UMinho
Mestrado Engenharia Informática
Requisitos e Arquiteturas de Software

Grupo PL1-3, PL1 / Entrega 3

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




16 de janeiro de 2023

2022/23

Prefácio

Nesta terceira fase, o grupo necessitava de completar a entrega anterior com algumas funcionalidades que não estavam ainda implementadas na aplicação, assim como desenvolver os novos requisitos propostos pelos docentes. Desta forma, o desenvolvimento desta terceira fase do trabalho prático decorreu de forma linear, sem grandes percalços.

Relativamente aos novos requisitos, em concreto o requisito de implementação de **apostas múltiplas**, este já se encontrava presente na aplicação desde a primeira entrega, sendo um dos requisitos presentes no documento entregue pelo grupo. Assim, não houve grandes alterações ou reestruturações de código, uma vez que todas as funcionalidades já estavam implementadas.

No que toca ao requisito de um apostador ser capaz de **seguir jogos**, este foi implementado com sucesso, tendo sido construído tendo por base um *desgin pattern* com o objetivo de facilitar a compreensão do comportamento do mesmo. Para além disto, a integração deste não alterou significativamente a estrutura e arquitetura da aplicação, mostrando que a solução atual possui flexibilidade e permite a expansibilidade de outras funcionalidades.

Por fim, apresentamos uma classificação do contributo dos membros para a terceira fase do trabalho, tal como requerido pelos docentes:

- Anabela Pereira -1
- Fernando Lobo +1
- Joana Alves +1
- Vicente Moreira -1

UMinho

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

RASBET: Solução Arquitetural
Entrega 3

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

Braga, 16 de janeiro de 2023

1. Expansão de Funcionalidades da Aplicação

1.1 Novos Requisitos

Nesta parte do projeto, foram identificadas certas funcionalidades consideradas essenciais para o sucesso da plataforma de apostas desportivas desenvolvida, sendo o objetivo implementá-las da forma mais flexível e modular possível, não sendo necessário grandes reestruturações.

- **Apostas Múltiplas -** Os apostadores poderão apostar em vários resultados simultaneamente, num total máximo de 20, sem que estes sejam relativos ao mesmo jogo. Estas apostas oferecem prémios maiores, mas também possuem um risco maior.
- **Seguir Jogos -** Os apostadores poderão seguir jogos específicos, de modo a serem notificados quando as cotas de algum dos seus resultados for alterada.

1.2 Requisitos Já Cumpridos

O requisito de apostas múltiplas já se encontra presente na solução da plataforma desenvolvida, sendo este um dos requisitos funcionais das entregas anteriores.
O apostador é capaz de fazer uma aposta de cota acrescida, sendo esta calculada a partir da multiplicação das cotas presentes em todos os resultados apostados. Esta aposta apenas é considerada "ganha" e o seu prémio entregue quando todos os resultados na aposta forem confirmados como os resultados vencedores.

1.3 Seguir Jogos

Nesta secção iremos explorar todos os aspetos acerca do planeamento, considerações e implementação final da funcionalidade "Seguir Jogos".

1.3.1 *Design Pattern*

Os *design patterns* permitem acelerar o processo de desenvolvimento ao fornecer paradigmas de desenvolvimento testados e provados. Assim, procuramos um paradigma que se adequasse à funcionalidade Seguir Jogos, tendo-o encontrado no domínio dos **padrões comportamentais**: padrão ***Observer***. Este padrão consiste na definição de uma dependência do tipo 1:N entre objetos para que quando um objeto muda de estado, todos os dependentes deste são notificados e atualizados automaticamente.
O objeto a ser observado (***Subject***) trata-se das cotas de um determinado jogo e o objeto dependente deste (***Observer***) trata-se de Apostadores com Registo. Assim, foi necessário criar uma tabela de correspondência entre que utilizadores seguiam quais jogos (mais informação na secção seguinte). Para além disto, era também necessário desenvolver *procedures* para adicionar ou remover informação nessa tabela, assim como lidar com a notificação de mudanças de cotas num jogo que se encontrasse a ser seguido.

1.3.2 Expansão da Base de Dados

Para o modelo lógico da Base de Dados, encontramos apenas necessário criar uma nova relação entre os apostadores e os jogos, sendo mais específico, uma relação N para N, visto que vários apostadores poderão seguir vários jogos e cada jogo poderá ser seguido por vários apostadores independentes. Esta nova relação resultou numa nova tabela 'jogo_has_apostador'.

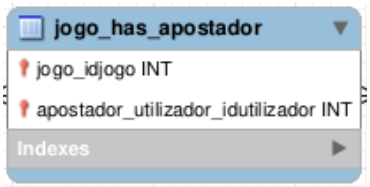


Figura 1.1: Nova tabela 'jogo_has_apostador'

Também foram necessárias algumas mudanças na lógica das *procedures* de retorno da informação de um apostador, visto que estes dados tiveram de ser alterados para incluir a lista de identificadores dos jogos que o apostador segue. Esta necessidade será explicada no capítulo seguinte.
Como a aplicação desenvolvida tem a maioria do seu poder computacional nesta camada, a definição dos métodos do *desgin pattern* escolhido foram transformados em *procedures* ou *triggers*. Assim, fazemos a correspondência entre os mesmos, incluindo uma pequena descrição:

Métodos	***Procedures***	**Descrição**
notifyObservers	notifica_jogadores	Notifica todos os apostadores que seguem o jogo após um UPDATE de cotas no mesmo (*trigger*)
addObserver	followGame	Adiciona uma entrada na tabela acima com o *id* do jogo e o *id* do apostador
removeObserver	unfollowGame	Elimina a entrada na tabela acima com o *id* do jogo e o *id* do apostador
update	notifica_jogadores	Presente no corpo desta *procedure*, criando notificações a todos os *observers*

1.3.3 Expansão do *Back-end*

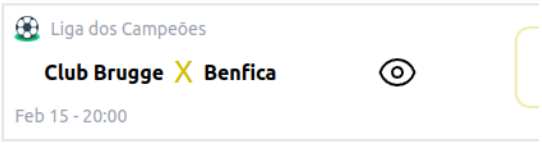
Para permitir que os apostadores pudessem seguir os novos jogos foi preciso acrescentar novas funções ao *back-end*, assim como expandir a API desta:

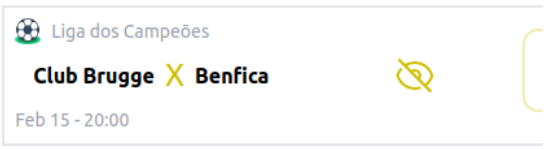
- **POST /seguir** - Recebe o *token* do utilizador autenticado, o *id* do jogo e uma *flag* "add"que indica qual a ação a efetuar (seguir ou deixar de seguir).
- Função **startFollowing** - Cria entrada na tabela 'jogo_has_apostador', ligando o utilizador ao jogo que este decidiu seguir.
- Função **stopFollowing** - Destrói a entrada na tabela correspondente a esse jogo, deixando o apostador de seguir esse jogo.

Outra alteração que achamos necessária foi o modo de envio da informação de quais jogos um certo apostador segue. Inicialmente planeamos por modificar a função de pedido de jogos para aceitar um *id* de apostador, ao qual o *back-end* iria responder com a lista de jogos e uma *flag* a indicar se este jogo está a ser seguido pelo apostador.
No entanto esta solução iria interferir no funcionamento do pedido de jogos dos Apostadores sem Registo e traria problemas na velocidade de acesso aos jogos para os apostadores com registo pois seria necessário fazer dois pedidos GET sequenciais, um para pedir as informações acerca do apostador autenticado e outro para pedir os jogos com o *id* associado, trazendo possíveis problemas de eficiência e atraso de resposta da aplicação.
No final, decidimos que a lista dos jogos que um apostador segue, visto que são de caráter "pessoal", deveriam ser enviados juntamente com a informação do apostador.

1.3.4 Expansão da View

No lado da *View*, foi necessário implementar novos ícones e lógica para seguir um jogo assim como adaptar ao novo formato de dados do utilizador proveniente do *back-end*. Também foi necessário alterar a *loginStore* para que os jogos pudessem consultar a lista de jogos seguidos pelo apostador de forma a saber se já estão a ser seguidos por este ou não.





2. Solução Arquitetural Final

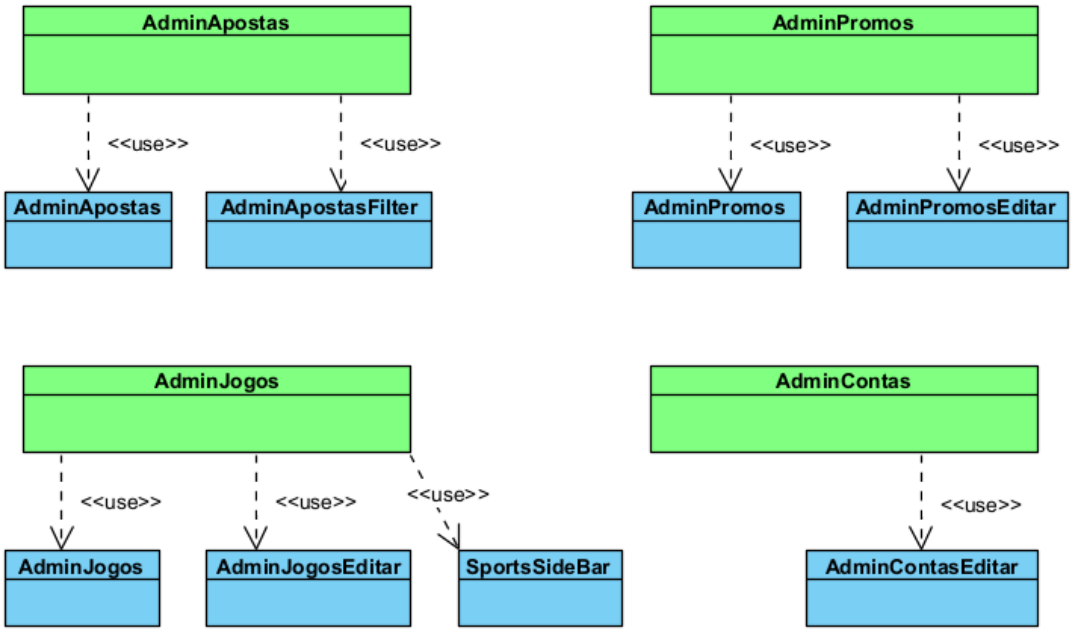
Relativamente à solução arquitetural, esta sofreu pequenas mudanças tal como referido na secção acima. No entanto, é também de salientar as adições de métodos e classes implementadas de modo a concluir as funcionalidades pedidas na segunda entrega. Assim, apresentamos os diagramas atualizados nas várias camadas da aplicação.

2.1 *View*

Nesta secção vamos apresentar apenas os componentes que sofreram alguma alteração, como a utilização de novos ficheiros ou remoção de antigos, tendo a sua visão mais pormenorizada sido entregue na fase anterior.

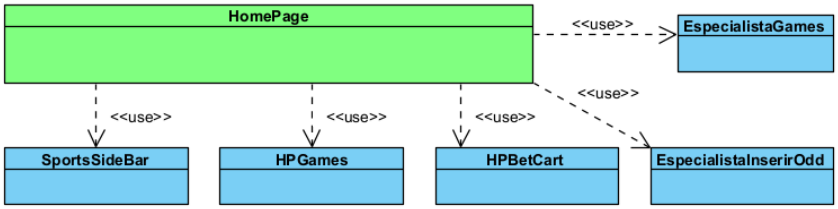
2.1.1 *Admin*

O administrador possui várias páginas de gestão, no entanto, as que sofreram alterações foram as páginas de gestão de Jogos e de Apostas. No caso da página de **Jogos**, foi adicionado o ficheiros *SportsSideBar* de modo a permitir a filtragem da listagem de jogos do sistema por desporto e competição. No caso das **Apostas**, foi adicionado o ficheiro *AdminApostasFilter* de modo a permitir a filtragram das apostas do sistema por jogo ou utilizador. Para além disto, foi removido o ficheiro *AdminApostasEditar*, uma vez que foi inutilizado.



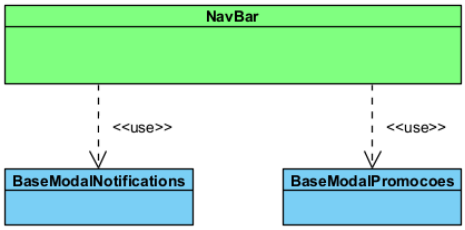
2.1.2 *HomePage*

Nesta página foi apenas adicionado um ficheiro para apenas dar *display* no caso de o utilizador com sessão iniciada se tratar de um especialista: *EspecialistaGames*.



2.1.3 *NavBar*

Na componente da barra de navegação, de modo a permitir a visualização das promoções presentes no sistema, foi adicionado o ficheiro *BaseModalPromocoes* que lista as promoções ativas do sistema.



2.2 *Business Logic*

Nesta camada denotamos a adição de vários métodos, como, por exemplo, na interface *IRasbetJogos*, assim como a criação de uma nova interface de gestão de promoções: ***IRasbetPromocoes***.

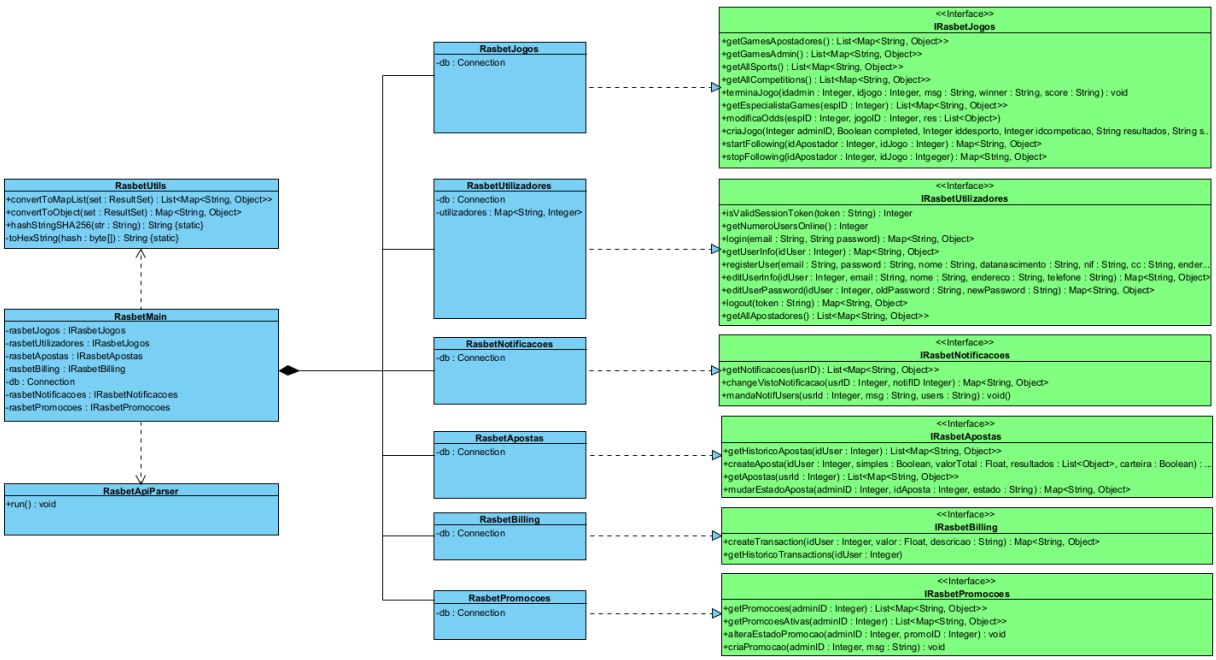


Figura 2.1: Diagrama de Classes da *Business Logic*.

2.3 Base de Dados

Na camada de base de dados da aplicação, tal como referido acima, a única diferença foi a inclusão de uma tabela de relação de N para N entre apostadores e jogos. Assim, passamos a apresentar o novo modelo lógico:

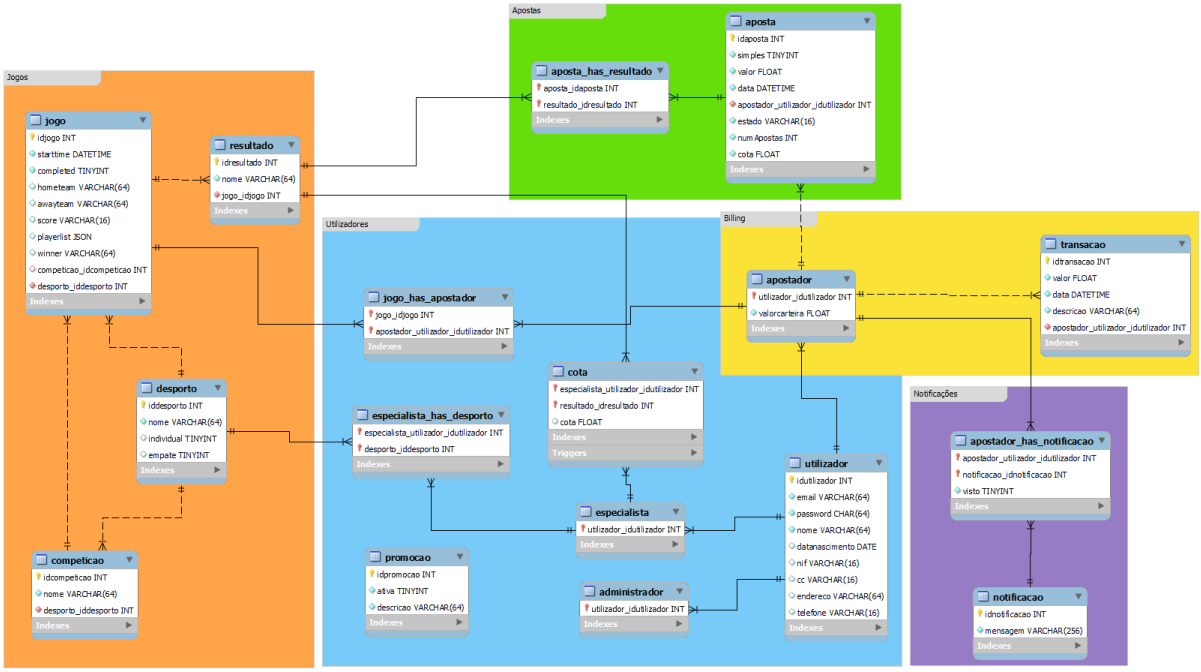


Figura 2.2: Modelo Lógico da Base de Dados.