Implementação de um sistema de apostas em corridas de cavalos

Relatório Final

Sistemas de Informação, SINF

2º Semestre 2016

MIEEC

Turma: 2

Grupo: 23

Nomes: Diogo Martins e Pedro Costa

Índice

1. Funcionalidades implementadas 2

2. Modelo Entidade-Associação..... 7

3. Modelo Relacional 8

4. Script SQL 9

5. Instruções de Compilação 11

6. Manual de Instruções..... 12

6.1 Funções de Anónimo (qualquer pessoa tem acesso) 12

6.1.1 \help 12

6.1.2 \identify |username| 12

6.1.3 \register <username> <password>..... 13

6.1.4 \login <username> <password> 13

6.1.5 \exit..... 13

6.2 Funções de Visitante, Utilizador e Administrador: 14

6.2.1 \agenda 14

6.2.2 \log <horse> <limit>..... 14

6.2.3 \ranking..... 14

6.3 Funções de Utilizador e Administrador: 15

6.3.1 \bet <raceId> <horse> <value> 15

6.3.2 \logout..... 15

6.4 Funções exclusivas do Administrador:..... 16

6.4.1 \add <horse> <speed> 16

6.4.2 \create <description> <number> <laps> 16

6.4.3 \start <raceId>..... 16

6.4.4 \shutdown..... 16

7. Exemplos de Interação com o Sistema Implementado..... 17

1. Funcionalidades implementadas

\help

- Mostra uma lista com os comandos possíveis.

\register <username> <password>

- Verifica se recebeu 2 parâmetros. Se não, indica condição de erro e pára;
- Verifica se o utilizador já existe na BD (SELECT * FROM user WHERE username='<username>';). Se sim, indica condição de erro e para;
- Cria utilizador na base de dados (INSERT INTO user VALUES (default, '<username>', '<password>', 200, false) ;).

\identify |<username>|

- Verifica se recebeu no máximo um parâmetro. Se não, indica condição de erro e para;
- Caso não haja parâmetros, é atribuído aleatoriamente ao cliente um *username* de visitante (*visitor_243235* p.e.), caso contrário é atribuído o *username* escolhido como parâmetro (*visitor_username* p.e.).

\login <username> <password>

- Verifica se recebeu 2 parâmetros. Se não, indica condição de erro e para;
- Verifica se o *username* e password existem na base de dados (SELECT * FROM user WHERE username='<username>' AND password='<password>';). Se sim, indica condição de erro e para;
- Guarda a informação sobre o utilizador que está a usar este *socket*.

\logout

- Encerra sessão dos utilizadores;
- Apaga a memória temporária associada ao utilizador;
- Fecha o *socket* associado ao utilizador.

\add <horse> <speed>

- Funcionalidade exclusiva do administrador;
- Verifica se recebeu 2 parâmetros. Se não, indica condição de erro e para;
- Verifica se já existe um cavalo com o mesmo nome (SELECT * FROM horse WHERE name='<horse>';). Se sim, indica condição de erro e para;
- Insere um novo cavalo na base de dados com um determinado nome e velocidade (INSERT INTO horse VALUES (default, '<name>', '<speed> ') ;).

\create <description> <number> <laps>

- Funcionalidade exclusiva do administrador;
- Verifica se recebeu 3 parâmetros. Se não, indica condição de erro e para;
- Verifica se os números de cavalos e de voltas são válidos;
- Recolhe uma lista aleatória de todos os cavalos (SELECT horse_id, name FROM project.horse ORDER BY random()) e verifica se existem cavalos suficientes para a corrida. Se não, indica condição de erro e para;
- Cria uma nova corrida com o número escolhido de voltas e de cavalos - primeiros da lista - (INSERT INTO race VALUES (default, '<description>', '<number>', '<laps>', 'betting'); INSERT INTO run_in VALUES (horse_id, currval(pg_get_serial_sequence ('race','race_id')), NULL); INSERT INTO run_in …).

\bet <raceId> <horse> <value>

- Um utilizador não pode apostar mais do que uma vez na mesma corrida.
- Verifica se recebeu 3 parâmetros. Se não, indica condição de erro e para;
- Verifica se o utilizador ainda não apostou na corrida e se tem saldo suficiente para a aposta (SELECT balance FROM user WHERE user_id = user_id AND user_id NOT IN (SELECT user_id FROM bet WHERE race_id = race_id);
- Verifica se a corrida existe, se está em fase de apostas, e se o cavalo existe e está inscrito na corrida (SELECT horse_id, race_id, state FROM run_in JOIN horse USING (horse_id) JOIN race USING (race_id) WHERE race_id = <race_id> AND horse.name = '<horse>'). Caso contrário, indica condição de erro e para;
- Aposta num determinado cavalo, na corrida atual, uma certa quantidade de créditos (UPDATE user SET balance = new_balance WHERE user_id = <user_id>; INSERT INTO bet VALUES (default, <value>, NULL, <user_id>, <horse_id>, <race_id>)).

\start <raceId>

- Funcionalidade exclusiva do administrador;
- Verifica se a corrida existe e se está na fase se apostas (SELECT state, description, laps, horses FROM project.race WHERE race_id = <race_id>);
- Inicia a corrida com o respetivo *raceid* (UPDATE race SET state = 'running' WHERE race_id = <race_id>; SELECT horse_id, name, speed FROM horse JOIN run_in USING(horse_id) WHERE race_id = <race_id>);
- No fim de cada volta, os utilizadores são avisados da posição de cada cavalo;
- No fim da última volta, a classificação final é anunciada e cada utilizador que apostou recebe uma mensagem com informação sobre os seus ganhos (UPDATE race SET state = 'finished' WHERE race_id = <race_id>; UPDATE run_in SET final_position = <final_position> WHERE horse_id = <horse_id> AND race_id = <race_id>; UPDATE bet SET result = <result> WHERE race_id = <race_id> AND user_id = <user_id> AND horse_id = <horse_id>; UPDATE user SET balance = <balance> WHERE user_id = <user_id>; UPDATE run_in SET…; UPDATE bet SET…; UPDATE user SET…; …).

\log <horse> <limit>

- Verifica se recebeu 2 parâmetros. Se não, indica condição de erro e para;
- Mostra uma lista das classificações do cavalo selecionado nas últimas corridas (SELECT race_id, description, final_position FROM run_in JOIN horse USING(horse_id) JOIN race USING(race_id) WHERE name = '<horse>' AND state = 'finished' ORDER BY race_id DESC LIMIT <limit>).

\info <username>

- Verifica se recebeu 1 parâmetro. Se não, indica condição de erro e para.
- Mostra informação sobre o utilizador *username*, o seu saldo atual e o número de apostas em cavalos que ganharam a corrida. Se o *username* não for indicado, o utilizador é o que executa o comando (SELECT balance, vitories FROM user LEFT JOIN (SELECT user_id, COUNT(*) AS vitories FROM bet JOIN run_in USING(race_id, horse_id) WHERE final_position = 1 GROUP BY user_id) AS temp USING(user_id) WHERE username = <username>).

\agenda

- Mostra a agenda das corridas, referindo as que estão em fase de apostas e as que estão a decorrer (SELECT race_id, state, description, laps, horses, horse.name FROM project.run_in JOIN race USING (race_id) JOIN horse USING (horse_id) WHERE state = 'betting' OR state = 'running').

\ranking

- Mostra um *ranking* dos utilizadores baseados no seu saldo atual (SELECT username, balance FROM project.user ORDER BY balance DESC).

\exit

- Desliga a conexão deste cliente.

\shutdown

- Funcionalidade exclusiva do administrador;
- Desliga a conexão de todos os clientes.
- Desliga o servidor.

2. Modelo Entidade-Associação

Entidades:	**Associações**:
User (<u>userId</u>, username, password, balance)	DoneOn (Bet, Horse) n:1 t/p
Horse (<u>horseId</u>, name, speed)	DoneBy (Bet, User) n:1 t/p
Bet (<u>betId</u>, value, result)	DoneOn (Bet, Race) n:1 t/p
Race (<u>raceId</u>, description, laps, horses, state)	RunIn (Horse, Race) n:n p/p

3. Modelo Relacional

user [user_id || username | password | balance | admin]
horse [horse_id || name | speed]
bet [bet_id || value | result | #user_id →user NN | #race_id →race NN | #horse_id→horse NN]
race [race_id || description | laps | horses | state]
run_in [#horse_id → horse | #race_id → race || final_position]

4. Script SQL

```
CREATE TABLE project.user
(
	user_id SERIAL PRIMARY KEY,
	username varchar(20) NOT NULL UNIQUE,
	password varchar(20) NOT NULL,
	balance numeric NOT NULL,
	admin boolean NOT NULL
);



CREATE TABLE project.horse
(
	horse_id SERIAL PRIMARY KEY,
	name varchar (100) UNIQUE,
	speed integer CHECK (speed > 0 AND speed <= 100) NOT NULL
);

CREATE TABLE project.race
(
	race_id SERIAL PRIMARY KEY,
	description varchar (200),
	laps integer CHECK (laps > 0 AND laps <= 50) NOT NULL,
	horses integer CHECK (horses > 0 AND horses <= 20) NOT NULL,
	state varchar(20) CHECK (state = 'betting' OR state = 'running' OR state = 'finished')
);


CREATE TABLE project.bet
(
	bet_id SERIAL PRIMARY KEY,
	value integer CHECK (value > 0 AND value <= 10) NOT NULL,
	result integer,
	user_id integer REFERENCES project.user NOT NULL,
```

```
	horse_id integer REFERENCES project.horse NOT NULL,
	race_id integer REFERENCES project.race NOT NULL,
	CONSTRAINT unique_bet UNIQUE (user_id, horse_id, race_id)
);

CREATE TABLE project.lap_position
(
	lap_position_id SERIAL PRIMARY KEY,
	lap_num integer CHECK (lap_num > 0) NOT NULL,
	position integer CHECK (position > 0) NOT NULL,
	horse_id integer REFERENCES project.horse NOT NULL,
	race_id integer REFERENCES project.race NOT NULL
);



CREATE TABLE project.run_in
(
	horse_id integer REFERENCES project.horse,
	race_id integer REFERENCES project.race,
	final_position integer CHECK (final_position > 0),
	PRIMARY KEY (horse_id, race_id)
);
```

5. Instruções de Compilação

O utilizador, para poder compilar o nosso programa, terá que estar conectado num computador da FEUP, na rede ou via VPN. O passo seguinte será utilizar um *software* que faça a ligação entre o servidor da Faculdade e o computador que está a ser usado, por exemplo, o ***Putty***. Através da máquina ***gnomo.fe.up.pt,*** o utilizador usará as suas credenciais do SIFEUP para dar entrada nas contas. Desde a linha de comandos, localiza o ficheiro através dos comandos ***cd*** e ***ls;*** e com este comando ***g++ -pthread server.cpp -o server -lpq*** estará a compilar o programa. O servidor pode ser com ***./server*** **<port>** em que *port* será a porta escolhida para a conexão ao servido.

6. Manual de Instruções

Tal como em 5. é necessário, em primeiro lugar, estar ligado à rede da FEUP e entrar no diretório aonde está localizado o programa. Em seguida inicia o programa (servidor) com o comando ***./server*** **<port>**, em que *port* é a porta escolhida para a conexão ao servidor.
Para se ligar ao servidor será necessário entrar na rede da FEUP com o respectivo username e password e executar a linha de comandos ***telnet gnomo.fe.up.pt <port>*** em que *port* é a porta do servidor já inicializado. A partir daí, use as instruções, que se seguem:

6.1 Funções de Anónimo (qualquer pessoa tem acesso)

6.1.1 \help

O utilizador receberá a listagem de funcionalidades existentes. Caso o utilizador seja um mero espectador, esta é a única interação que poderá ter com o programa. Portanto o utilizador tem agora três opções: ir embora, efetuar *login* (ou registo, caso não esteja ainda registado), ou aproveitar a função de visitante.

6.1.2 \identify |username|

O utilizador que não queira fazer *login* ou efetuar um registo, poderá, usando este comando, ter informações sobre a agenda, *ranking* de utilizadores e informações dos cavalos. Ao usar o comando, poderá acrescentar um nome de utilizador.

6.1.3 \register <username> <password>

O utilizador pode registar-se usando o comando e introduzindo um nome de utilizador e uma *password*. O limite é de 20 carateres em ambos os parâmetros. Os utilizadores são únicos, não havendo possibilidade de nomes repetidos.

6.1.4 \login <username> <password>

Usando o comando e preenchendo os seguintes parâmetros, o utilizador pode agora fazer todas as operações que não sejam exclusivas ao administrador.

6.1.5 \exit

O utilizador registrado ou visitante poderá sair do servidor.

6.2 Funções de Visitante, Utilizador e Administrador:

6.2.1 \agenda

São listadas as especificações das várias corridas agendadas, número e nome da corrida, cavalos a participar e número de voltas.

6.2.2 \log <horse> <limit>

Com esta funcionalidade, o utilizador poderá saber o histórico de classificações do cavalo selecionado. Logo, após a \log, deverá escrever o nome do cavalo do qual quer saber informações e introduzir um valor numérico, que significará o número de corridas que queremos pesquisar.

6.2.3 \ranking

Listagem da classificação, com nome do utilizador e os respetivos saldos.

6.3 Funções de Utilizador e Administrador:

6.3.1 \bet <raceId> <horse> <value>

O utilizador poderá apostar na corrida colocando o comando, o número da corrida, o cavalo que pretende apostar e o valor de 0 a 10 que quer na aposta. Apenas pode apostar uma vez na mesma corrida.

6.3.2 \logout

O utilizador que estava registado pode fazer *logout*.

6.4 Funções exclusivas do Administrador:

6.4.1 \add <horse> <speed>

O administrador adiciona cavalos à base de dados, inserindo nome e respetivo valor/speed.

6.4.2 \create <description> <number> <laps>

Criação de corridas na agenda. Para isso, basta nomear a mesma, dar um número de cavalos que podem participar e também o número de voltas que a corrida tem.

6.4.3 \start <raceId>

O administrador inicializa a corrida. Todos os utilizadores receberão as informações do decorrer da corrida, volta a volta, nos seus respetivos terminais. No fim, apresentará a classificação e o dinheiro ganho.

6.4.4 \shutdown

O servidor vai abaixo, terminando com tudo o que está a acontecer.

7. Exemplos de Interação com o Sistema Implementado

```
\help
> Commands:

        \help
        \register <username> <password>
        \identify |<username>|
        \login <username> <password>
        \logout
        \agenda
        \add <horse> <speed>
        \create <description> <number> <laps>
        \bet <raceId> <horse> <value>
        \start <raceId>
        \log <horse> <limit>
        \info |<username>|
        \ranking
        \exit
        \shutdown
```

```
\register antonio 1234
> Success. Welcome antonio!
```

```
\identify
> Success. Welcome visitor_367535!
```

```
\identify joana
> Success. Welcome visitor_joana!
```

```
\login pedro 1234
> Success. Welcome pedro!
```

```
\add storm 90
> Success.
\create aveiro 4 20
> Success. Selected horses: rapido, LL, moderado, JJJ.
```

```
\create coimbra 14 20
```

```
> Fail. The number of existing horses is 11.
```

```
\bet 34 rapido 8
> Success. You bet 8 credits on the horse 'rapido' for the race
34. Your balance is 189 credits.
```

```
\bet 34 rapido 3
> Fail. You have already bet in race 34.
```

```
\start 34

> Race 34 (aveiro) started.

> Race 34 - lap 1:
        1 - moderado (7300 points)
        2 - rapido (7200 points)
        3 - JJJ (4000 points)
        4 - LL (1400 points)


> Race 34 - lap 2:
        1 - moderado (25550 points)
        2 - rapido (14400 points)
        3 - JJJ (12000 points)
        4 - LL (3500 points)
…

> Race 34 - lap 20:
        1 - JJJ (228000 points)
        2 - moderado (222650 points)
        3 - rapido (80640 points)
        4 - LL (42000 points)

> Race 34 (aveiro) is over.
You bet 8 credits and you won 0. Your balance is 189 credits.

> Success. Race 34 (aveiro) has finished.
```

```
\log rapido 5
> Horse 'rapido' results:
        race 34 (aveiro): 3rd
        race 32 (aveiro): 2nd
        race 28 (lisbonrun): 2nd
        race 27 (portorun): 1st
```

```
\info
```

```
> Player 'pedro' info:
        balance - 189
        victories - 2
```

```
\info diogo
> Player 'diogo' info:
        balance - 166
        victories - 1
```

```
\agenda
> Agenda:
        -Race 33:
                state: running
                description: alentejo
                number of laps: 10
                horses (5):
                        veloz
                        moderado
                        JJ

        -Race 34:
                state: running
                description: algarve
                number of laps: 8
                horses (5):
                        pachorrento
                        lento
                        veloz
```

```
\ranking
> Raking:
        1 - andre (200 credits)
        2 - diogo (200 credits)
        3 - antonio (200 credits)
        4 - joana (200 credits)
        5 - joao (189 credits)
        6 - pedro (187 credits)
```

```
\exit
> Success. Bye!
```