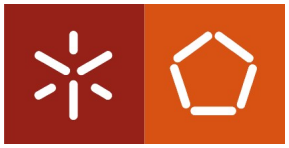

Universidade do Minho
Mestrado Integrado em Engenharia Informática
Departamento de Informática

Engenharia Web

BetESS Web Platform

Daniel Fernandes Veiga Maia A77531
Vitor Emanuel Carvalho Peixoto A79175

Ano Letivo 2018/2019

Resumo

Este relatório especifica as decisões tomadas no decorrer da primeira parte do desenvolvimento deste projeto. Esta primeira parte é composta também por diversas fases.

Numa primeira fase, foi feita uma análise dos requisitos propostos para a aplicação, sendo estes requisitos transpostos num modelo de domínio do sistema a desenvolver. Após uma verificação minuciosa da capacidade do modelo de domínio em responder aos problemas propostos, procedeu-se à sua conversão para uma base de dados relacional.

A segunda fase do projeto, consistiu na modelação do fluxo de interações do sistema. Para tal, foi utilizada a linguagem de modelação IFML (*Interaction Flow Modeling Language*). A modelação do sistema consistiu na criação de diversas vistas (*site views*) que correspondessem às diferentes funcionalidades autorizadas a cada ator do sistema. Dentro de cada vista foram criadas as diversas páginas e componentes necessários ao funcionamento da plataforma de apostas de acordo com os requisitos propostos.

Conteúdo

1 Introdução

Temos assistido a um crescente aumento do custo de desenvolvimento de *software*, com destaque para a sua componente *front-end*, maioritariamente causado por uma inadequada abordagem ao processo de desenvolvimento. A escalabilidade, portabilidade e modificabilidade que é pedida aos aplicativos de hoje em dia, tem de ser suportada por um desenvolvimento de modelos abstratos à linguagem a ser utilizada, baseados nas interações entre utilizadores e o sistema.

Neste capítulo, a linguagem de modelação IFML traz inúmeros benefícios ao processo de desenvolvimento das componentes *front-end* de um sistema, uma vez que suporta a especificação do *front-end* de diferentes perspetivas, permitindo encontrar falhas e corrigi-las numa fase precoce do desenvolvimento, reduzindo o custo da correção de erros e aumentando a escalabilidade e portabilidade da plataforma.

Seguindo esta linha, a ferramenta *WebRatio* permite modelar na linguagem IFML, num ambiente gráfico e simples, gerando automaticamente a aplicação através da compilação dos fluxos interativos modelados no sistema.

Esta ferramenta é a ideal para ir de encontro aos requisitos deste projeto: uma aplicação escalável, responsiva e com uma interface amigável e de fácil uso. A aplicação a ser desenvolvida é então uma plataforma de apostas, denominada *BetESS*.

Este relatório incide então no processo de desenvolvimento desta primeira fase do projeto, iniciando na análise dos requisitos e abordando a modelação do sistema e a criação das vistas.

2 Desenvolvimento

Tendo em conta a introdução feita ao *WebRatio* no capítulo anterior, a sua utilização torna-se lógica. O ambiente gráfico amigável, baseado em IFML para projetar a componente *front-end* do sistema e a fácil integração com o *back-end* fazem com que esta ferramenta contribua para a escalabilidade, extensibilidade e alta performance da solução a desenvolver. O processo de desenvolvimento do projeto BetESS é composto então por duas principais fases:

- Construção do **modelo de domínio** que especifica as entidades do sistema, os seus atributos e os relacionamentos entre si.
- Desenvolvimento de um conjunto de **vistas do *site***, que demonstram o fluxo de interações entre utilizadores e entre estes e o sistema. Nas vistas são construídas as diferentes páginas que compõem o *website* e os componentes que satisfazem os requisitos funcionais do sistema.

2.1 Modelo de Domínio

O modelo de domínio do *WebRatio* permite definir qual a informação que a aplicação irá ter acesso e a sua estrutura. Esta representação dos dados é importante pois será a ligação entre a camada de negócio da aplicação e a base de dados.

Um modelo de domínio do *WebRatio* inclui sempre, por defeito, três entidades com os seus respetivos atributos. São elas a entidade *User*, *Group* e *Module*.

A primeira refere-se a cada utilizador individual do sistema e é composto pelos atributos definidos pelo *WebRatio* e ainda por um atributo que guarda o dinheiro que cada utilizador tem na aplicação.

A entidade *Group* refere-se os atores do sistema, ou seja, o tipo de utilizadores que compõem esta aplicação. Neste caso, analisando os requisitos exigidos, esta aplicação será composta por três grupos: Administrador, Utilizador padrão e Utilizador *premium*.

Por fim, a entidade *Module* define as áreas que serão criadas nas vistas de *site* e que queremos que sejam restritas ou protegidas. O acesso de um determinado grupo a um módulo é definido no relacionamento N-N entre estas duas entidades, registando o grupo e o respetivo módulo a que podem

ter acesso. Esta gestão de acessos aos utilizadores será melhor abordada mais à frente.

Após percebermos a importância das três entidades geradas pelo programa, temos de as integrar no modelo de domínio que satisfaz os requisitos do sistema.

A entidade *Team* representa uma equipa, armazenando a sua informação essencial, como nome ou símbolo. Esta equipa encontra-se registada numa ou mais competições (entidade *Competition*) que por sua vez são compostas também por várias equipas. O núcleo dos dados deste sistema assenta-se depois nas entidades *Event* e *Bet*. A primeira representa um evento, como um jogo de futebol e é composta por duas equipas, pertencendo a uma competição. Essa entidade tem ainda atributos como as *odds* (probabilidades de vitória de cada equipa), o *status* (define se um evento ainda está a decorrer ou se já se encontra encerrado), o resultado final do encontro e o tipo de evento, que define se o evento é para todos os utilizadores (*type=0*) ou apenas para utilizadores *premium* (*type=1*). A entidade *Bet* representa uma aposta. A aposta é efetuada sobre um único evento e inclui o atributo *result* (1 para vitória da equipa da casa, 2 para empate, 3 para vitória da equipa de fora), o atributo *amount* que estipula o valor apostado, a *odd* dessa aposta e o lucro que pode vir a dar a aposta. Caso a aposta seja perdida, a variável *profit* passa a 0. Por fim, a entidade *Notification* regista as notificações sobre resultados de apostas efetuadas pelo utilizador.

Tendo em conta todas as entidades supra mencionadas e respetivos atributos e relacionamentos, foi possível construir o seguinte modelo de domínio:

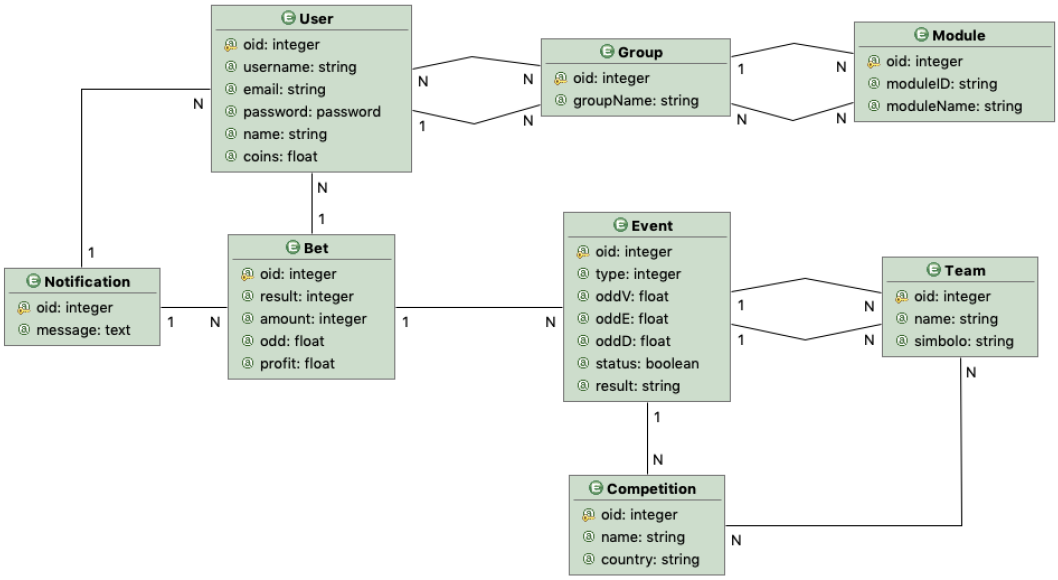

Figura 1: Modelo de domínio do sistema.

2.2 Base de dados

A plataforma *WebRatio* facilita a criação da base de dados para o modelo de domínio gerado. O seu suporte a diversos motores e a fácil integração tornam o processo de geração das tabelas muito simples. Assim, o motor de base de dados relacional utilizado foi o *MySQL* e a geração das tabelas foi efetuado dentro do próprio *WebRatio*, utilizado a opção *synchronize* que se encontra dentro da página do modelo de domínio.

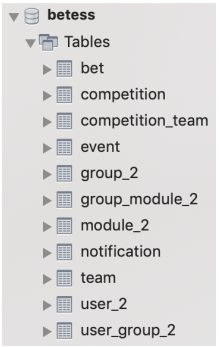

Figura 2: Tabelas da base de dados no *MySQL Workbench*.

2.3 Vistas do *site*

Uma vista do *site* (*site view*) é uma porção do fluxo de interações entre um utilizador ou dispositivo específico e o sistema. Diferentes grupos de utilizadores têm diferentes funcionalidades e responsabilidades dentro de um sistema, logo, a cada um deles é atribuída uma vista distinta.

Neste caso em específico, teremos três vistas distintas: *Login*; Administrador e Utilizador.

2.3.1 Login

A vista de *Login* é a área comum a todos os utilizadores do sistema. Inicialmente, é demonstrada uma página a partir da qual é possível efetuar a autenticação para entrada na área do respetivo utilizador ou fazer o registo no sistema como um novo apostador. Durante a autenticação, é introduzido o nome de utilizador bem como a palavra-passe do mesmo. Caso as credenciais não sejam válidas, o utilizador é redirecionado de volta à página inicial. No momento de registo, caso houverem dados de identificação já presentes na base de dados do sistema ou os dados introduzidos forem de outro modo inválidos, será demonstrada uma mensagem de erro. Caso contrário, será criada a conta. Em ambos os casos, o utilizador é depois redirecionado de volta à página inicial.

Com isto em conta, foi criada a seguinte vista do site:

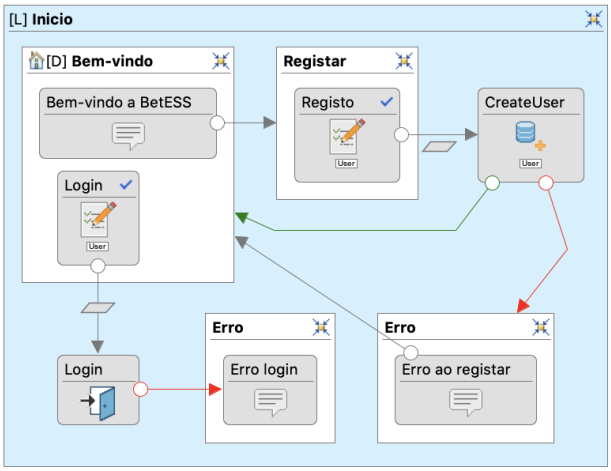

Figura 3: *Site view* do *Login*.

Ao proceder ao *login*, o utilizador autenticado é dirigido à sua vista correspondente. O relacionamento entre o tipo de utilizador (grupo) e a vista a que deve ser redirecionado é feita no relacionamento 1-N entre *Group* e *Module*.

oid	moduleid	modulename
0	sv1	Admin
1	sv4	User

Figura 4: Tabela *Module*.

oid	groupname	module_2_oid
0	Admin	0
1	User	1
2	UserPremium	1

Figura 5: Tabela *Group*.

Na tabela *Module* são registados os IDs dos módulos restritos, neste caso as *site views* do administrador (ID=sv1) e de um utilizador (ID=sv4). Depois na tabela *Group* registamos o módulo por defeito de cada ator do sistema.

2.3.2 User

A vista *User* demonstra as atividades que um determinado apostador pode efetuar após a autenticação no sistema. Como esta necessita de comportar uma variedade mais larga de funcionalidades, cada uma destas é seccionada na sua própria área:

- As áreas **Eventos** são as responsáveis por disponibilizar ao apostador todos os eventos ativos naquele momento. Para verificar se o evento encontra-se ativo adicionou-se uma restrição ao atributo *status*. O facto de existirem duas áreas distintas deve-se ao facto de uma apresentar todos os eventos que estejam ativos (incluíndo eventos *premium*) e a outra apresentar apenas os eventos ativos para utilizadores padrão. A gestão do acesso restrito a cada uma destas áreas é explicado mais à frente.

- A área **Evento** é aberta apenas quando um utilizador seleciona um evento da lista de jogos disponíveis. Assim sendo, não se encontra disponibilizada no menu da vista (*Landmark off*). Nesta área o apostador pode ver as informações sobre o jogo e formar uma aposta sobre ele.

  No formulário *Apostar* existem os campos *Resultado* e Valor. Estes campos definem o resultado previsto do evento (vitória para a equipa da casa, empate ou vitória para a equipa de fora) e o valor apostado. Os dados sobre o evento são passados ao selecionar o evento na lista através de *parameter binding*. O utilizador que está a efetuar a aposta é obtido através de um *session component* que devolve vários dados sobre o utilizador autenticado, incluíndo o seu ID. O atributo *profit* da tabela *Bet* é obtido automaticamente pela multiplicação entre a *odd* e o valor da aposta. Desta forma, temos todos os atributos necessários para o registo da aposta.

  O registo é efetuado por um *operation component* que associa os campos que lhe são passados aos atributos definidos no modelo de domínio através, novamente, de *parameter binding*.

  Caso a aposta seja inválida, é disponibilizada uma mensagem. A verificação da validade de uma aposta é efetuada com o auxílio de *triggers* na base de dados. Falaremos mais à frente disso.

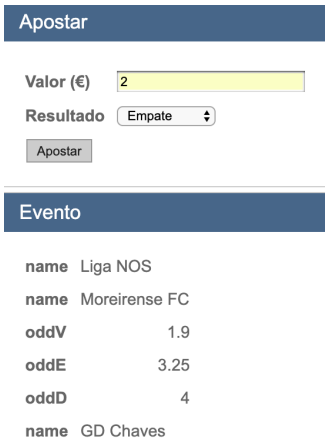

Figura 6: Preenchimento do formulário de efetuar uma aposta.

- Caso a aposta seja válida, o utilizador é redirecionado para a sua lista de **Apostas**. Nesta área encontram-se todas as apostas efetuadas pelo utilizador e cujo evento ainda não foi encerrado. Uma vez que o evento ainda continua ativo, o utilizador pode remover a sua aposta, sendo o valor apostado ressarcido e a aposta eliminada da sua lista. Nesta área, o utilizador pode também ver mais informações acerca do evento sobre o qual apostou.

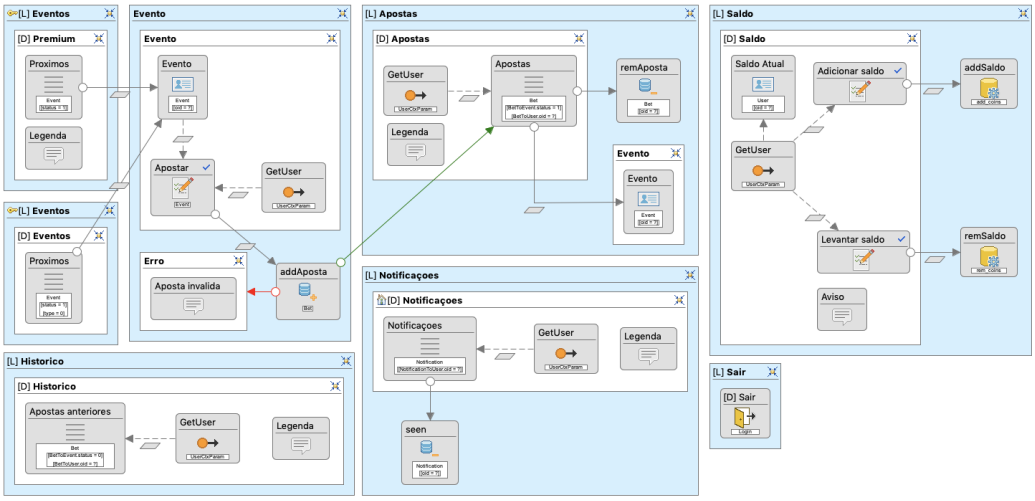

Figura 7: *Site view* completa do *User*.

- A área **Notificações** é a página inicial deste sistema. Nela são apresentadas as novidades de eventos sobre os quais o atual utilizador efetuou apostas. As notificações são geradas no momento em que um administrador define o resultado de um evento e são enviadas para todos os utilizadores que registaram apostas. Incluem uma pequena mensagem que notifica o utilizador do sucesso ou insucesso da sua aposta e dos ganhos obtidos na aposta.

Figura 8: Notificações de apostas de um utilizador.

- O **Histórico** apresenta todas as apostas efetuadas pelo utilizador em eventos que já terminaram. Estas apostas são obtidas através de duas condições sobre os atributos *status* de *Event* e o utilizador que efetuou a aposta. O primeiro deve ser 0, pois são apostas sobre eventos já encerrados, e o segundo atributo é obtido através do *session component getUser* que devolve o ID do utilizador autenticado, como já foi explicado anteriormente. Ao contrário das notificações, o histórico não apresenta uma mensagem, nem permite "marcar como lidas"as apostas.

- A área **Saldo** comporta a lógica responsável por mostrar ao utilizador o seu saldo atual, bem como providenciar a opção de adicionar ou levantar dinheiro da sua conta na aplicação.

- A área **Sair**, como o nome indica, permite terminar a sessão do utilizador no sistema, levando o mesmo de volta à página principal da vista *Login*.

2.3.3 Admin

A vista *Admin* está associada, como o nome indica, aos administradores da aplicação e às permissões associadas a estes nos requisitos exigidos pelo sistema.

Esta vista apresenta uma larga variedade de funcionalidades ao administrador, separadas nas suas áreas distintas:

- A área **Jogos** encontra-se dividida em três páginas principais acessíveis pelo menu da vista (*landmark on*), também elas com funcionalidades distintas:

– A página **Jogos ativos** é a página inicial da vista de administrador e apresenta todos os jogos ativos de momento, incluíndo eventos *premium*. Aqui é permitido eliminar esse evento ou introduzir o resultado final. Ao selecionar a opção de inserir resultado, o administrador é redirecionado para a página *Terminar jogo*, onde pode inserir o resultado e registar no evento. O processo de término do jogo é efetuado através de uma *stored procedure* na base de dados, uma vez que a ação de terminar uma partida envolve muitos passos, como a distribuição dos ganhos por utilizadores, geração de notificações, alteração do estado do evento, etc.

– A página dos **Jogos anteriores** apresenta uma lista dos eventos que já encerraram e dados que o administrador pode achar interessantes, como o número de apostas efetuadas sobre aquele jogo (*count(Bet)*), o total de dinheiro apostado nesse evento *(sum(amount)*) e o total de ganhos que os utilizadores obtiveram nesse evento (*sum(profit)*).

Lista de jogos

type	name	name	result	name	oddV	oddE	oddD	count(Bet)	sum(amount)	sum(profit)
1	UEFA Europa League	Eintracht Frankfurt	2-0	SL Benfica	1.75	3.75	4.5	1	2	0
1	UEFA Europa League	Napoli	0-1	Arsenal	1.62	4	4.5	0		
1	UEFA Europa League	Chelsea	4-3	Slavia Praha	1.32	4.75	8.3	0		
1	UEFA Europa League	Valencia CF	2-0	Villarreal CF	1.7	3.7	4.7	1	1	0
1	UEFA Champions League	Juventus	1-2	Ajax	1.69	3.99	5.48	1	2	0
1	UEFA Champions League	FC Barcelona	3-0	Manchester United	1.49	5.68	9.25	0		
1	UEFA Champions League	FC Porto	1-4	Liverpool FC	3.68	3.72	2.47	1	2	4.94
1	UEFA Champions League	Manchester City	4-3	Tottenham	1.31	6.16	11.27	1	2	2.62
0	Liga NOS	GD Chaves	2-2	Belenenses SAD	1.9	3.21	4.34	1	1	3.21
0	Liga NOS	CD Santa Clara	1-1	Moreirense FC	2.27	3.03	3.13	1	2	0

Figura 9: Página *Jogos anteriores*.

– Por fim, a página **Criar jogo** é a responsável por permitir a criação de um novo evento. A página apresenta um único formulário com os campos necessários à criação de um evento: competição; equipa de casa e fora; odds e tipo de evento (default ou premium).

A competição e as equipas são selecionadas através de uma *selection field* que apresenta todos os dados disponíveis na base de

dados para esse campo, cabendo ao utilizador selecionar o que pretende. Também é obtido o último ID da lista de eventos. Isto deve-se a um problema encontrado no *WebRatio* que, de alguma forma, dava erro se a criação da instância de Evento fosse feita da maneira tradicional, ou seja, através de um *create operation component*. A utilização de um *stored procedure* foi a alternativa encontrada para contornar esse problema, mas exigiu a necessidade de obter o último ID dos eventos.

O facto de a criação de um evento ser efetuada por uma *stored procedure* permitiu por outro lado a verificação imediata das condicionantes que permitam a adição de um jogo, como por exemplo, se as equipas selecionadas se encontram inscritas na competição selecionada ou se não foram selecionadas as mesmas equipas.

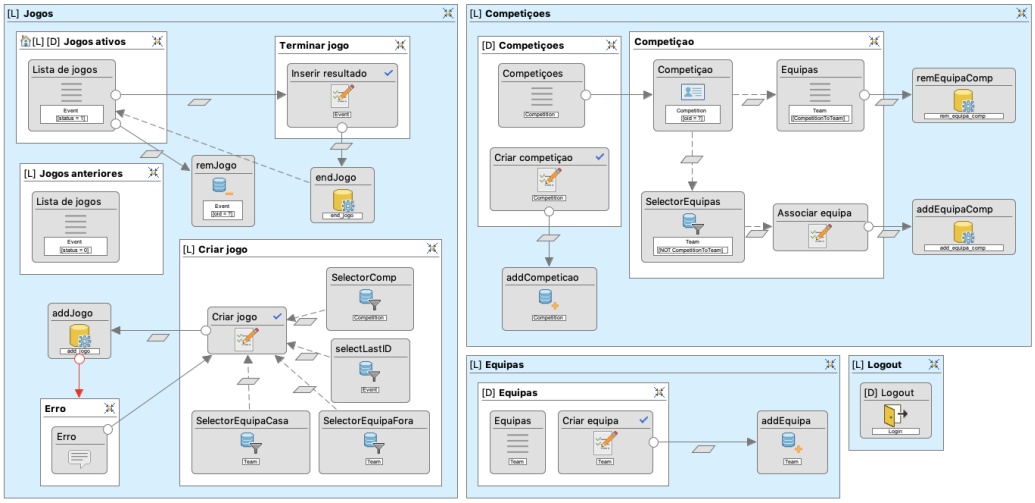

Figura 10: *Site view* completa do *Admin*.

- A área **Competições** apresenta como sua página inicial a lista das competições existentes e um formulário onde é possível criar uma nova competição. O formulário é bastante simples, sendo apenas necessário indicar o nome e o país da competição.

  Ainda na lista das competições, podemos selecionar uma competição e, noutra página, apenas acessível através dessa ligação, visualizar as equipas que se encontram registadas nessa competição e associar novas

equipas a esta competição, bem como remover essa associação a equipas que já estejam na lista.

Como a associação entre equipas e competições é uma ligação N-N, a tabela *competition_team*, fruto desse relacionamento, não se encontra no modelo de domínio desenvolvido no *WebRatio*, logo a associação terá de ser feita recorrendo a *stored procedures*.

O formulário para associar uma nova equipa à competição é bastante simples, apresentando apenas um *selection field* com todas as equipas, exceto as que já se encontram associadas, filtradas através de uma condição no *selector*. O administrador apenas tem de selecionar uma das equipas disponíveis e registar. O ID da competição é passado desde a lista de competições até ao *stored procedure* através de *parameter binding*, na opção *passing*.

- Por fim, a área **Equipas** apresenta a lista completa de equipas existentes e ainda um formulário para a adição de uma nova equipa. Este formulário é bastante simples, pedindo apenas nome e símbolo (link de imagem) do clube. Esses dados são então passados à base de dados através de um *create operation*.

- A área **Logout**, como o nome indica, permite terminar a sessão do administrador no sistema, levando o mesmo de volta à página principal da vista *Login*.

2.3.4 Verificação de condições

Uma das vantagens do *WebRatio* e da linguagem IFML é a abstração da modelação feita e o facto de o fluxo de interações ser fácil e prático de gerar. Porém, esta linguagem e o seu método de gerar um aplicativo, não permitem definir alguma parte da lógica do negócio.

A alternativa encontrada para verificar algumas das condições necessárias para o cumprimento dos requisitos impostos neste trabalho prático e a execução de ações em cadeia foi a criação de *stored procedures* e *triggers*.

Algumas das condições existentes resolvidas com *triggers* ou *stored procedures*:

- Utilizador não pode apostar num evento ao qual já apostou. Solução: *trigger 'new_bet'*.

- Utilizador não pode apostar um montante superior ao seu saldo. Solução: *trigger 'bet_add_coins'*.
- Ao remover uma aposta, restaurar o dinheiro apostado ao utilizador. Solução: *trigger 'bet_rem_coins'*.
- Ao registar uma aposta, preencher os atributos *odd* e *profit*. Solução: *trigger 'bet_add_odd_profit'*.
- Ao registar um novo utilizador, verificar se o *username* ou o *email* já se encontram registados no sistema. Solução: *trigger 'new_user'*.
- Quando o administrador introduz o resultado de um evento e o fecha, o sistema deve automaticamente guardar esse resultado, passar o estado do evento para fechado (*status=false*) e, por cada aposta nesse evento: distribuir os ganhos de cada utilizador; atualizar o valor de *profit* para 0 caso o utilizador tenha perdido a aposta e gerar a notificação para o utilizador. Solução: *stored procedure 'end_jogo'*.
- Ao adicionar um novo evento, verificar se as equipas pertencem à competição selecionada e se as equipas que se vão defrontar não são a mesma. Solução: *stored procedure 'add_jogo'*.

Todos estes *triggers* e *stored procedures* encontram-se guardados num ficheiro SQL no repositório do projeto.

2.3.5 Restrições de acesso

Como referimos acima, para além das restrições de acesso às respetivas vistas, também dentro de uma vista, nomeadamente a vista *User*, temos duas áreas protegidas, são elas as áreas dos Eventos.

Isto deve-se a uma área apresentar todos os eventos, incluíndo os *premium*, e a outra área apresentar apenas os eventos *default*.

A definição dos grupos que têm acesso a estas áreas protegidas é feita na ligação N-N entre *Group* e *Module*. Assim, nos módulos, adicionamos as áreas restritas e na tabela do relacionamento N-N são introduzidos os grupos que têm acesso aos respetivos módulos. O resultado final dessas tabelas é o seguinte:

oid	moduleid	modulename
0	sv1	Admin
1	sv4	User
2	area14	EventosAll
3	area15	EventosDefault

Figura 11: Tabela *Module* e as respetivas áreas restritas.

group_2_...	module_2_oid
1	1
1	3
2	1
2	2

Figura 12: Tabela do relacionamento entre *Group* e *Module* e os módulos a que cada grupo pode aceder.

3 Conclusões e trabalho futuro

Terminando o desenvolvimento desta primeira fase do trabalho que incidiu na modelação do fluxo de interações do sistema, podemos afirmar que a sua realização permite ter uma noção rápida do sistema que estamos a construir e permite também que a deteção de erros e falhas na sua construção seja rápida e a sua correção imediata e fácil, uma vez que a ferramenta *WebRatio* permite a compilação da modelação numa aplicação *web* real.

Na próxima etapa, tanto o modelo de domínio e consequente base de dados, como as vistas desenvolvidas serão utilizadas como base para o desenvolvimento de uma nova aplicação *web full-stack*, sendo que agora podemos passar diretamente para a produção de código, uma vez que sabemos de que forma o projeto deve ser desenvolvido, de forma a minimizar a quantidade e custo dos erros e falhas no desenvolvimento.