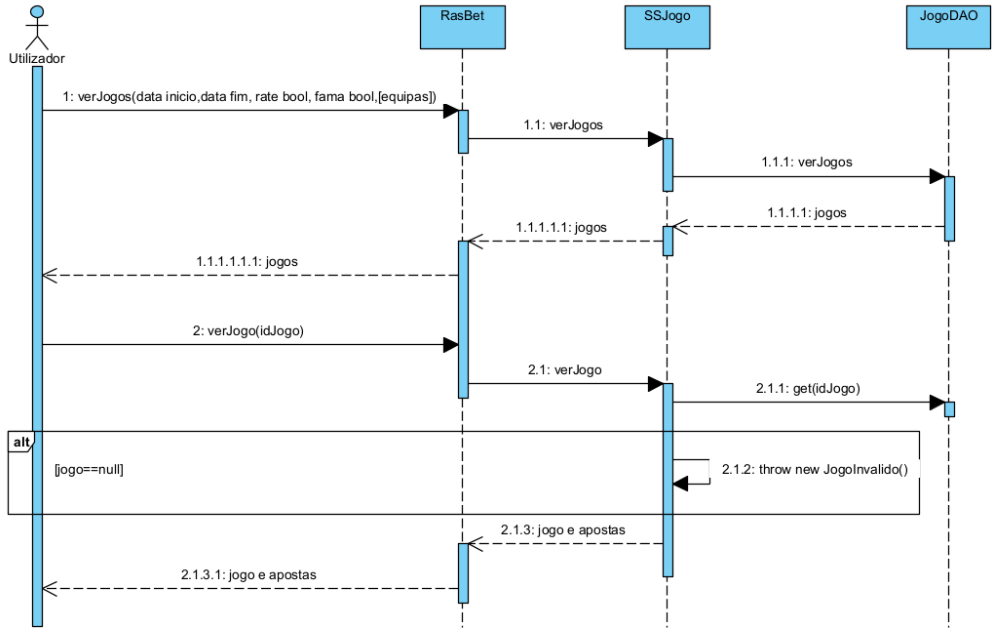

Figura 8: Diagrama de sequência - Filtrar Jogo

Explicação:

1. Utilizador insere conjunto e tipo de filtros que deseja.
2. Sistema filtra os jogos da base de dados numa lista.
3. Sistema Mostrar lista de jogos ao utilizador.
4. Utilizador seleciona Jogo.
5. Sistema acede ao jogo e as suas apostas na base de dados
6. Sistema mostra jogo e apostas ao utilizador.

Fazer Aposta

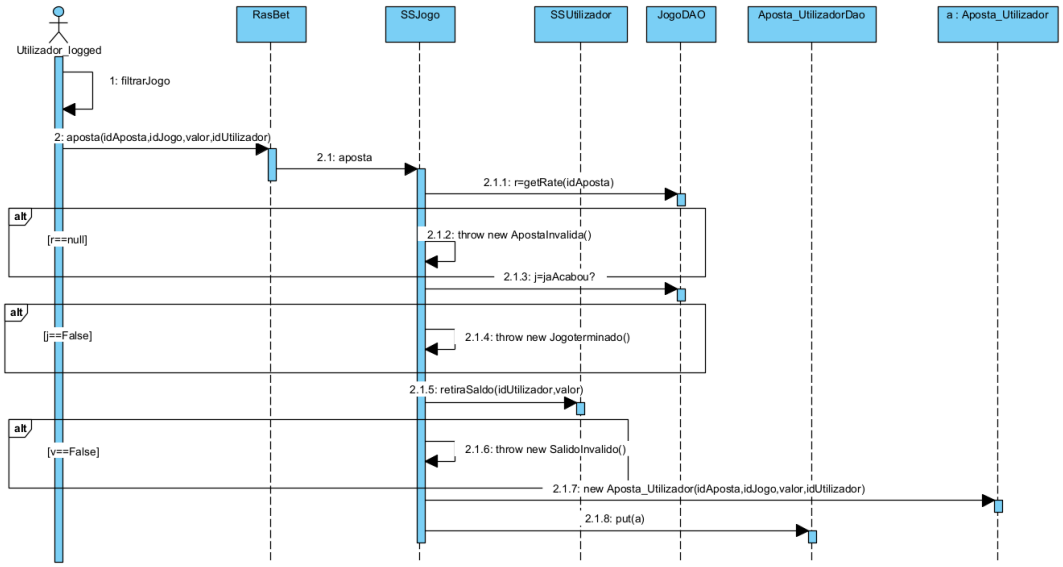

Figura 9: Diagrama de sequência - Fazer Aposta

Explicação:

1. Mostra o jogo e as suas apostas (diagrama Filtrar Jogo)
2. Utilizador indica quanto quer apostar em qual aposta.
3. Sistema verifica se a aposta existe na base de dados.
4. Sistema verifica se o jogo já terminou.
5. Sistema retira saldo ao utilizador verificando se este têm suficiente.
6. Sistema cria registo da aposta por parte do utilizador.
7. Sistema adiciona registo da aposta na base de dados.

Levantar Dinheiro

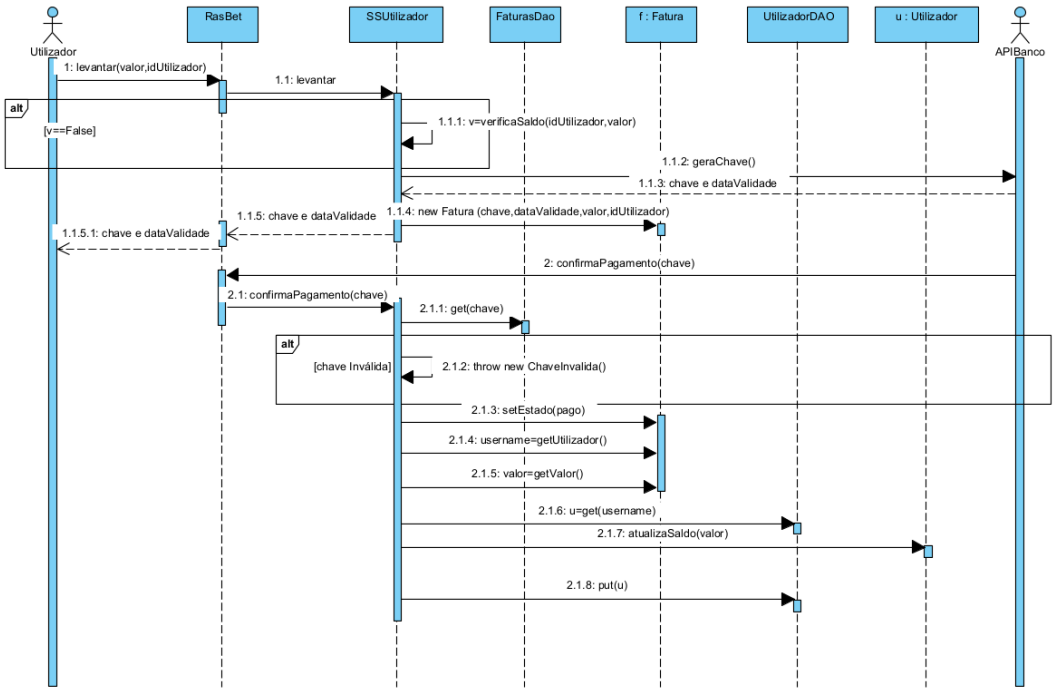

Figura 10: Diagrama de sequência - Levantar Dinheiro

Explicação:

1. Utilizador indica que deseja receber um certo valor.
2. Sistema verifica se utilizador tem saldo suficiente para receber.
3. Sistema pede a API Multibanco para criar chave do pagamento e recebe a chave.
4. API Multibanco confirma o pagamento dando a chave.
5. Sistema verifica se chave existe.
6. Sistema altera o estado da fatura para pago.
7. Sistema atualiza o saldo do utilizador.

Nota: A funcionalidade de depositar dinheiro atua de forma bastante semelhante ao representado no diagrama Levantar Dinheiro, sendo a única diferença considerável o facto de não ser necessário verificar se o utilizador têm saldo suficiente (passo numero 2).

Inserir Jogo

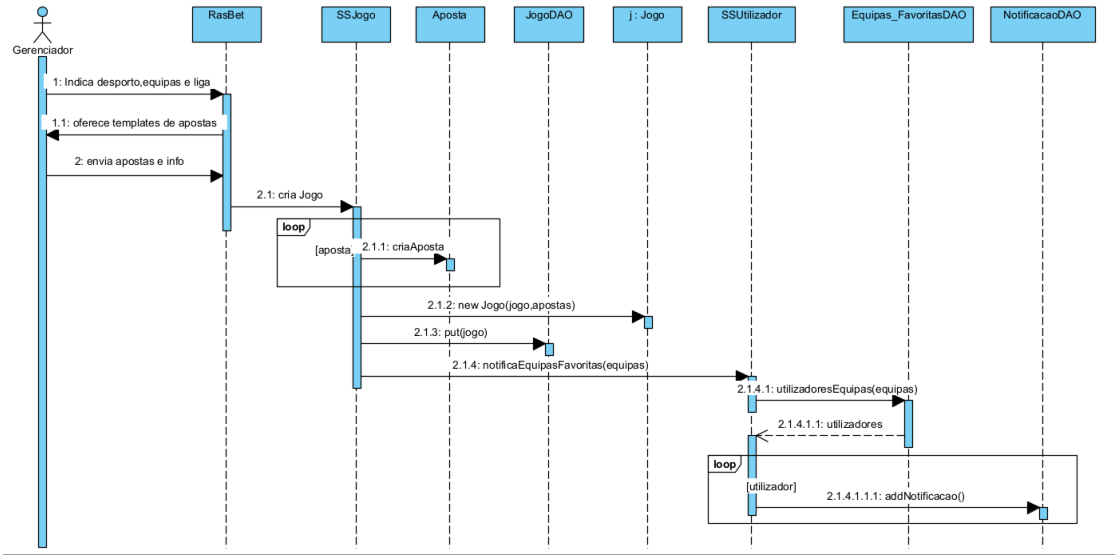

Figura 11: Diagrama de sequência - Inserir Jogo

Explicação:

1. Gerenciador indica o desporto, liga e equipas do jogo a criar.
2. Sistema oferece template de apostas dependendo do desporto.
3. Gerenciador preenche template de apostas.
4. Sistema cria as aposta para o jogo.
5. Sistema cria o jogo e as suas apostas.
6. Sistema adiciona o jogo a base de dados.
7. Sistema acede aos utilizadores que tem como favorito as equipas envolventes no jogo.
8. Sistema adiciona notificações para todos os utilizadores previamente acedidos.

Modifica Jogo

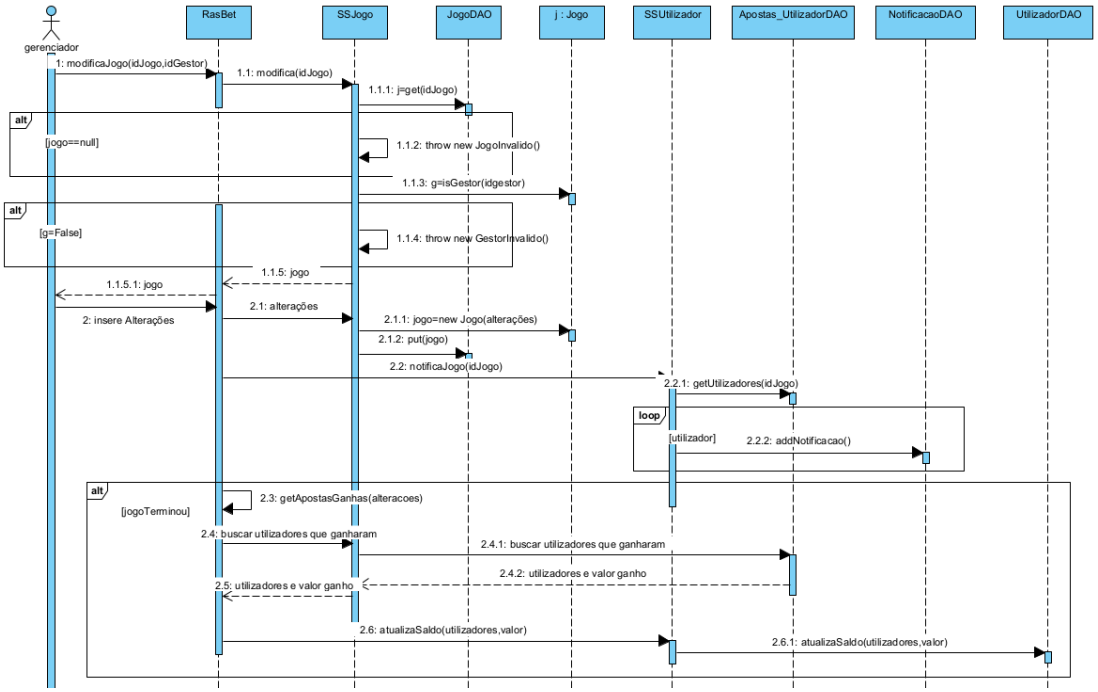

Figura 12: Diagrama de sequência - Modifica Jogo

Explicação:

1. Gerenciador indica jogo que deseja alterar.
2. Sistema verifica se jogo existe na base de dados.
3. Sistema verifica se gestor é o criador do jogo e pode altera-lo.
4. Sistema devolve jogo e mostra-o ao gerenciador.
5. Gerenciador altera o jogo, principalmente estado e apostas.
6. Sistema cria novo jogo.
7. Sistema atualiza jogo na base de dados.
8. Sistema acede aos utilizadores que realizaram apostas naquele jogo.
9. Sistema adiciona notificações para todos os utilizadores previamente acedidos.
10. Se o jogo terminou
    (a) Sistema calcula quais apostas terminaram em sucesso.
    (b) Sistema acede aos utilizadores que apostaram nessas apostas e os valores que cada um ganhou.
    (c) Sistema atualiza o saldo de cada utilizador

Histórico Equipa

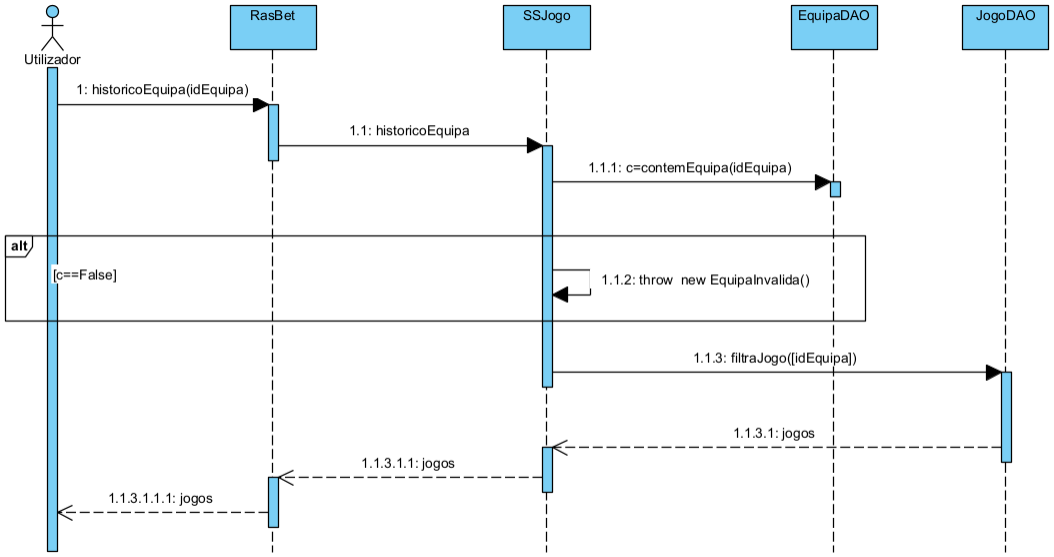

Figura 13: Diagrama de sequência - Histórico Equipa

Explicação:

1. Utilizador indica nome da equipa.
2. Sistema verifica se equipa existe.
3. Sistema acede aos últimos jogos realizador pela equipa na base de dados.
4. Sistema mostra a lista de jogos realizados pela equipa.

8. Crosscutting Concepts

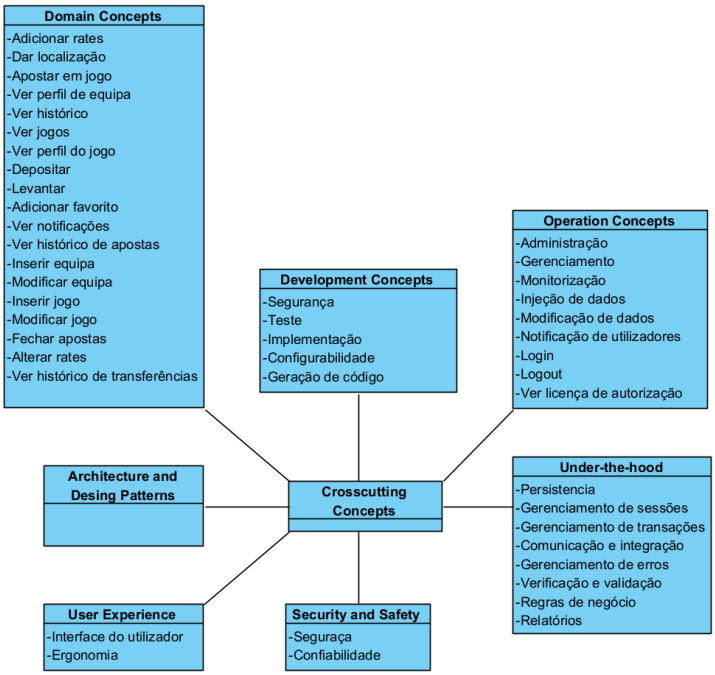

Figura 14: Diagrama de Conceitos

Design Patterns

Em Engenharia de Software, um padrão de desenho (design pattern) é uma solução geral para um problema que ocorre com frequência dentro de um determinado contexto no projeto de software. Iremos de seguida enumerar os padrões que consideramos mais importantes e ajustados ao nosso projeto.

Padrão de Comportamento - Observer

O design pattern 'Observer' será utilizado na mecânica de notificação de utilizadores do sistema. Este padrão define uma dependência um-para-muitos entre objetos de modo que quando um objeto muda o estado, todos seus dependentes são notificados e atualizados automaticamente. Permite que objetos interessados sejam avisados da mudança de estado ou outros eventos ocorrendo num outro objeto. Os utilizadores que apostam numa determinada partida desportiva, são automáticamente registados como observadores do evento de término da mesma. Este padrão vem a propósito dos seguintos pontos:

1. Uma relação de dependência de um para muitos deve ser definida entre diferentes objetos (neste caso jogo e utilizador) sem os ligar fortemente.

2. Deve ser assegurado que quando o objeto observado muda de estado, os varios dependentes são automaticamente atualizados.

3. Deve ser assegurado que um objeto por si só seja capaz de notificar um grande número de outros objetos

Assim que o jogo conclui, o objeto observado percorre todos os observadores e notifica-os. Uma vez que o uso de notificações são uma prática bastante atual no desenvolvimento moderno, o uso deste tipo de padrões para desenvolver estes sistemas já está estudado e presente em vários outros softwares. Este padrão pode ser posteriormente reutilizado para diferentes tipos de notificações: término ou início de jogo, alteração do resultado, cancelamento ou adiamento da partida, etc.

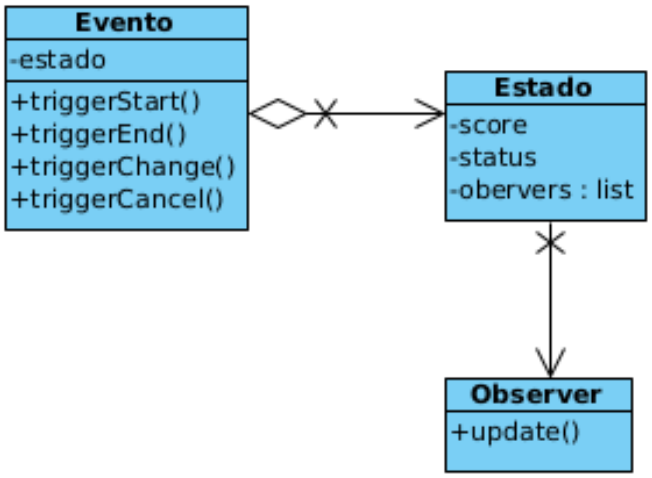

Figura 15: Representação do Padrão Observer

Padrão Estrutural - Facade

O Padrão de projeto Facade (ou Fachada) é um padrão de design de software usado comumente com programação orientada a objetos. Um Facade é um objeto que provê uma interface simplificada para um corpo de código maior, como por exemplo, uma biblioteca de classes. Ñuma primeira análise, este padrão será indispensável no que toca a gerir a nossa aplicação uma vez que este agrega diferentes módulos com interfaces diferentes (incluindo a possibilidade de software de terceiros, assim como foi referido no documento de requisitos) e dessa forma é necessário um mecânismo capaz de gerir, manter e alterar diferentes componentes de uma forma simples e direta.

Como consequência, temos os seguintes pontos:

1. Torna o sistema mais fácil de se usar, protegendo os clientes dos componentes do sistema, reduzindo o número de objetos que terão que lidar. fortemente.
2. Promove fraco acoplamento entre os subsistemas e seus clientes.
3. Não evita que as aplicações possam acessar as subclasses diretamente, pode-se escolher entre a facilidade de uso ou a generalidade.

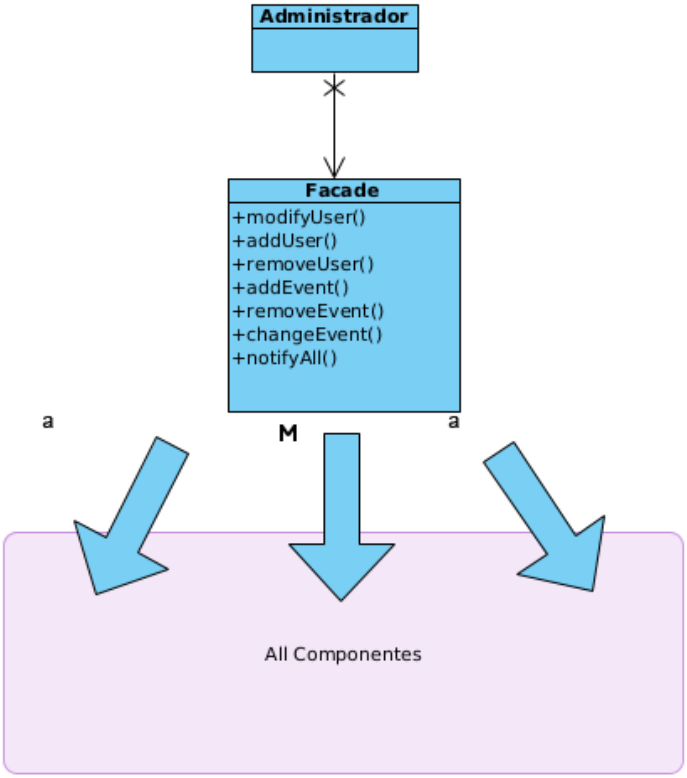

Figura 16: Representação do Padrão Facade

Padrão Criacional - Object Pool

Object Poll é um padrão de software que usa e reusa objetos inicializados e prontos a consumir em vez de alocar e destroí-los por demanda. Na aplicação poderá ser utilizado este padrão para gerir o grande número de eventos em simultâneo, utilizando por exemplo uma thread pool: diferentes threads encarregam-se de tarefas diferentes e quando se encontram livres escolhem uma nova.

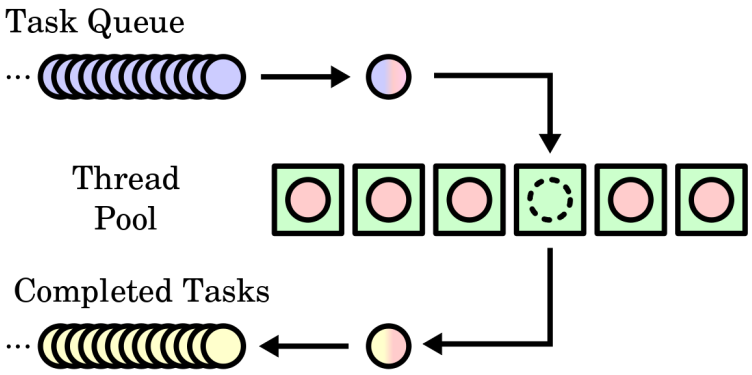

Figura 17: Representação do Object Pool

9. Architectural Decisions

Diversas decisões de arquitetura foram sido tomadas ao longo do desenvolvimento desta segunda fase assim como da primeira, portanto, cabe apenas realçar aqui a importância do padrão MVC que seguimos no desenvolvimento da arquitetura assim como o diagrama de caixas no tópico 5 deste relatório que estrutura a organização de todo o código a desenvolver em fases posteriores e por fim a importância dos DAOs pois serão os objetos que irão interagir diretamente com a base de dados.

10. Quality Requirements

A seguinte tabela é uma pequena referência ao trabalho realizado nos capítulos 10, 11, 12, 13, 15 e 17 do relatório RASBet-1 onde esta mencionado uma descrição muito mais detalhada dos requisitos não funcionais, incluindo uma tabela e pequeno texto para cada categoria de requisito.

Categoria Qualidade	**Qualidade**	**Descrição**
Aparência	Personalização	A interface do software tem de ser personalizável ao gosto e necessidades dos seus utilizadores
Usabilidade	Facilidade	O software deverá ser fácil de usar pelos utilizadores
Performance	Rapidez	O software tem de responder a um pedido do utilizador em menos do que 3 segundos
Operacionalidade	Conexão	O software deverá ser inutilizável caso não exista conexão permanentecom os servidores da empresa fornecedora
Legalidade	Exibição	Cada empresa compradora tem de ser portadora de uma licença para autorização na utilização do software.

Tabela 9: Requisitos de qualidade

11. Risks and Technical Debt

Riscos de Eficiência

Tratando-se de um sistema que envolve uma grande quantidade de utilizadores simultaneamente em uso e de imensa informação toda esta em constante alteração, a equipa reconhece a dificuldade em manter o serviço rápido e eficiente para todos os seus usuários. Não existindo grande experiência na criação de sistemas distribuídos e outras estratégias que consigam alternar o problema, reconhecemos que a decisão inicial de manter uma única base de dados centralizada provavelmente não será a melhor das soluções e certamente será necessário no futuro despender de tempo e pesquisa no desenvolvimento de uma solução de calibre superior.

Riscos de Segurança

A probabilidade de serem encontradas falhas no sistema que tornem capazes a terceiros a observação de informação discreta ou conseguir contornar os mecanismos do sistema e tornar possível a realização de atividades que prejudiquem o seu funcionamento natural, como aumentar o saldo sem realizar o pagamento, ou fazer apostas sem saldo suficiente, ou com rates desatualizados é uma preocupação de nível elevado para a equipa, será necessário atribuir uma boa quantidade de testes á procura destas vulnerabilidades e provavelmente uma forma mais segura de guardar e partilhar toda a informação do sistema.

Riscos de Robustez

Para um sistema que deve estar sempre disponível para os seus utilizadores, a prespetiva de ter uma falha nas APIs necessárias ou na base de dados ou qualquer outro componente do sistema tanto software como hardware deixa a equipa um pouco a preocupada, devido a pequena quantidade de contra medidas preparadas para tal, assim, no caso de o sistema for direcionado para ambientes de elevada dimensão e alto risco será então necessário corrigir esta debilidade.

12. Glossary

- **ERA**: Entidade Reguladora de Apostas
- **GCLR**: Garante Cumprimento de Leis e Regulamentos
- **IAJ**: Instituto de Apoio ao Jogador
- **SD**: *System Developer*
- **GP**: Gestor de Projecto
- ***Odds***: Valor associado à probabilidade de um determinado evento ocorrer
- **Cliente**: Empresa ou Sujeito que irá comprar o produto.
- **Utilizador**: Sujeito que usa o produto.
- ***Rates***: Multiplicador de valor ganho nas apostas.
- **MVC**:Model-View-Controller.
- **DAO**:Data Acess Object.
- **API Rates**:API externa responsável por fornecer e atualizar informação sobre jogos e apostas ao nosso sistema.
- **API MultiBanco**:API externa responsável por gerar chaves associadas a transferências bancarias e partilha-las com o sistema assim como confirmar esses mesmos pagamentos.