



Universidade do Minho - Escola de Engenharia

Relatório do trabalho prático de Arquiteturas de Software

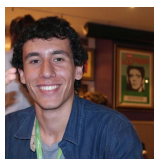
BetESS

Autores :

Diana Costa (A78985)



Marco Silva(A79607)



Versão 1.0
13 de Dezembro de 2018

Resumo

Neste relatório será feita uma abordagem ao projeto de Arquiteturas de Software ao qual está associado o desenvolvimento de um programa, em Java, responsável pela gestão de uma plataforma de apostas - BetESS. Foi elaborada uma versão sem qualquer padrão de arquitetura ou design, e outra aplicando o padrão arquitetural MVC (Model-View-Controller), juntamente com o padrão de design Observer. Para além disto, foi implementada uma última versão de acordo com um requisito lançado a dois dias da entrega do projeto. Assim, este documento apresenta, detalhadamente, a perspetiva tomada pelo grupo em relação ao problema proposto pela equipa docente de AS.

Conteúdo

1	Introdução	4
2	Problema	4
3	Requisitos	5
3.1	Requisitos funcionais	5
3.2	Requisitos não-funcionais	10
3.3	Especificação de requisitos de qualidade	11
4	Modelação UML	12
4.1	Modelo de Domínio	12
4.2	Diagrama de Use Cases	13
4.3	Diagramas de Sequência de Sistema	14
4.3.1	Fazer aposta	14
4.3.2	Adicionar evento desportivo	15
4.3.3	Alterar estado de evento desportivo	16
4.4	Diagrama de Instalação	17
4.5	Diagrama de Classes - versão 1	18
4.5.1	Breve explicação da solução	18
4.5.2	Diagrama	19
4.6	Diagrama de Classes - versão 2	20
4.6.1	Breve explicação da solução	20
4.6.2	Diagrama	22
4.7	Comparação conclusiva e concreta entre as duas versões	23
5	Introdução do novo requisito - mudanças	24
6	Mockups	25
6.1	Login/Registo	25
6.2	Área autenticada - Jogador	26
6.2.1	Apostar	26
6.2.2	Minhas apostas	27
6.2.3	Créditos	27
6.2.4	Notificações	28
6.2.5	Editar perfil	29
6.3	Área autenticada - Administrador	30
6.3.1	Ver jogadores	30
6.3.2	Eventos Desportivos	31
6.3.3	Registar evento desportivo	31
6.3.4	Jogadores Bloqueados	32
6.3.5	Apostas	33
6.3.6	Nova Equipa	33
6.3.7	Nova Liga	34
6.4	Área autenticada - Bookie	34
6.4.1	Eventos Desportivos	34
6.4.2	Adicionar evento desportivo	35
6.4.3	Notificações	35

7	Implementação	36
7.1	Interface Gráfica	36
7.1.1	Login/Registo	36
7.2	Área autenticada - Jogador	36
7.2.1	Apostar	36
7.2.2	Minhas apostas	37
7.2.3	Notificações	37
7.3	Área autenticada - Administrador	37
7.3.1	Eventos Desportivos	37
7.3.2	Registar evento desportivo	38
7.3.3	Apostas	38
7.4	Área autenticada - Bookie	38
7.4.1	Eventos Desportivos	38
7.4.2	Adicionar evento desportivo	39
7.4.3	Notificações	39
8	Conclusões e Sugestões	40

1 Introdução

Este projeto foi elaborado no âmbito da Unidade Curricular de Arquiteturas de Software, do 4^a ano do Mestrado Integrado em Engenharia Informática, com vista à implementação de uma plataforma de apostas - BetESS. Esta aplicação deverá suportar o registo de apostadores, e será gerida por um administrador, que deverá criar eventos, associando as equipas existentes com as odds respetivas. Os utilizadores registados podem aceder a uma lista de eventos disponíveis e realizar uma ou mais apostas, sendo que apenas é possível apostar num resultado (i.e., um resultado por evento). Aquando da realização da aposta, é necessário que o jogador indique a quantia a apostar e o resultado pretendido, para que o sistema possa manter uma lista das apostas realizadas por cada apostador. Esta aposta, que possui um estado conforme a sua condição, quando passar de aberta para fechada, notifica os apostadores do resultado da mesma, e se for o caso, do valor ganho em ESScoins conforme a odd.

Assim, neste relatório, pode-se constatar o trabalho desenvolvido pelo grupo referente a três fases: uma primeira sem qualquer padrão de arquitetura ou design, uma outra aplicando o padrão arquitetural MVC (Model-View-Controller), juntamente com o padrão de design Observer, e uma última onde o grupo viu-se obrigado a implementar um requisito lançado perto do prazo de entrega. Todas estas fases implicaram a tomada de decisões pelo grupo, que são explicadas ao longo do relatório.

Desta forma, o relatório começa por introduzir o problema e declarar os requisitos, funcionais, não-funcionais e de qualidade referentes ao sistema a implementar. De seguida, apresenta-se toda a modelação UML. Nesta secção são discutidas as duas versões, com e sem padrões de arquitetura ou design, ao nível do diagrama de classes. Depois, o grupo achou por bem introduzir uma secção para que fosse discutido todo o trabalho que o novo requisito implicou. Por fim, aparecem os mockups elaborados, assim como uma secção de implementação, onde é possível ver o resultado final da aplicação quanto à sua interface, bem como se se implementaram todos os requisitos finais do enunciado. O relatório termina com as conclusões, em que o grupo faz uma análise do trabalho desenvolvido ao longo do semestre, tendo em conta as dificuldades obtidas.

2 Problema

Pretende-se desenvolver um sistema que possibilite o registo de utilizadores, que indicam uma quantia monetária (ESScoins) disponível para aposta. Ao verificar uma lista de eventos, o apostador escolhe um evento "Aberto" com a respetiva odd (no formato equipa1 - equipa2(<odds respetivas>)), e decide apostar uma quantia, sendo que o sistema deverá manter uma lista de apostas por utilizador. Quando as apostas realizadas por um user passarem ao estado "Fechada" (assume-se que o resultado do evento é conhecido), o utilizador deverá ser notificado de tal ocorrência, juntamente com o seu ganho com a mesma. Para determinar o valor ganho numa aposta deverá ser definido para cada evento as odds para os possíveis resultados e sobre essas odds será calculado o valor ganho numa aposta. Exemplificando, no caso de vitória do SC Braga na aposta "FC Barcelona - SC Braga (1.38 , 4.40, 8.00):

1. os apostadores que tenham indicado a vitória do SC Braga recebem 8.00 ESScoins por cada ESScoin apostada;
2. os apostadores que tenham indicado a vitória do FC Barcelona ou o empate recebem 0.

É de realçar que **não existe o estado "A decorrer"** como em muitos sistemas reais, sendo que este, no sistema do grupo, é apenas ilusório. Não serão permitidas efetuar operações como atualizar odds ou cashout enquanto um evento decorre.

3 Requisitos

Por forma a restringir, priorizar e padronizar o que o sistema deveria fazer e fornecer, foram definidos um conjunto de requisitos. Estes requisitos, fulcrais ao projeto para descrever o funcionamento do sistema, obrigaram a várias reuniões entre o grupo.

Nas próximas secções apresentar-se-ão os requisitos de utilizador, bem como os respetivos de sistema, que se dividem, por sua vez, em funcionais e não funcionais.

3.1 Requisitos funcionais

Os requisitos devem ter diferentes níveis de detalhe, uma vez que têm, também, diferentes tipos de leitores, que os usam de maneiras diferentes. Assim, para cada requisito de utilizador, apresentam-se os de sistema que lhe correspondem, em formato tabular, de forma a facilitar a leitura. Como maneira a clarificar o raciocínio, dividem-se, também, os requisitos em temas: **requisitos correspondentes ao apostador, administrador, ao bookie, e partilhados entre dois ou mais atores**. Antes de cada requisito tem uma pequena descrição (items) relativa ao que este se refere.

- Apostador
 - Ver notificações

Definição do requisito de Utilizador
1. O utilizador deve poder consultar as suas notificações.
Especificação do(s) requisito(s) de Sistema
1.1. O sistema mostrará a lista completa de notificações do utilizador, mostrando o seu estado;
1.2. A lista de notificações será ordenada por ordem de criação;
1.3. O sistema deverá também permitir marcar notificações como "lidas", caso as hajam não lidas.

- Consultar apostas

Definição do requisito de Utilizador
2. O utilizador deve poder consultar as suas apostas, quer pendentes quer passadas.
Especificação do(s) requisito(s) de Sistema
2.1. O sistema mostrará a lista completa de apostas cujo evento desportivo ainda não foi realizado ou já terminou.

- Fazer aposta

Definição do requisito de Utilizador
3. O utilizador deve poder fazer uma aposta.
Especificação do(s) requisito(s) de Sistema
3.1. O sistema mostrará a lista de eventos desportivos a realizar;
3.2. O sistema deve permitir que se escolha um evento a apostar;
3.3. Deverá também permitir inserir a quantia que o utilizador pretende apostar no evento;
3.4. O sistema deve autorizar escolher 'Equipa casa', 'Equipa fora' e 'Empate' conforme a previsão do utilizador para cada equipa. 'Equipa casa' indica que o utilizador aposta na vitória da equipa não visitante, 'Equipa fora' na equipa visitante e 'Empate' no empate entre as duas equipas;
3.5. O sistema deverá permitir que seja registada uma proposta de uma aposta, sendo que mais tarde será atribuído um determinado resultado à mesma.

- Creditar quantia monetária

Definição do requisito de Utilizador
4. O utilizador deve poder carregar uma quantia monetária na plataforma, sendo que só depois de tal é que possuirá de ESScoins para apostar.
Especificação do(s) requisito(s) de Sistema
4.1. O sistema deverá permitir que seja elaborado um método de carregamento (online), que será sempre a favor da plataforma de apostas. Este método é utilizado quando o utilizador desejar, mas é necessário para a existência de ESScoins para realização de uma aposta.

- Efetuar cashout

Definição do requisito de Utilizador
5. O apostador poderá efetuar o cashout de uma aposta, mas apenas antes da mesma ser fechada. O dinheiro que apostou ser-lhe-á devolvido, à exceção de uma pequena penalização.
Especificação do(s) requisito(s) de Sistema
5.1. O sistema permitirá que sejam exibidas todas as apostas possíveis de efetuar cashout, juntamente com a opção de cashout;
5.2. Deverá também calcular a respectiva penalização e debitar o valor ao utilizador.

- Verificar e editar perfil

Definição do requisito de Utilizador
6. O apostador deve poder verificar o seu perfil, que deverá conter a informação pessoal inserida aquando do registo, assim como as listagens de apostas passadas, pendentes e ganhos. Também devem poder ser editados dados não fixos.
Especificação do(s) requisito(s) de Sistema
6.1. O sistema deverá fornecer o perfil de utilizador completo e intuitivo;
6.2. Deverá ser possível editar o Nome, Contacto telefónico, palavra-passe e email.

- Consultar ganhos

Definição do requisito de Utilizador
7. O apostador deverá ser capaz de consultar o seu saldo.
Especificação do(s) requisito(s) de Sistema
7.1. O sistema deve mostrar o saldo atual do utilizador. Este saldo não inclui os valores associados a ganhos de apostas de eventos ainda abertos.

- Registo

Definição do requisito de Utilizador
8. O utilizador deve poder registar-se no sistema.
Especificação do(s) requisito(s) de Sistema
8.1. O sistema deve solicitar email, username, nome, password e contacto;
8.2. O sistema não deve permitir o registo de utentes com um username/email já registado;
1.3. O sistema deve armazenar os dados.

- Administrador

- Criar liga

Definição do requisito de Utilizador
1. O utilizador poderá ser capaz de criar uma nova liga.
Especificação do(s) requisito(s) de Sistema
1.1. O sistema deve permitir criar uma nova liga, deixando que o utilizador a nomeie;
1.2. O sistema tem que manter registo de cada liga criada pelo utilizador.

- Remover evento desportivo

Definição do requisito de Utilizador
2. O utilizador deverá ser o responsável por remover os eventos desportivos do sistema, quando assim o entender.
Especificação do(s) requisito(s) de Sistema
2.1. O sistema deve listar todos os eventos desportivos que tem registados, de forma a que o utilizador possa escolher um;
2.2. Deve também fornecer a possibilidade de remoção do evento pretendido.

- Alterar estado de evento desportivo

Definição do requisito de Utilizador
3. O administrador deverá alterar o estado de um evento desportivo.
Especificação do(s) requisito(s) de Sistema
3.1. O sistema deve permitir seleccionar um evento da base de dados, e alterar o seu estado;
3.2. O estado de um evento pode ir de "Aberto" para "Fechado".
3.2. Depois de passar para "Fechado", o sistema efetua todas as alterações necessárias e regista apostas e notificações do evento.

- Ver lista de apostadores

Definição do requisito de Utilizador
4. O utilizador deverá poder acessar à lista de apostadores existentes no sistema.
Especificação do(s) requisito(s) de Sistema
4.1. O sistema deve mostrar a lista de todos os apostadores existentes no sistema, bem como os detalhes sobre os mesmos;
4.2. A lista deverá ser ordenada por identificador de apostador.

- Ver e adicionar apostadores à lista negra

Definição do requisito de Utilizador
5. O utilizador poderá ver apostadores da lista negra, bem como adicioná-los à mesma.
Especificação do(s) requisito(s) de Sistema
5.1. O sistema deve mostrar a lista de todos os apostadores que estejam na lista negra;
5.2. Deve também fornecer a opção de adicionar algum jogador a esta lista.

- Banir apostador

Definição do requisito de Utilizador
6. O administrador deverá poder banir apostadores.
Especificação do(s) requisito(s) de Sistema
6.1. O sistema deve mostrar a lista de todos os apostadores que estejam na lista negra;
6.2. A lista deverá ter a opção de "Banir" um apostador.

- Criar equipa

Definição do requisito de Utilizador
7. O utilizador poderá ser capaz de criar uma nova equipa
Especificação do(s) requisito(s) de Sistema
7.1. O sistema deve permitir criar uma nova equipa, deixando que o utilizador a nomeie, tal como que escolha a que liga (já existente) esta equipa pertence;
7.2. O sistema tem que manter registo de cada equipa criada pelo utilizador.

- Ver lista de apostas de apostadores

Definição do requisito de Utilizador
8. O administrador poderá ver a lista de todas as apostas realizadas por apostadores.
Especificação do(s) requisito(s) de Sistema
8.1. O sistema mostrará uma lista de todas as apostas realizadas por jogadores;
8.2. Esta lista terá os detalhes relativos à informação básica de uma aposta. A cada aposta é também vinculado o id do jogador a ela associado.

- Bookie

- Ver notificação de valor ganho em evento

Definição do requisito de Utilizador
1. O utilizador deve poder consultar as suas notificações de fim de evento.
Especificação do(s) requisito(s) de Sistema
1.1. O sistema mostrará a lista completa de notificações do utilizador;
1.2. Cada notificação deverá apresentar os ganhos relativos ao evento, perdas e o balanço;
1.3. A lista de notificações será ordenada por ordem de fecho de evento;
1.4. O sistema deverá também permitir descartar as notificações lidas, caso as hajam.

- Ativar notificações

Definição do requisito de Utilizador
2. O utilizador deve conseguir, aquando da criação de um evento, explicitar se quer ou não receber notificações quando este acabar.
Especificação do(s) requisito(s) de Sistema
2.1. O sistema deverá dispor ao utilizador a opção de indicar o desejo de notificações;
2.2. Caso esta vontade seja indicada, quando um evento for fechado, o utilizador deverá receber a respetiva notificação.

- Administrador e Bookie

- Adicionar evento desportivo (definindo odds)

Definição do requisito de Utilizador
1. O utilizador deve poder adicionar um evento.
Especificação do(s) requisito(s) de Sistema
1.1. O sistema disponibilizará uma opção de adição de evento.
1.2. O sistema deve permitir que se escolham duas equipas para disputar o evento.
1.3. Deverá também permitir inserir as odds de casa, fora e empate.
1.4. Adicionalmente, se o registo de evento for registado por um Bookie, o sistema disponibilizará também a opção de registar o interesse em receber uma notificação (quando o mesmo acabar).
1.5. O sistema deve registar todos os eventos criados.

- Apostador, Administrador e Bookie

- Autenticação

Definição do requisito de Utilizador
1. O utilizador deve conseguir autenticar-se no sistema.
Especificação do(s) requisito(s) de Sistema
1.1. O sistema deve solicitar email/username e password para se autenticar.
1.2. O sistema deve verificar a validade dos dados inseridos e associar os dados a um perfil de utilizador, de modo a reconhecê-lo.

- Ver lista de eventos desportivos

Definição do requisito de Utilizador
2. O utilizador deve poder ver as listas de eventos desportivos disponíveis.
Especificação do(s) requisito(s) de Sistema
2.1. O sistema deve disponibilizar uma lista de todos os eventos desportivos possíveis de realizar uma aposta (eventos abertos).

3.2 Requisitos não-funcionais

Em relação à parte técnica e às características internas do programa, enunciam-se os RNF que se subdividem, por decisão do grupo, em **Organizacionais e Culturais**, **Usabilidade**, **Manutenção e suporte**, **Desempenho**, **Operacionais** e **Segurança**. Esta decisão baseou-se, apenas, no que o grupo considerou mais adequado detalhar.

(Em alguns requisitos, utilizou-se uma notação [subtipo] para melhor identificar o subtipo do requisito não funcional.)

Tipo de Requisito	Motivação
Organizacionais e Culturais	O produto deve usar português de Portugal.
	O utilizador deve autenticar-se com username e password respectiva.
Usabilidade	O produto deve ser intuitivo de usar para adultos (idade >18).
	O produto deve ser de fácil uso a partir dos 5 minutos de utilização. O nível de satisfação dos utilizadores com a aplicação deverá ser elevado.
Manutenção e Suporte	O código deverá ser de fácil manutenção. [MODIFICABILIDADE]
	O programa do produto de software deve conter comentários. O código deverá permitir a fácil adição de novos componentes sem que seja necessário uma reestruturação da arquitetura.
Desempenho	O sistema guarda e efectua cálculos precisos, sendo que mostra apenas os resultados até às centésimas de euros. [PRECISÃO]
Operacionais	O produto deve carregar os dados a partir de um objeto binário, e exportá-los para o mesmo formato, por forma a garantir a persistência dos dados.
Segurança	Os dados intrínsecos a um cliente devem ser fornecidos apenas ao próprio e ao seu único superior, o administrador.
	O produto deve rejeitar a introdução de dados incorretos.

3.3 Especificação de requisitos de qualidade

De seguida apresentam-se as especificações dos requisitos de qualidade no que toca à extensibilidade e manutenção do sistema.



Figura 1: Especificação do requisito de qualidade extensibilidade.



Figura 2: Especificação do requisito de qualidade referente à manutenção do sistema.

4 Modelação UML

4.1 Modelo de Domínio

O modelo de domínio analisa o problema de uma perspectiva concetual e é uma aproximação da representação gráfica das classes do programa e dos seus atributos, assim como o relacionamento entre estas. O modelo de domínio gerado a partir dos requisitos propostos é o mostrado abaixo. As principais entidades identificadas são Jogador, Evento Desportivo, e Apostas, o que é explicado pelo principal foco da aplicação, a realização de apostas, envolverem estas principalmente. As restantes entidades completam o esquema, uma vez que uma aposta tem de estar associada a um valor e resultado pretendido, por exemplo. A BetESS gere a maioria das entidades.

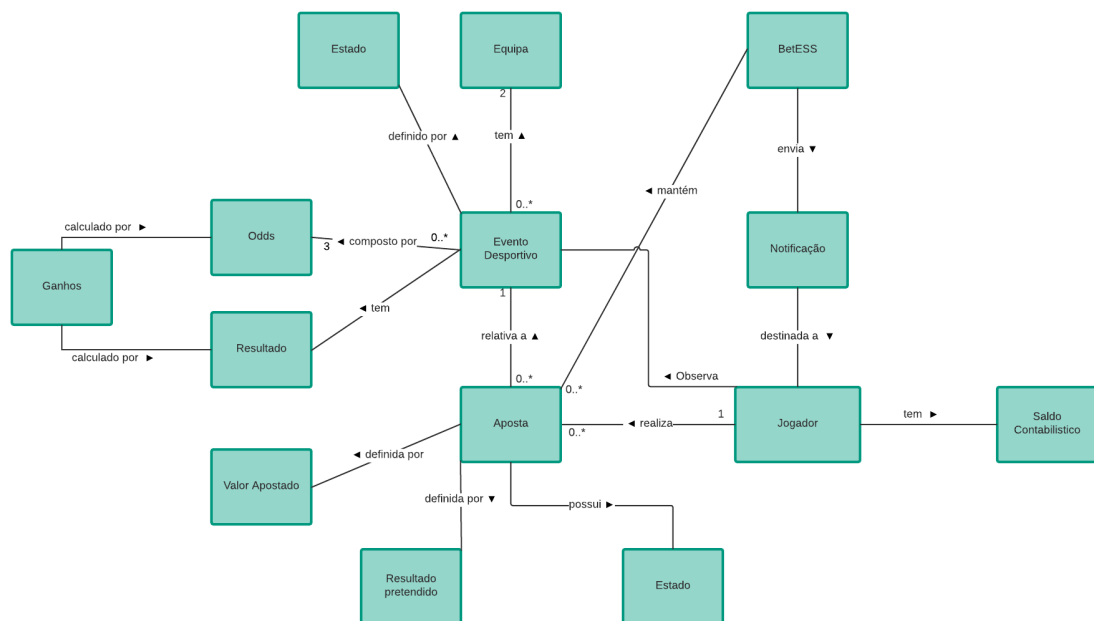


Figura 3: Modelo de Domínio

4.2 Diagrama de Use Cases

Uma vez validados os requisitos anteriores, estes fornecerão a base para a criação do diagrama de use cases, que poderia servir como auxílio à comunicação entre o cliente (que, neste caso, é a equipa docente) e o grupo. Assim, de maneira a delimitar bem as fronteiras e funcionalidades do sistema do ponto de vista do cliente, e para saber, posteriormente, o que teria de ser implementado, foram identificados os atores, use cases e associações do presente sistema de apostas. Pode ver-se, de seguida, a existência de três atores – Apostador, Administrador e Bookie - que exercem tarefas e têm funções distintas.

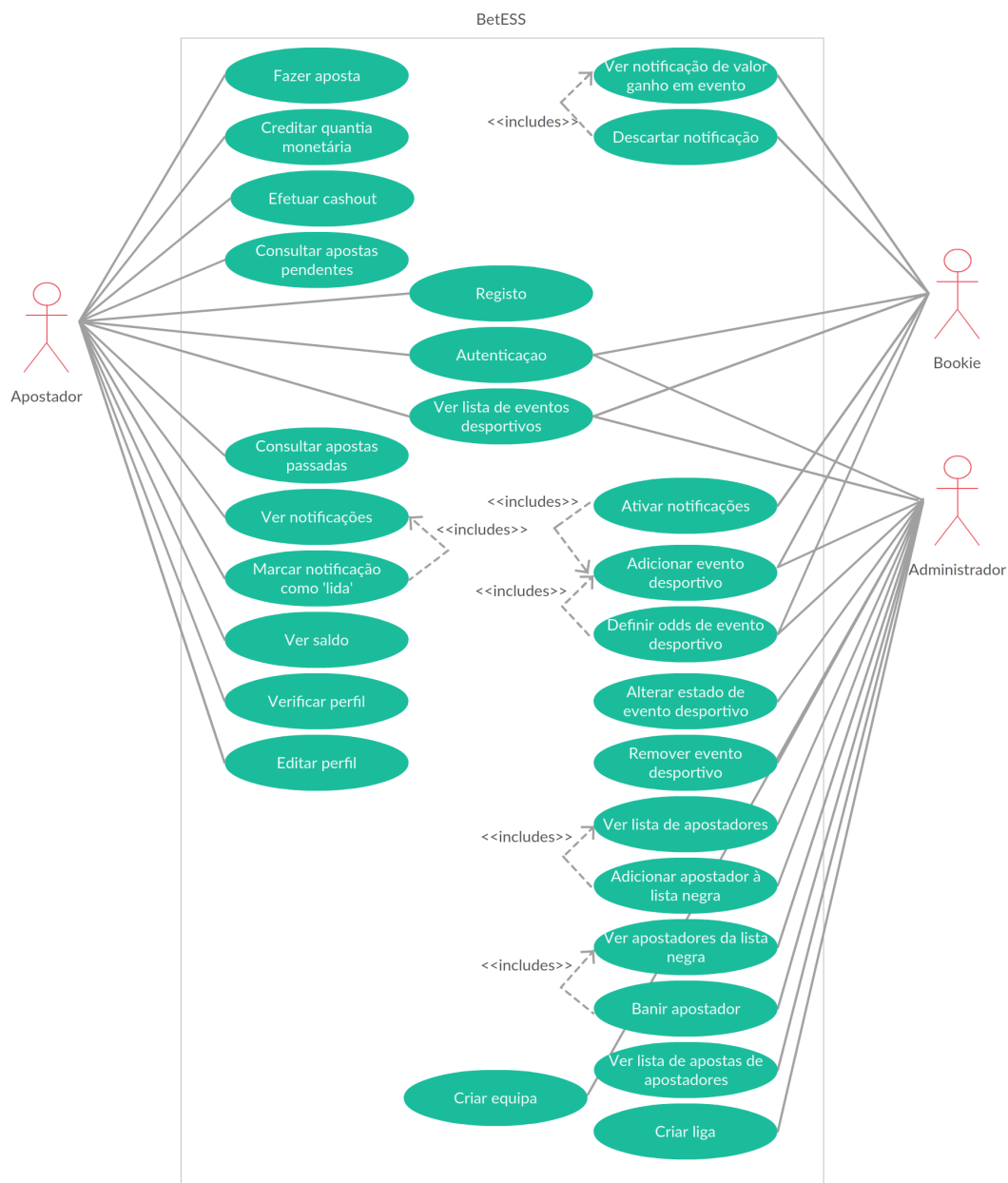


Figura 4: Diagrama de Use Cases

4.3 Diagramas de Sequência de Sistema

Por forma a melhor descrever o comportamento do sistema, principalmente em relação ao utilizador, desenvolveram-se alguns diagramas de sequência. Estes, que aprofundam os Use Cases especificados, descrevem as funcionalidades do sistema a encontrar a cada passo. Este tipo de diagrama, pela sua tipologia de interação Utilizador-Sistema, permite a aproximação e perceção das necessidades relativas à fase de desenvolvimento. É assim tornado claro para a equipa de desenvolvimento os componentes e funcionalidades a desenvolver.

4.3.1 Fazer aposta

Em sequência com o Use Case fornecido anteriormente, dado pelo nome de “Fazer aposta”, é apresentado, de seguida, o seu diagrama de sequência. Neste diagrama é possível verificar que o ator corresponde ao apostador, que usará a aplicação. Para realizar uma aposta, o apostador deverá seleccionar o evento desportivo em que quer apostar, bem como inserir a quantia monetária e o resultado que o ator confia ser o que pode vir a acontecer. Adicionalmente, o sistema calcula os ganhos possíveis com a aposta, de forma ao apostador melhor ponderar a aposta. Se o apostador concordar e prosseguir com a realização da aposta, são confirmados os dados inseridos, já que podem surgir duas situações de erro, que resultam no término da interação. Neste momento, caso tudo tenha corrido bem, será registada a aposta e indicado o sucesso. Caso o ator não tenha prosseguido com a aposta, esta não é registada.

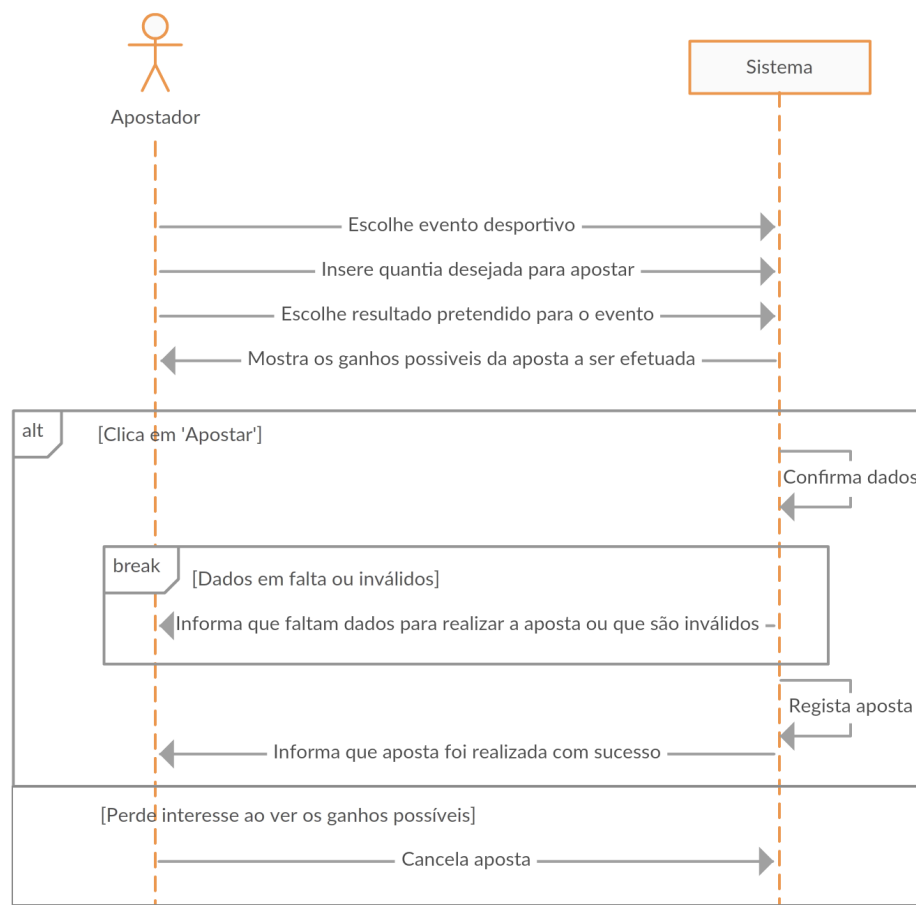


Figura 5: Diagrama de Sequência de Sistema - Fazer aposta

4.3.2 Adicionar evento desportivo

Para o Use Case fornecido anteriormente, dado pelo nome de “Adicionar evento desportivo”, é apresentado, de seguida, o seu diagrama de sequência. Esta tarefa cabe ao bookie (que pode escolher receber notificações futuras do resultado do evento), mas também é possível ser feita pelo administrador. Desta forma, é necessário escolher as respectivas equipas adversárias, identificando a que joga em casa e fora. Depois, escolhem-se as odds para a casa, fora e empate. No momento do registo do evento, o sistema confirma os dados inseridos, para o caso de estarem em falta ou serem inválidos. Se isto se verificar, o evento não será adicionado até que o ator corrija os dados. Caso contrário, o evento é registado.

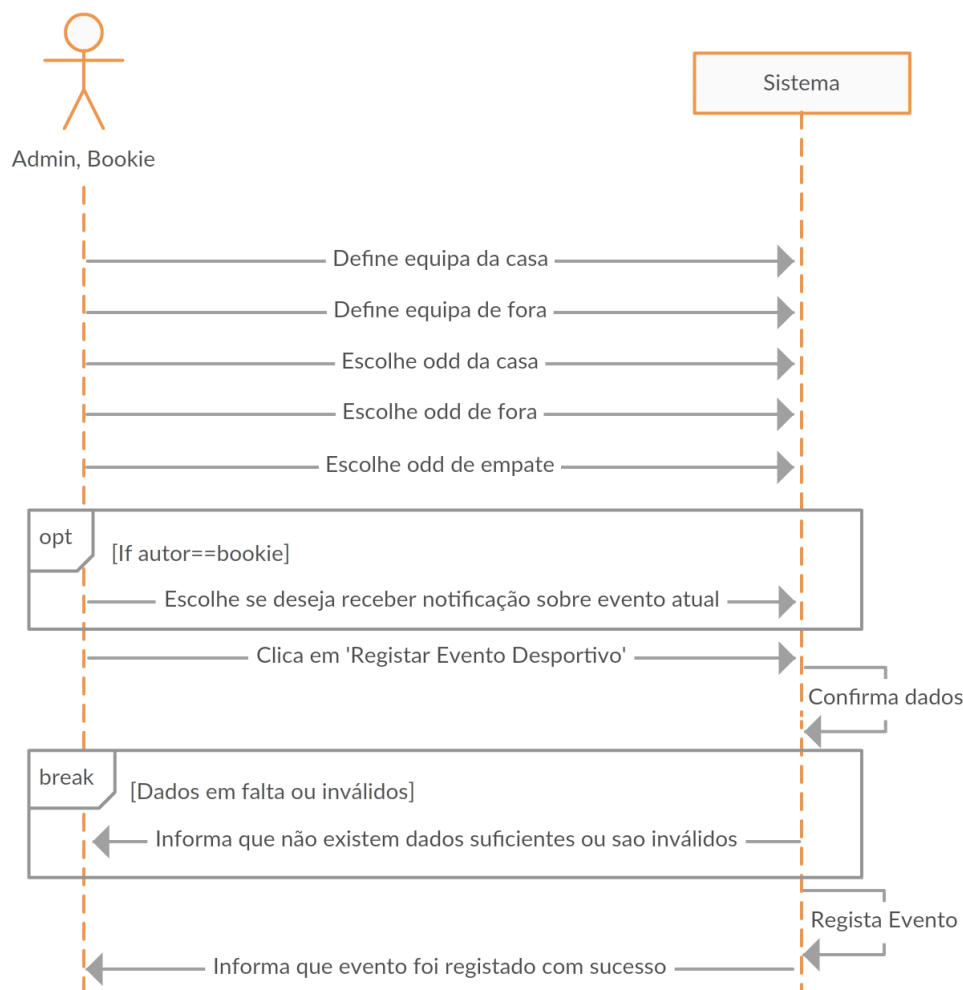


Figura 6: Diagrama de Sequência de Sistema - Adicionar evento desportivo

4.3.3 Alterar estado de evento desportivo

Por fim, para o Use Case fornecido anteriormente “Alterar estado de evento desportivo”, é apresentado, de seguida, o seu diagrama de sequência. Uma mudança de estado, no sistema em questão, correspondem ao fecho do evento, com a consequente declaração de resultados. Assim, neste diagrama é possível verificar que o ator corresponde ao administrador. Para alterar o estado de um evento desportivo, este ator precisa, basicamente, de indicar a equipa vencedora do evento. Consequentemente, o sistema verifica se o administrador escolheu a equipa vencedora, e em caso afirmativo, regista as alterações. Com o fecho de um evento, o apostador irá receber uma notificação, tal como o bookie se assim o tiver declarado. Por este motivo, ao fechar o evento, são também registadas alterações a este nível.

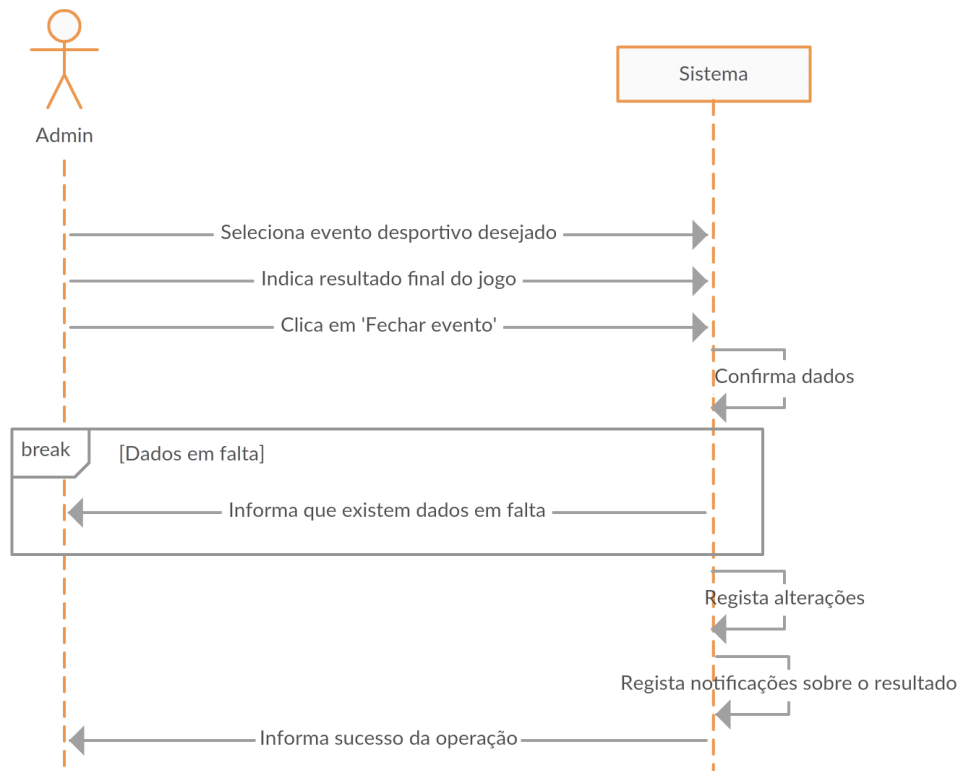


Figura 7: Diagrama de Sequência de Sistema - Alterar estado de evento desportivo

4.4 Diagrama de Instalação

O Diagrama de instalação descreve os componentes de hardware e software representando, assim, a configuração e a arquitetura da BetESS. A simplicidade do diagrama deve-se à simplicidade da solução do grupo.

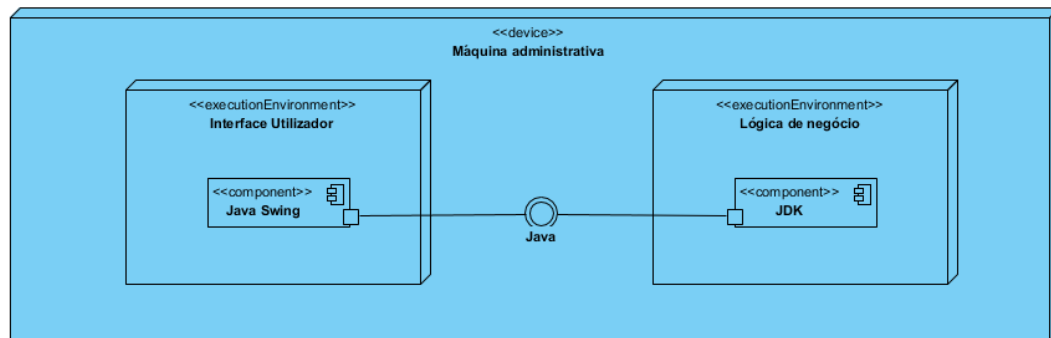


Figura 8: Diagrama de instalação

4.5 Diagrama de Classes - versão 1

4.5.1 Breve explicação da solução

Nesta versão, não foram utilizados quaisquer design patterns pelo grupo, como era requerido. Quanto aos architecture patterns foi utilizado uma espécie de MVC, padrão quase inevitável para este tipo de problema, mas não está implementado de acordo com as regras típicas desta arquitetura, podendo ser bastante melhorado e os dados isolados. Assim, como é visível no esquema abaixo, não há qualquer relação direta entre o Model e a View. Imaginando um fluxo de informação, um dado inserido pelo utilizador, passa pelo controlador e é armazenado no model (database). Quando é necessário efetuar alguma transformação nos dados e enviar de volta para a view, para ver de imediato uma alteração, essa transformação é feita no controlador, dando apenas a ilusão de um mvc onde a informação e a lógica do programa proviriam do model. Exemplificando, quando um administrador fecha uma aposta (aposta passa do estado aberta para fechada), o cálculo do pagamento das apostas referentes a cada jogador é feito no controlador e, posteriormente, armazenado. Este cálculo deveria ser efectuado no model, e depois a view acedia aos valores pretendidos.

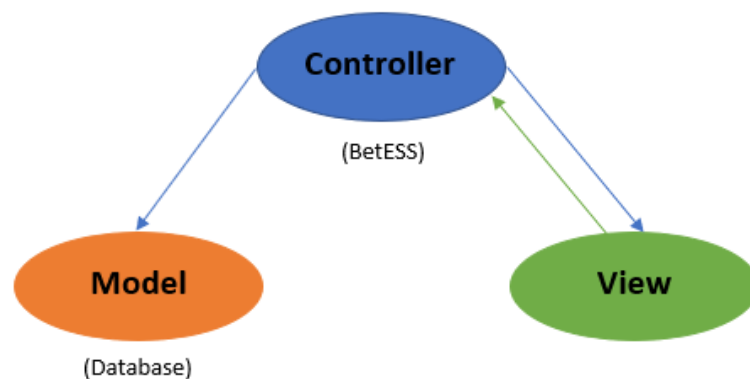


Figura 9: MVC da primeira fase do projeto.

4.5.2 Diagrama

Apresenta-se, então, o diagrama de classes do grupo para a primeira versão do trabalho. É visível a classe Database (model) com toda a lógica do programa, e com várias listas e maps necessários à manutenção de jogadores, apostas, eventos desportivos, entre outros. A BetESS encontra-se mais isolada, servindo de ponte entre a Database e as views.

(As classes relativas à interface foram omitidas, uma vez não ser usual considerar a sua organização neste tipo de diagrama).

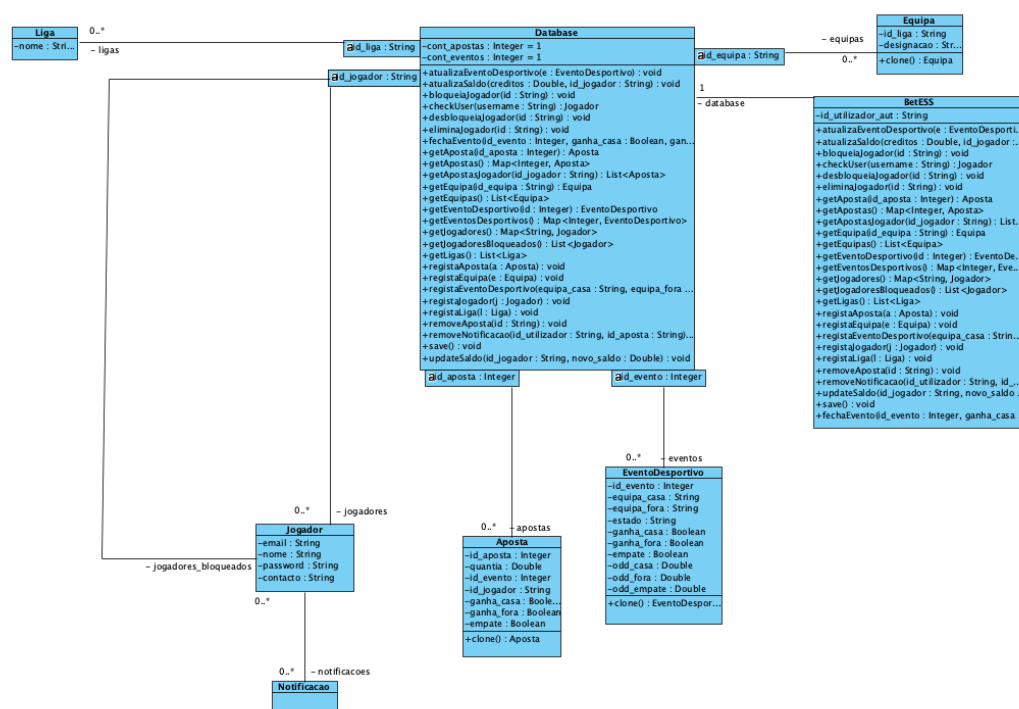


Figura 10: Diagrama de classes.

4.6 Diagrama de Classes - versão 2

4.6.1 Breve explicação da solução

Para a segunda fase do projeto, o grupo decidiu utilizar o design pattern **Observer** e o architecture pattern **MVC** - Model, View, Controller. Assim, para justificar a decisão de cada um destes padrões, segue-se uma lista das vantagens de cada um, adequadas ao problema proposto pela equipa docente.

- Vantagens do padrão arquitetural MVC, no caso do projeto atual:
 - **Permite múltiplas views:** uma vez que a View está separada do Model e não há dependência direta do Model para a View;
 - **Adaptação a mudanças:** os requisitos da interface têm tendência a mudar mais rapidamente do que os da camada de negócio. Como o Model não depende das Views, adicionar novas Views geralmente não afeta o Model;
 - **Possui re-usabilidade:** é possível criar bibliotecas e adicionar interfaces facilmente;
 - **Paralelização:** é possível desenvolver o sistema em paralelo, ou seja, pode-se começar o projeto por qualquer camada;
 - **Divisão de responsabilidades:** programadores na programação e web designers na construção visual do software, ou, neste caso, diferentes elementos do grupo em classes diferentes;
 - **Redução do esforço na manutenção do software:** as alterações são efetuadas separadamente não afetando as outras camadas do sistema.

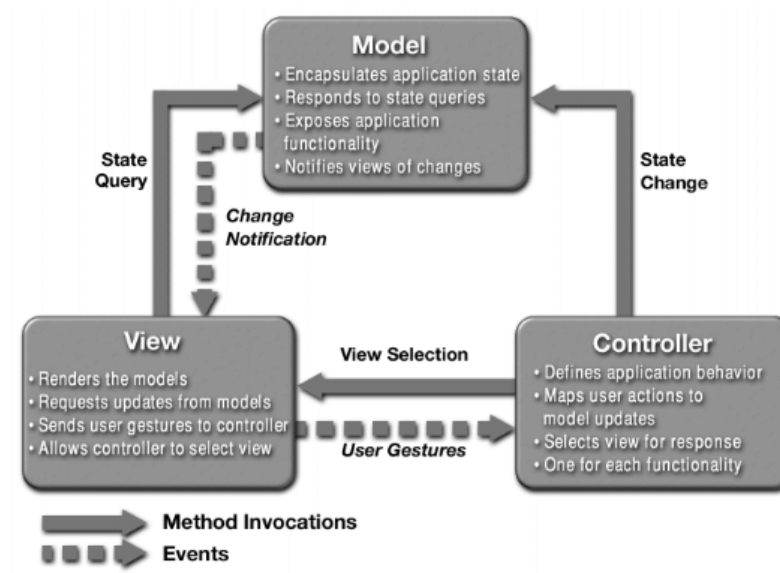


Figura 11: Esquema MVC utilizado pelo grupo na segunda fase

Desta forma, o grupo decidiu utilizar uma **abordagem do MVC centralizado no Model**, por achar que melhor encaixava no padrão Observer, explicado de seguida.

- Vantagens do padrão de design Observer, no caso do projeto atual:
 - Podem ser adicionados ou removidos Observers em qualquer altura, sem alteração do Subject;
 - Permite enviar dados para outros objetos eficientemente, sem qualquer mudança na classe Subject ou Observer
 - Suporta o princípio do acoplamento fraco entre objetos que interagem entre si.
 - O Padrão Observer define uma dependência um-para-muitos entre os objetos de modo que quando um objeto muda de estado, todos os seus dependentes são notificados e atualizados automaticamente

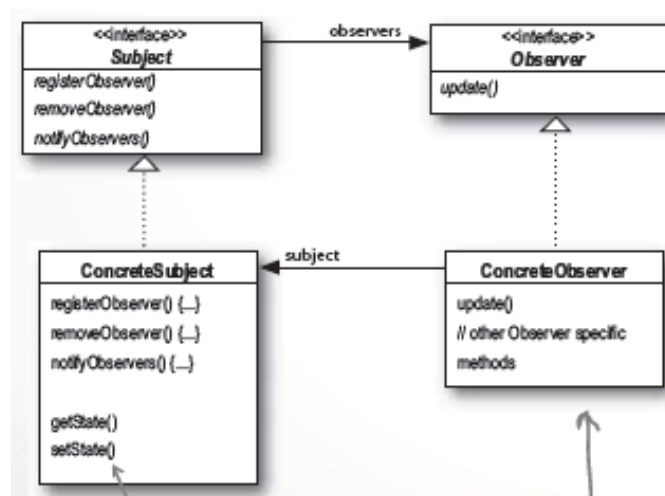


Figura 12: Diagrama de classes para Observer pattern

Embora **não fosse mandatário o controlador estar como Observer**, o grupo decidiu mantê-lo. Isto deve-se, acima de tudo, a uma questão de modificabilidade/adaptabilidade do código para algum requisito futuro, tornando o sistema extensível, e também para facilitar/simplificar o problema.

No que toca ao **método update()**, um dos mais importantes, de forma a obedecer aos requisitos não-funcionais e manter um sistema rápido, este foi desenvolvido para evitar que fosse trazido a totalidade dos dados do model.

Para terminar a contextualização das decisões do grupo, é importante notar que foi decidido que a implementação da base de dados seria através de lists e sets, uma vez que simplificaria o problema e não era objeto de avaliação principal.

4.6.2 Diagrama

Apresenta-se, então, o diagrama de classes do grupo para a segunda versão do trabalho. É visível a classe Model, com toda a lógica do programa, e com várias listas e maps necessárias à manutenção de jogadores, apostas, eventos desportivos, entre outros. O Model é o Concrete Subject que, como é observável, está ligado à interface Subject. O Model conhece os seus observadores para saber quem notificar (não envia informação!). Os Concrete Observers são as Views e o Controller, que estão ligados à interface Observer. Ambos o Controller e as Views observam as mudanças no model, e têm acesso à informação de mudança de estado.

(As classes relativas à interface não foram omitidas neste diagrama, uma vez que o grupo considerou relevante a sua organização para melhor observar os componentes do Observer pattern).

(Algumas setas de ligação foram simplificadas, de forma a evitar a complexidade e confusão do diagrama. Exemplificando, se o grupo fosse a elaborar todas as setas, todas as views teriam que estar ligadas com o model, o que se tornaria pesado. Deste modo, explicitaram-se as instâncias nestas classes, mas não se desenharam as setas.)

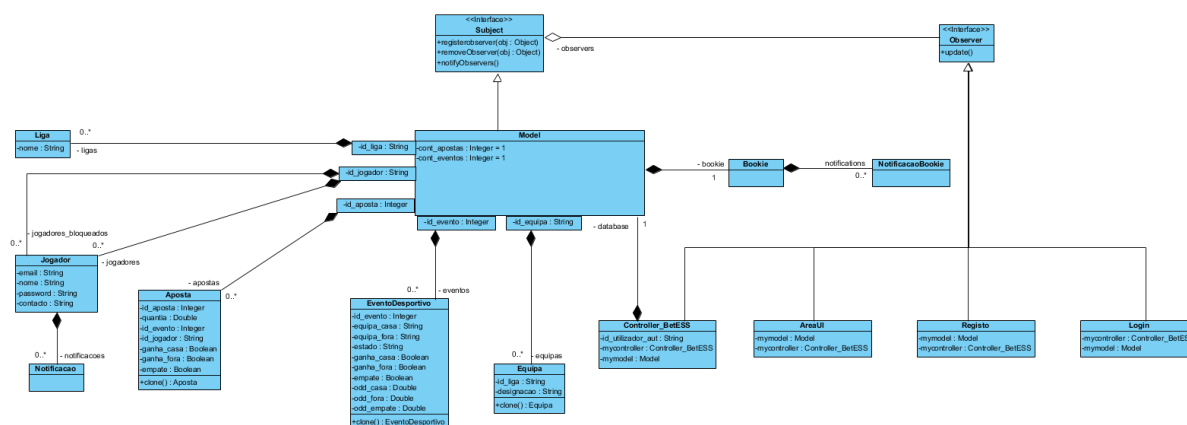


Figura 13: Diagrama de Classes com padrões

4.7 Comparação conclusiva e concreta entre as duas versões

Uma vez que já foram apresentadas tanto a versão sem padrões como com padrões, assim como as respectivas vantagens consideradas pelo grupo, sumariza-se agora quais as principais diferenças e conclusões entre estas duas versões.

Dado que, na primeira versão, o padrão *MVC* não foi implementado completamente, este foi um dos pontos a ter em atenção na implementação da versão com padrões. Deste modo, toda a lógica e dados passaram para a classe *Model*, tal como o padrão define (Modelo centralizado nos dados, escolhido pelo grupo).

No que toca à interface, grande parte do sistema, a escolha do padrão de desenho da solução teve como principal objetivo de otimizar o sistema também a este nível. Desta forma, nesta segunda versão, foi implementado então o padrão *Observer*. Deste modo, cada uma das *views* regista-se como observadora dos dados, o que possibilita à camada de negócio no momento da atualização de qualquer tipo de dados, notificar os seus observadores e, assim, manter a interface gráfica sempre atualizada e isolada.

Foram tomadas também algumas decisões de implementação no que diz respeito à notificação da interface e como esta atualiza os seus elementos (`update()`). Por forma a realizar este processo da forma mais eficiente possível, as notificações são identificadas por etiquetas identificativas do tipo de dados a atualizar, evitando assim que a interface carregue os dados na totalidade a cada notificação que recebe.

Para terminar, a lógica dos elementos constituintes da interface sofreu também alterações, sendo esta agora dinâmica. Dependendo do utilizador autenticado, são mostrados apenas os elementos referentes a esse utilizador, ao contrário do que acontecia na versão anterior, em que cada utilizador tinha definida uma *view* só sua. Assim, a gerência das notificações dirigidas à interface torna-se bastante mais simples. Esta última modificação não se deveu a nenhum padrão, mas sim ao que o grupo considerou mais adequado (especialmente com o surgimento de mais um utilizador, que implicaria mais uma *view*).

5 Introdução do novo requisito - mudanças

Com a introdução do novo requisito, uma vez que a solução do grupo encontra-se implementada seguindo o padrão arquitetural *MVC* e de design *Observer*, a integração deste novo requisito foi bastante fácil e rápida.

Uma vez que o novo interveniente no sistema (*Bookie*) exigia a definição de permissões específicas no que diz respeito ao acesso ao sistema, foi desenvolvida uma área de exploração dedicada a este novo utilizador. Para o desenvolvimento desta, apenas foi necessário alterar a classe *AreaUI.java* definindo um novo painel de elementos a que um *bookie* teria acesso, e conforme o utilizador logado, apenas seria necessário fazer a troca entre painéis. Foi também bem como modificado o método *update()*, permitindo assim que os novos elementos da interface fossem também devidamente atualizados para o novo utilizador. Por forma a que toda a informação relativa ao utilizador *bookie* fosse mantida entre sessões, foi adicionada à classe *Model.java* uma instância de *Bookie* bem como o envio das notificações necessárias à interface. O controlador sofreu apenas pequenas alterações, sendo necessário definir métodos de acesso aos dados deste novo user.

Finalmente, foram desenvolvidas as classes *Bookie.java* e *NotificacaoBookie.java*. Esta primeira, contém uma lista de objetos *NotificacaoBookie* por forma a que as notificações sejam apresentadas corretamente. Para além disso, é mantida também uma lista dos identificadores dos eventos marcados pelo *bookie* para receber notificações. Assim, no momento do fecho de um evento, esta lista será consultada e serão lançadas as notificações.

6 Mockups

Segue-se a prototipagem simples da interface, realizada na primeira fase do projeto.

6.1 Login/Registo

The mockup displays two adjacent form panels. The left panel, titled 'Área de Login', contains labels for 'Email:' and 'Password:' followed by input fields, and an 'OK' button at the bottom right. The right panel, titled 'Área de Registo', contains labels for 'Nome:', 'Email:', 'Password:', and 'Contacto:' followed by input fields, and a dark 'Submit' button at the bottom right.

Área de Login	Área de Registo
Email: <input type="text"/>	Nome: <input type="text"/>
Password: <input type="text"/>	Email: <input type="text"/>
	Password: <input type="text"/>
	Contacto: <input type="text"/>
OK	Submit

Figura 14: Área de Login e de Registo

6.2 Área autenticada - Jogador

6.2.1 Apostar

[Apostar](#) [Minhas Apostas](#) [Créditos](#) [Notificações](#) [Editar Perfil](#)

Evento	Casa	Fora	Odd Casa	Odd Fora	Odd Empate
Evento nº 1	FCPorto	SLBenfica	1.38	4.40	8.00
Evento nº 2	Watford	Man. City	9.25	1.28	5.85
Evento nº 3	Levante	Barcelona	8.00	1.28	5.65
Evento nº 4	Rio Ave	Sporting CP	4.40	1.80	3.70

☒ Casa
☐ Fora
☐ Empate

Quantia (€):

Ganhos possíveis (€):

Submit

Terminar sessão

Figura 15: Área de Jogador - Efectuar uma aposta

6.2.2 Minhas apostas

Apostar Minhas Apostas Créditos Notificações Editar Perfil							
Evento	Casa	Fora	Vitória Casa	Vitória Fora	Empate	Quantia	Estado
Evento nº 1	FC Porto	SL Benfica		<input checked="" type="checkbox"/>		2	Não paga
Evento nº 2	Watford	Man City	<input checked="" type="checkbox"/>			10	Não paga
Evento nº 3	Levante	Barcelona		<input checked="" type="checkbox"/>		5	Paga
Evento nº 4	Rio Ave	Sporting CP			<input checked="" type="checkbox"/>	8	Paga

Cashout

Terminar sessão

Figura 16: Área de Jogador - Lista de minhas apostas

6.2.3 Créditos

Apostar Minhas Apostas Créditos Notificações Editar Perfil				
Saldo (€):	<input type="text" value="85.2"/>	<div>Creditar</div>		
Creditar (€):	<input type="text"/>			

Terminar sessão

Figura 17: Área de Jogador - Saldo e Creditar quantia

6.2.4 Notificações

Apostar

Minhas Apostas

Créditos

Notificações

Editar Perfil

Evento	Aposta	Equipa Casa	Equipa Fora	Quantia Apostada	Balanco	Estado
Evento nº 1	Aposta nº1	FCPorto	SLBenfica	2	6	Lida
Evento nº 2	Aposta nº2	Watford	Man. City	5	12	Lida
Evento nº 3	Aposta nº3	Levante	Barcelona	10	0	Lida
Evento nº 4	Aposta nº4	Rio Ave	Sporting CP	5	10	Não Lida

Descartar

Terminar sessão

Figura 18: Área de Jogador - Lista de notificações

6.2.5 Editar perfil

[Apostar](#) [Minhas Apostas](#) [Créditos](#) [Notificações](#) [Editar Perfil](#)

Nome:

ina_45@email.com

Email:

919191919

Password:

Contacto:

Inácio Santos

Submit

Terminar sessão

Figura 19: Área de Jogador - Editar perfil

6.3 Área autenticada - Administrador

6.3.1 Ver jogadores

Ver Jogadores

Eventos Desportivos

Jogadores Bloqueados

Apostas

Nova Equipa

Nova Liga

Email	Nome	Contacto	Saldo
ines_123@email.com	Inês Rodrigues	919191919	150.00
joelAndrade2@email.com	Joel Andrade	929292929	15.90
a_1997@email.com	André Campos	939393939	209.01
fcp_the_best@email.com	Ivo Costa	969696969	1055.50

Bloquear

Remover

Terminar sessão

Figura 20: Área de Administrador - Gestão de jogadores

6.3.2 Eventos Desportivos

Evento	Casa	Fora	Vitória Casa	Vitória Fora	Empate	Estado
Evento nº 1	FCPorto	SLBenfica		<input checked="" type="checkbox"/>		Terminada
Evento nº 2	Watford	Man. City	<input checked="" type="checkbox"/>			Terminado
Evento nº 3	Levante	Barcelona				A decorrer
Evento nº 4	Rio Ave	Sporting CP				A decorrer

Registrar EventoFechar Evento

Terminar sessão

Figura 21: Área de Administrador - Gestão de eventos desportivos

6.3.3 Registrar evento desportivo

Novo evento desportivo

Equipa fora:

Equipa casa:

Odd casa:

Odd fora:

Odd empate:

Registrar evento

Figura 22: Área de Administrador - adicionar novo evento

6.3.4 Jogadores Bloqueados

Ver Jogadores

Eventos Desportivos

Jogadores Bloqueados

Apostas

Nova Equipa

Nova Liga

Email	Nome	Contacto	Saldo
ines_123@email.com	Inês Rodrigues	919191919	150.00
joelAndrade2@email.com	Joel Andrade	929292929	15.90
a_1997@email.com	André Campos	939393939	209.01
fcp_the_best@email.com	Ivo Costa	969696969	1055.50

Desbloquear Jogador

Terminar sessão

Figura 23: Área de Administrador - Gestão de jogadores bloqueados

6.3.5 Apostas

Aposta	Evento	Jogador	Vitória Casa	Vitória Fora	Empate	Quantia
Aposta nº 1	Evento nº1	Jogador1		<input checked="" type="checkbox"/>		5
Aposta nº 2	Evento nº2	Jogador2	<input checked="" type="checkbox"/>			15
Aposta nº 3	Evento nº3	Jogador3		<input checked="" type="checkbox"/>		30
Aposta nº 4	Evento nº4	Jogador4			<input checked="" type="checkbox"/>	10

Eliminar

Terminar sessão

Figura 24: Área de Administrador - Gestão de apostas

6.3.6 Nova Equipa

Ver Jogadores	Eventos Desportivos	Jogadores Bloqueados	Apostas	Nova Equipa	Nova Liga
Nome Equipa: <input type="text"/>					
Liga: <input type="text"/>					
<div>Submit</div>					
<div>Terminar sessão</div>					

Figura 25: Área de Administrador - Adicionar nova equipa

6.3.7 Nova Liga

The form is part of a web application with a navigation bar at the top containing links: Ver Jogadores, Eventos Desportivos, Jogadores Bloqueados, Apostas, Nova Equipa, and Nova Liga. The main form area has a label 'Nome Liga:' followed by a text input field. Below the input field is a 'Submit' button. In the bottom right corner of the form area is a 'Terminar sessão' button.

Figura 26: Área de Administrador - Adicionar nova liga

6.4 Área autenticada - Bookie

6.4.1 Eventos Desportivos

The form is part of a web application with a navigation bar at the top containing links: Eventos Desportivos and Notificações. The main form area contains a table with the following data:

Evento	Equipa Casa	Equipa Fora	Ganha Casa	Ganha Fora	Empate	Estado
Evento nº1	FCPorto	SLBenfica	<input checked="" type="checkbox"/>			Terminado
Evento nº2	Watford	Man. City		<input checked="" type="checkbox"/>		Terminado
Evento nº3	Levante	Barcelona				A decorrer

Below the table is a button labeled 'Adicionar evento desportivo'. In the bottom right corner of the form area is a 'Terminar sessão' button.

Figura 27: Área do Bookie - Gestão de eventos desportivos

6.4.2 Adicionar evento desportivo

Novo evento desportivo

Equipa fora:

Equipa casa:

Odd casa:

Odd fora:

Odd empate:

☒ Notificações

Registrar evento

Figura 28: Área do Bookie - adicionar novo evento

6.4.3 Notificações

Eventos Desportivos		Notificações	
Evento	Ganhos	Perdas	Balanço
Evento nº1	1000	500	-500
Evento nº2	6000	0	-6000
Evento nº3	200	500	+300

Descartar

Terminar sessão

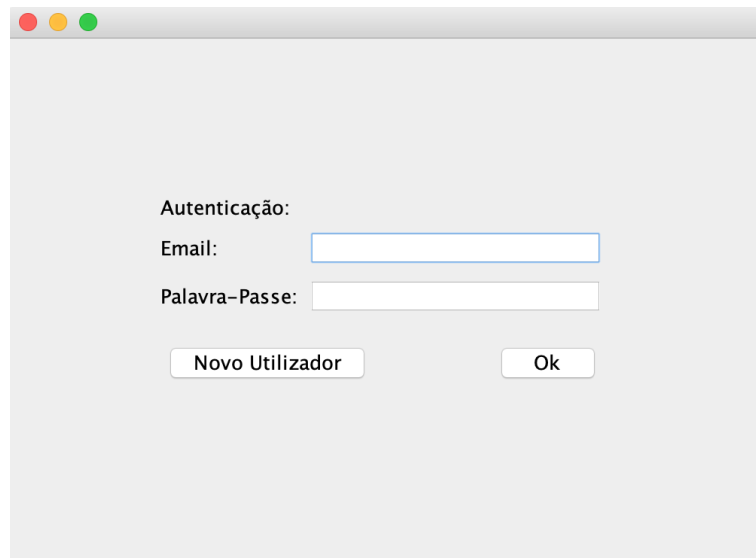
Figura 29: Área do Bookie - Gestão de notificações

7 Implementação

7.1 Interface Gráfica

Apresentam-se, de seguida, as diversas páginas da interface implementada pelo grupo.

7.1.1 Login/Registo



A screenshot of a web application window titled 'Autenticação:'. It contains two input fields: 'Email:' and 'Palavra-Passe:'. Below these fields are two buttons: 'Novo Utilizador' and 'Ok'.

Figura 30: Área de Login e de Registo

7.2 Área autenticada - Jogador

7.2.1 Apostar



A screenshot of a web application window titled 'Área autenticada'. It features a sidebar menu with buttons: 'Apostar', 'Minhas apostas', 'Créditos', 'Notificações', and 'Editar Perfil'. The main content area displays a table of betting events and a form for placing a bet.

Evento número	Casa	Fora	Odd Casa	Odd Fora	Odd Empate
1	FC Porto	SC Braga	1.7	1.2	2.1
2	SC Braga	FC Porto	1.2	1.7	2.1

Below the table, there is a form with the following elements:

- Quantia: € (euros)
- Ganhos possíveis:
- ☐ Casa ☐ Fora ☐ Empate
-
-

Figura 31: Área de Jogador - Efectuar uma aposta

7.2.2 Minhas apostas



Figura 32: Área de Jogador - Lista de minhas apostas

7.2.3 Notificações

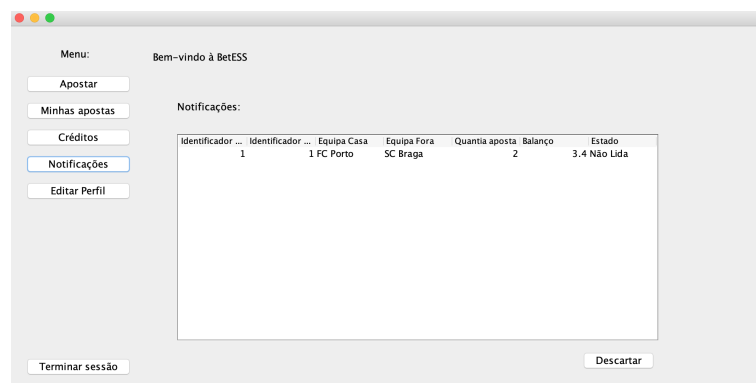


Figura 33: Área de Jogador - Lista de notificações

7.3 Área autenticada - Administrador

7.3.1 Eventos Desportivos

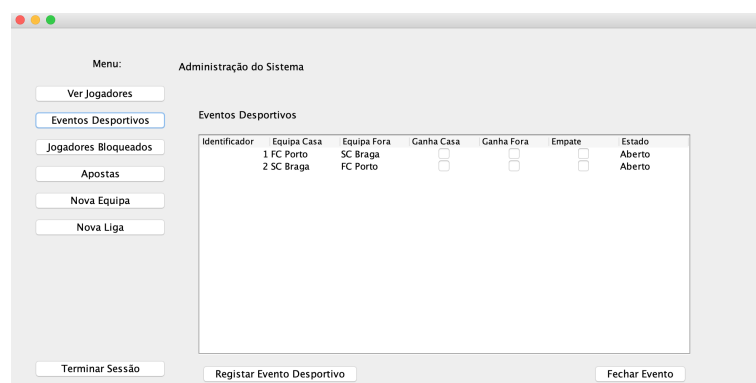


Figura 34: Área de Administrador - Gestão de eventos desportivos

7.3.2 Registrar evento desportivo

Menu: Administração do Sistema

Ver Jogadores
Eventos Desportivos
Jogadores Bloqueados
Apostas
Nova Equipa
Nova Liga

Terminar Sessão

Novo Evento Desportivo

Equipa Casa: FC Porto
Equipa Fora: SC Braga
Odd Casa: 1.7
Odd Fora: 1.2
Odd Empate: 2.1

Registrar Evento

Figura 35: Área de Administrador - adicionar novo evento

7.3.3 Apostas

Menu: Administração do Sistema

Ver Jogadores
Eventos Desportivos
Jogadores Bloqueados
Apostas
Nova Equipa
Nova Liga

Terminar Sessão

Apostas

Identificador	Identificador ...	Identificador J...	Ganha Casa	Ganha Fora	Empate	Quantia
1	1	marant_silva...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2

Eliminar

Figura 36: Área de Administrador - Gestão de apostas

7.4 Área autenticada - Bookie

7.4.1 Eventos Desportivos

Menu: Bem vindo à BetESS, Bookie!

Eventos Desportivos
Notificações

Terminar Sessão

Registrar Evento Desportivo

Eventos Desportivos

Identificador	Equipa Casa	Equipa Fora	Ganha Casa	Ganha Fora	Empate	Estado
1	FC Porto	SC Braga	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Terminado
2	SC Braga	FC Porto	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Terminado

Figura 37: Área do Bookie - Gestão de eventos desportivos

7.4.2 Adicionar evento desportivo

The screenshot shows a web application window titled 'Menu: Bem vindo à BetESS, Bookie!'. On the left, there is a sidebar with two buttons: 'Eventos Desportivos' and 'Notificações'. The main area is titled 'Novo Evento Desportivo'. It contains several input fields: 'Equipa Casa:' with a dropdown menu showing 'FC Porto', 'Equipa Fora:' with a dropdown menu showing 'SC Braga', 'Odd Casa:', 'Odd Fora:', and 'Odd Empate:'. There is also a checkbox labeled 'Notificações'. At the bottom left is a button 'Terminar Sessão' and at the bottom right is a button 'Registrar Evento'.

Figura 38: Área do Bookie - adicionar novo evento

7.4.3 Notificações

The screenshot shows the same web application window, but the 'Notificações' button in the sidebar is selected. The main area is titled 'Notificações:'. It contains a table with the following data:

Identificador Evento	Ganhos	Perdas	Balanco
3	0	-8.8	-8.8

At the bottom left is a button 'Terminar Sessão' and at the bottom right is a button 'Descartar'.

Figura 39: Área do Bookie - Gestão de notificações

8 Conclusões e Sugestões

A resolução trabalho prático foi bastante importante e enriquecedora, pois permitiu aos membros do grupo perceber e interiorizar melhor os conceitos abordados nas aulas da Unidade Curricular de Arquiteturas de Software.

Deste modo, foram aprofundados conhecimentos relativos a modelação UML e engenharia de requisitos, com a construção de todo o problema inicial, antes da implementação. O grupo alcançou uma maior consciência acerca da importância da divisão e boa escrita de requisitos, de forma a garantir as necessidades dos utilizadores e as restrições que são colocadas a um sistema. A modelação do problema, sendo um ingrediente essencial em todas as disciplinas da engenharia, certamente será útil para o posterior desenvolvimento dos conteúdos da UC e da vida profissional, uma vez que ajuda na formalização, simplificação e abstração de qualquer problema.

No que diz respeito ao maior objetivo do trabalho prático, entender a diferença entre o uso ou não de padrões de design e arquitetura, o grupo sente que foi onde mais evoluiu, apesar de alguma dificuldade inicial e necessidade de planear e entender cada padrão e as respetivas vantagens para cada problema. A partir do momento que se entendeu a capacidade de reutilização, manutenção e extensibilidade dos design patterns, e a relação organizada entre elementos que um architecture pattern fornece, tornou-se cada vez mais fácil executar a segunda fase do projeto. Assim, através da correta divisão de tarefas e revisão por toda a equipa, foi conseguida uma solução simples e eficaz.

Quanto à inserção do novo requisito perto do prazo de entrega, o grupo conseguiu sentir na pele a "pressão" e o esforço que um novo requisito de um novo cliente pode trazer a uma equipa de desenvolvimento. O grupo também reparou que, graças à eficiência da solução com padrões, as novas alterações provenientes do requisito foram simplificadas, constatando-se assim a importância da sua aplicação.

Em suma, é feita uma apreciação positiva relativamente ao trabalho realizado, visto que a implementação de todas as funcionalidades propostas foram conseguidas com sucesso. O grupo conseguiu tirar partido dos conhecimentos adquiridos neste projeto, sentido-se capaz de, num contexto futuro, aplicar os conceitos subjacentes de forma eficaz. É evidente que, num outro contexto (como um projeto de grandes dimensões), seria benéfico que fossem implementadas um maior conjunto de funcionalidades adicionais para uma melhor gestão do sistema.