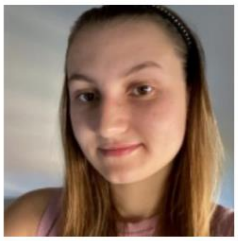
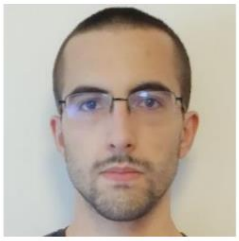
Mestrado em Engenharia Informática

Requisitos e Arquiteturas de Software

Grupo X – PL4
Ana Filipa Rodrigues Pereira PG46978
Bruno Alexandre Dias Novais de Sousa PG45577
Carolina Gil Afonso Santejo PG47102
Raquel Sofia Miranda da Costa PG47600

18 de janeiro de 2022
2021/2022

ÍNDICE

I. INTRODUÇÃO

Nesta terceira e última fase do projeto da unidade curricular de Requisitos e Arquiteturas de *Software* foi pedido ao grupo que implementasse a aplicação *RASBet* tendo em conta todo o processo de desenvolvimento efetuado nas fases anteriores.
Além disto, foi necessário evoluir a aplicação e implementar um novo requisito de forma a permitir que as carteiras dos utilizadores fossem multi-moeda e que estes possam apostar em eventos utilizando a moeda que desejarem.

Ao longo deste documento será descrito não só como foram adicionadas as novas funcionalidades e as alterações que isto implicou na arquitetura, mas também serão abordadas algumas mudanças feitas à fase anterior.

II. ADIÇÃO DOS NOVOS REQUISITOS

Nesta fase, a equipa docente pediu ao grupo que evoluísse a aplicação de forma a permitir a adição de novas funcionalidades sendo que estas consistiram em permitir que o utilizador possua uma carteira multi-moeda e que possa apostar num dado evento usando a moeda que desejar.

Desta forma, a primeira mudança que se efetuou foi adicionar uma classe *currency*, a qual possuiria um identificador único para cada moeda e o seu respetivo nome (por exemplo, o *id* EUR corresponde ao nome Euro). Além disto, a classe *currency* possui ainda uma estrutura *hashmap*, denominado *currencyTaxes*, no qual a *key* corresponde ao *id* de uma outra *currency* e o *value* ao valor da taxa de câmbio para essa mesma moeda.

Por outro lado, e de maneira a permitir as carteiras com várias moedas, foi acrescentado à classe do utilizador uma estrutura *hashmap* denominada *wallets*, na qual a chave corresponde ao *id* de uma *currency* e o *value* ao valor monetário que o *user* possui nessa mesma *currency*. Além disto, foi também preciso adicionar à classe *Bet* uma variável com o *id* da moeda de forma a identificar que moeda foi utilizada quando a aposta foi realizada.

É importante também realçar que, nesta aplicação, o utilizador poderá adicionar ao seu boletim várias apostas utilizando diferentes unidades monetárias em cada uma. Quando o boletim for submetido, será calculado o valor total a pagar em cada uma das *currencies* usadas e, por fim, caso o utilizador possua saldo (nas várias moedas), os valores monetários serão cobrados na sua *wallet*.

Além do que já foi descrito, foi também criado um *DAO* para a *currency* e outro para as taxas de câmbio de maneira a persistir as informações relativas a estes dois fatores.

Concluindo, e tendo em conta o que foi descrito neste tópico é possível afirmar que a adição das novas funcionalidades não implicou muitas dificuldades nem alterações muitas significativas à arquitetura do sistema que já tinha sido planeada.

III. ALTERAÇÕES REALIZADAS NA FASE ANTERIOR

A. MODELO LÓGICO

Nesta aplicação, e visto que se tem em conta a persistência de dados, foi desenvolvido um modelo lógico no qual se identifica as várias entidades cuja informação será guardada na base de dados, bem como a relação entre elas.

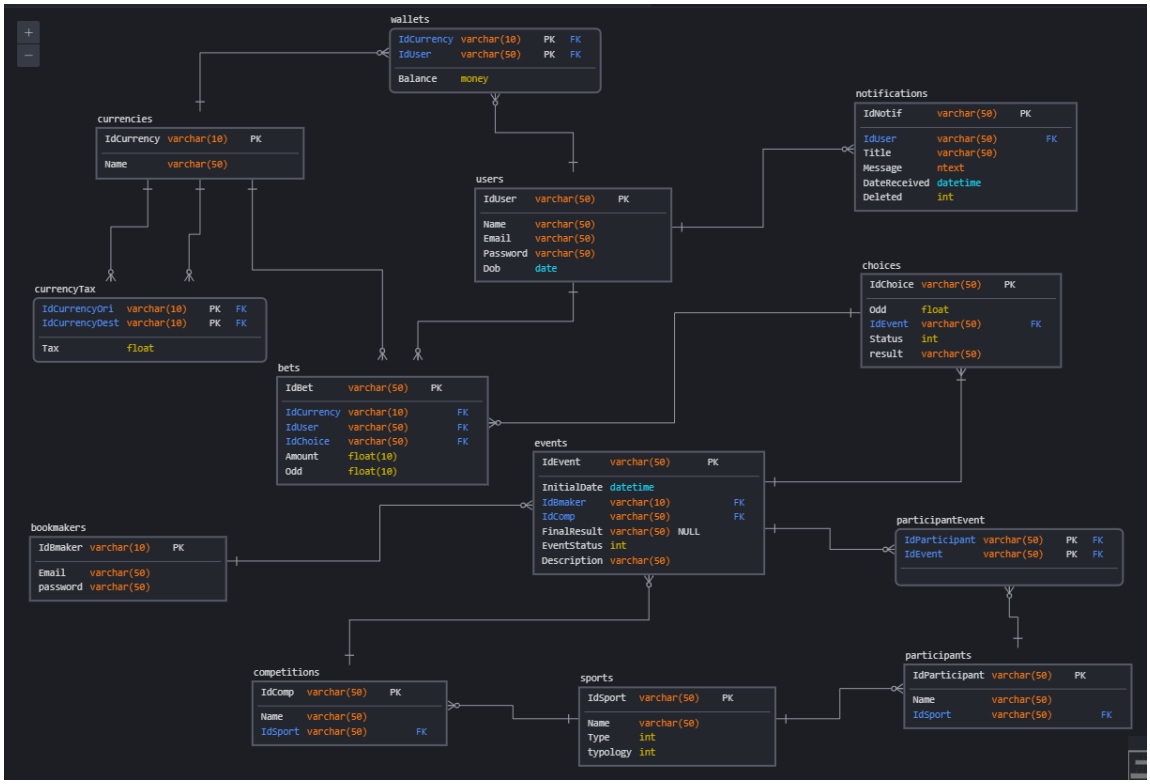


FIGURA 1-MODELO LÓGICO DA BASE DE DADOS

B. DIAGRAMA DE CLASSES

1. USERBL

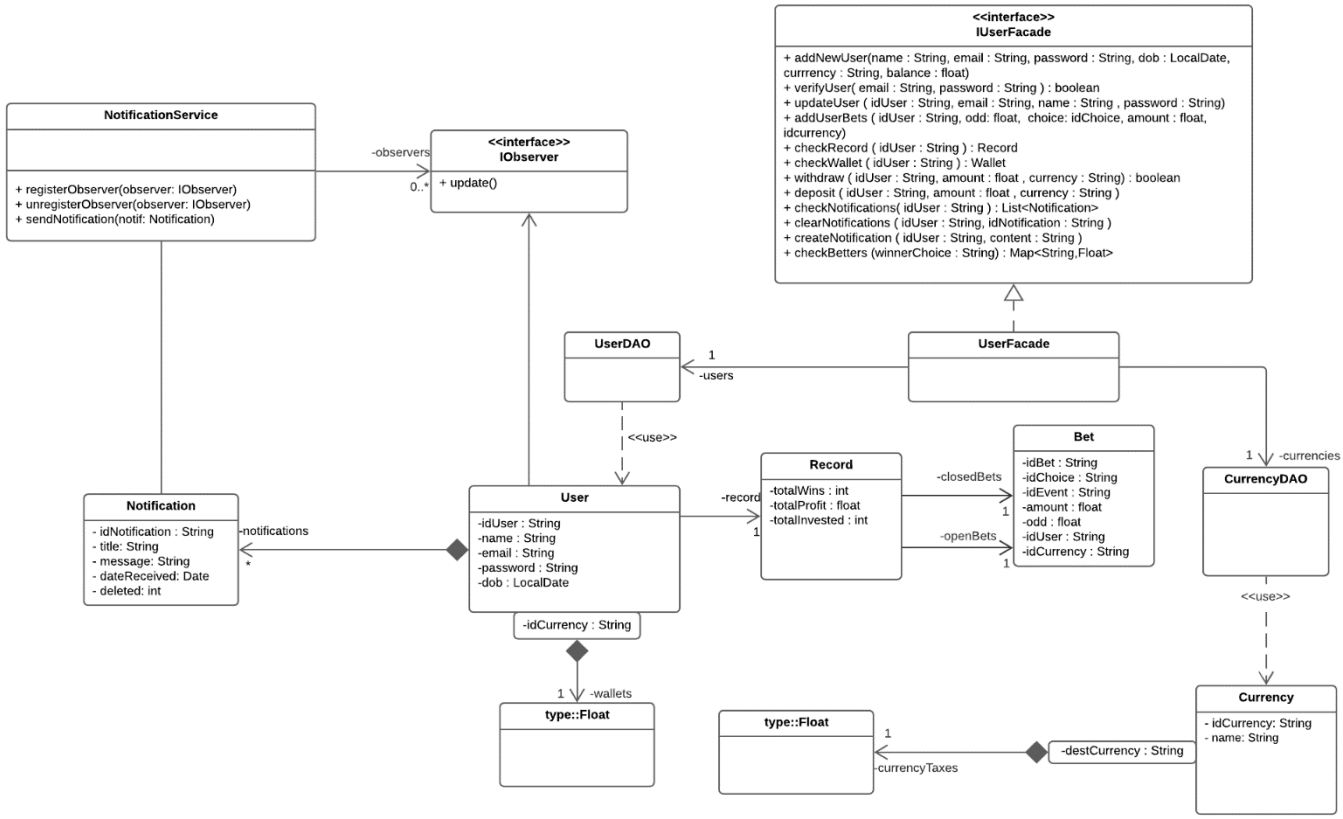


FIGURA 2 - DIAGRAMA DE CLASSES ATUALIZADO P/SUBSISTEMA "USERBL"

No presente diagrama de classes, foi necessário incluir algumas alterações não só por causa do novo requisito proposto pela equipa docente, como também devido ao facto de que quando começamos a implementação da aplicação reparamos que foi necessário realizar algumas mudanças.

Um dos exemplos é a alteração dos DAO's, uma vez que com a estruturação da base de dados e implementação da mesma, reparamos que existiam alguns aspetos a definir.

2. BOOKMAKERBL

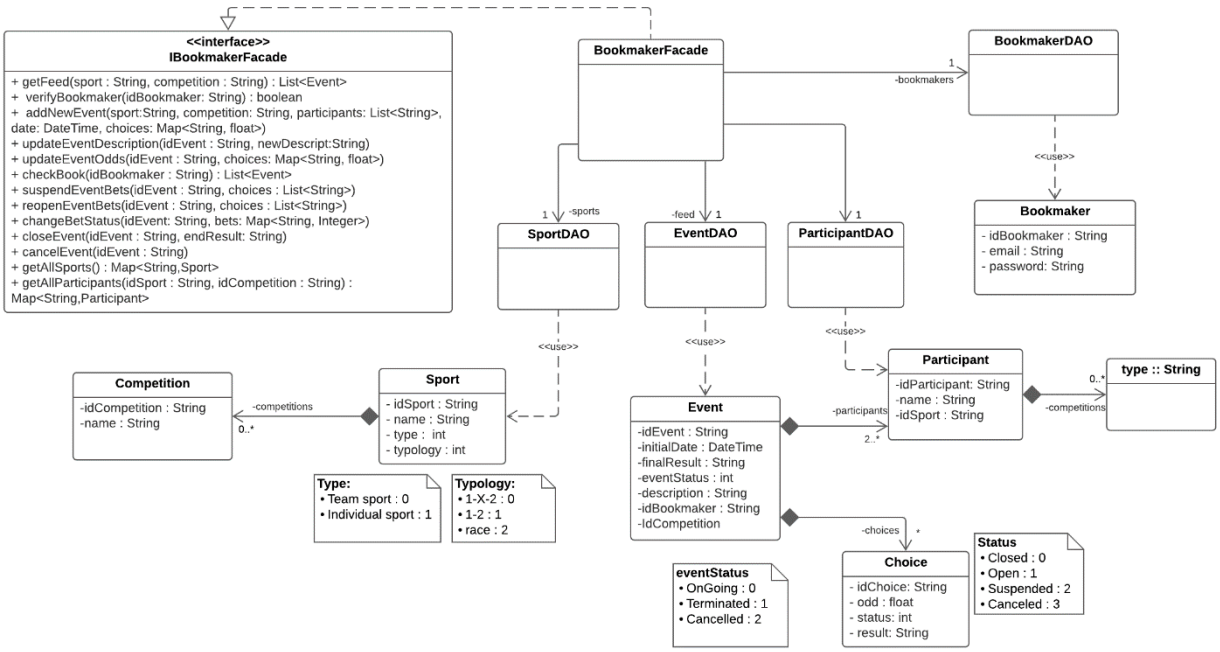


FIGURA 3 - DIAGRAMA DE CLASSES ATUALIZADO P/SUBSISTSEMA "BOOKMAKERBL"

Em relação ao presente subsistema, também foi necessário realizar alterações, uma vez que notamos que a versão anterior era bastante redundante.

IV. DIAGRAMA DE PACKAGES

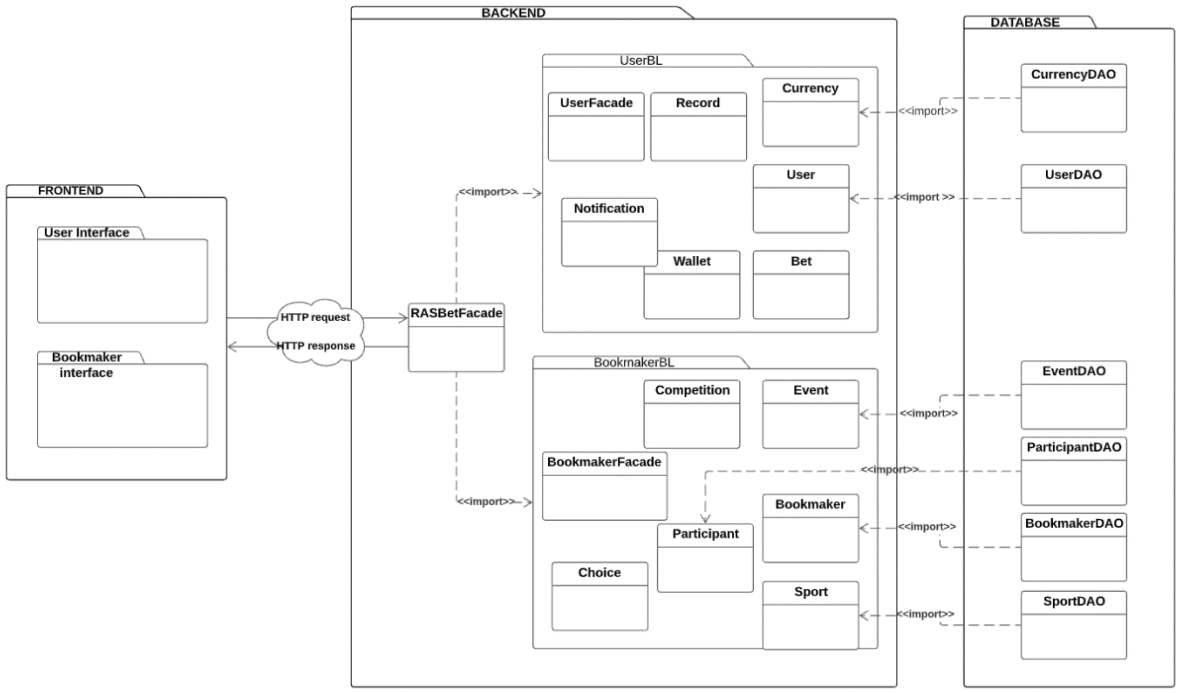


FIGURA 4 - DIAGRAMA DE PACKAGES ATUALIZADO

É de notar, que retiramos também a classe Controller, uma vez que, o frontend encarrega-se de comunicar com a classe RASbetFacade, de modo a conseguir fazer os devidos pedidos à lógica de negócio. Além disso, a estratégia abordada anteriormente, mostrou-se ser um bocado redundante.

V. CONCLUSÃO

A realização deste trabalho prático permitiu ao grupo entender melhor as várias fases que fazem parte do processo de desenvolvimento de *software*, começando no levantamento e análise de requisito, passando pelo planeamento da arquitetura e por fim a implementação da solução final.

Esta aplicação foi desenvolvida de forma faseada, tendo cada fase sido cuidadosamente planeada e analisada. Isto contribuiu para que a *RASBet,* além de já possuir várias funcionalidades típicas de uma app de apostas, permita a adição de novos requisitos sem que sejam preciso alterações muito significativas na sua arquitetura. Por estas razões o grupo considera que realizou este projeto prático com sucesso.