

# Raport Project Databases Maart

Mathias Beke      Bruno Van de Velde      Elias Van Langenhove  
Alexander Vanhulle      Timo Truyts

19 maart 2014

## Inhoudsopgave

### 1 Status

#### 1.1 Taakverdeling

1. ERM Schema: samen opgesteld. Digitalisatie: Mathias<sup>1</sup>
2. ERM Model omzetten in SQL tables: Alexander, Timo, Elias<sup>2</sup>
3. Parser schrijven: Bruno, Mathias<sup>3</sup>
4. Basis datastructuren: Alexander, Elias
5. *Teams* tabel vullen (Bruno)
6. *Coaches* tabel vullen (Bruno)
7. *Player* tabel moet extra informatie meekrijgen zoals lengte, gewicht, positie, geboortedatum, ... Bruno
8. *Date* attribuut van tabellen in orde brengen (Elias)
9. Feitelijk login script, met sessions, registraties, login pagina, ... (Alexander)
10. Verdere implementatie van *get* queries voor de klassen. (Elias)

---

<sup>1</sup>De verdere digitalisatie en verdere aanpassingen werden door Alexander gedaan.

<sup>2</sup>De verfijning hiervan (wat het meeste tijd in beslag neemt) werd vooral door Timo gedaan.

<sup>3</sup>Bruno heeft zich vooral toegelegd op de parser, en heeft er dan ook veruit het meeste voor gedaan.

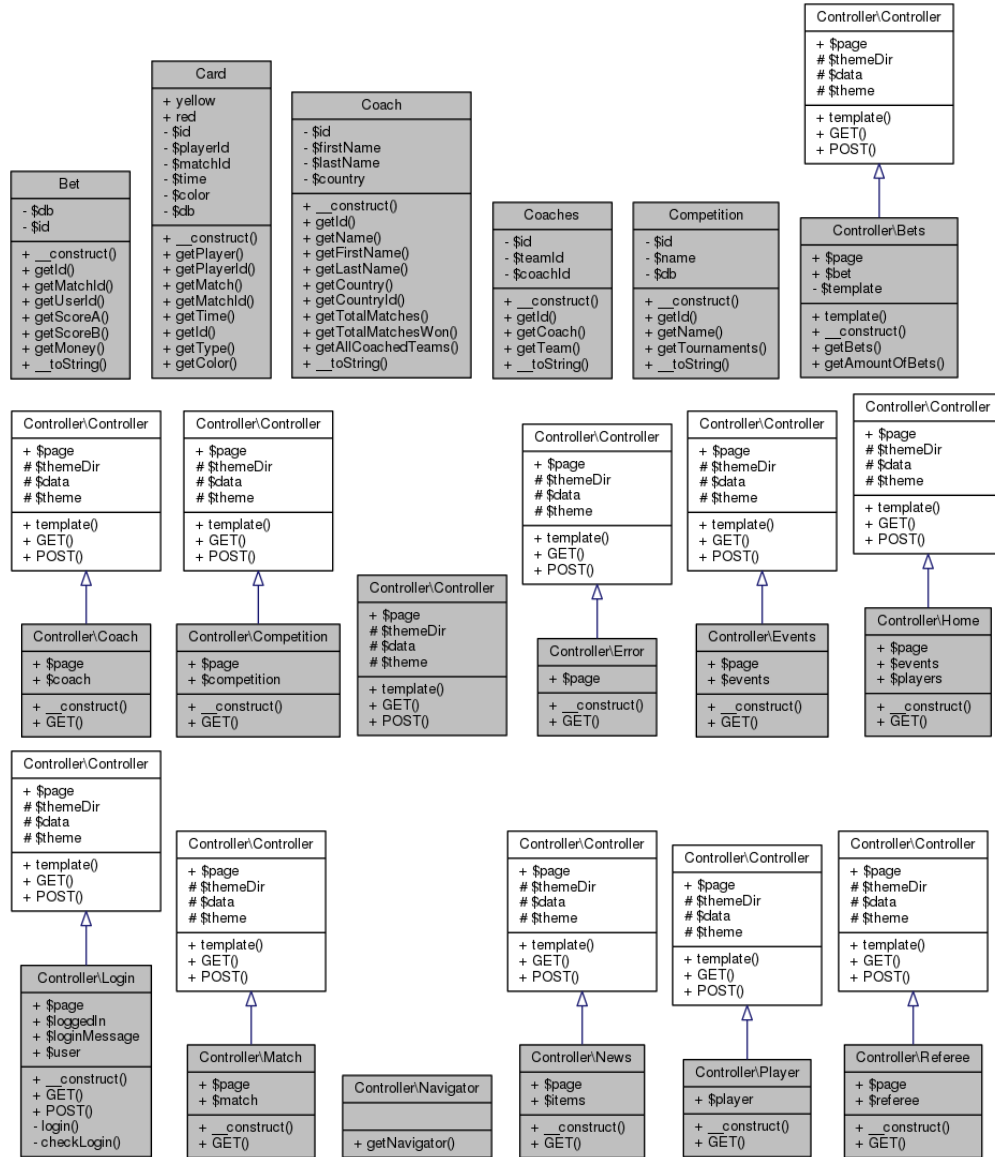
11. Pagina's en algemeen ontwerp van de site (Timo en Mathias)
12. *Analyse van de data* Timo
13. *Herstructureren van de klassen* Elias
14. *Crawler die automatisch nieuwe wedstrijden binnenhaalt, en de scores update van gespeelde wedstrijden.* Bruno
15. *Informatie includen op pagina's* Mathias
16. *RSS feed ophalen* Mathias
17. *Configuratie scherm gebruiker* Alexander
18. *Pagina's stylen, nadat ze geïmplementeerd zijn* Alexander<sup>4</sup>
19. *Update functies/queries voor goals en wedstrijden* Elias
20. *Functies voor de 'bets'* Alexander
21. *Geavanceerde view, met statistieken e.d.* Mathias
22. *Parser afmaken* Bruno
23. *Prognose* Timo
24. *Constraints* Timo
25. *Bets uitbreiden* Alexander
26. *Unit tests database* Elias
27. *Laatste details aan user interface* Mathias

---

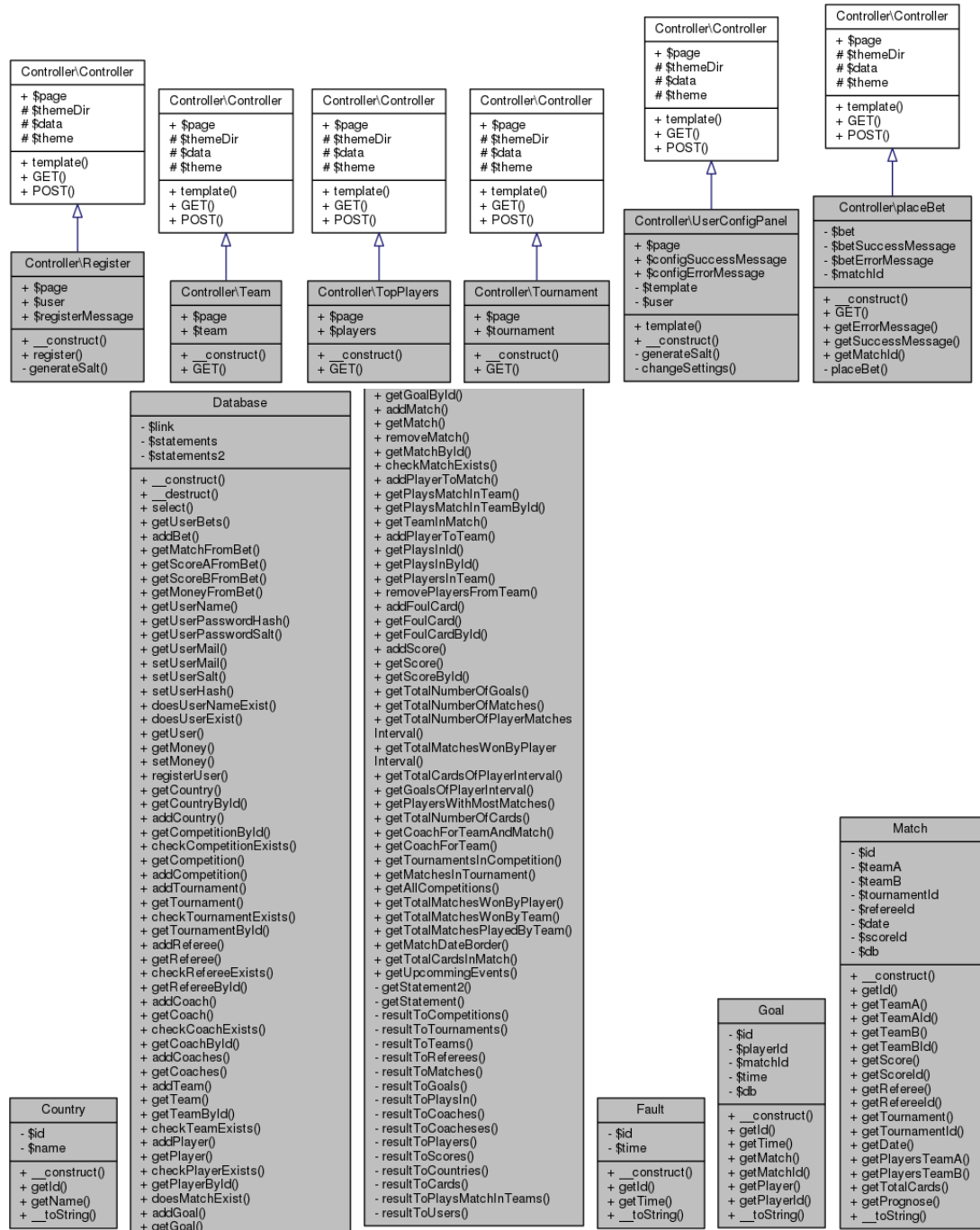
<sup>4</sup>Alexander heeft vooral de gebruikers- en wed pagina's van een design voorzien. Het algemeen design werd vooral door Mathias gedaan, gebruik makende van Bootstrap

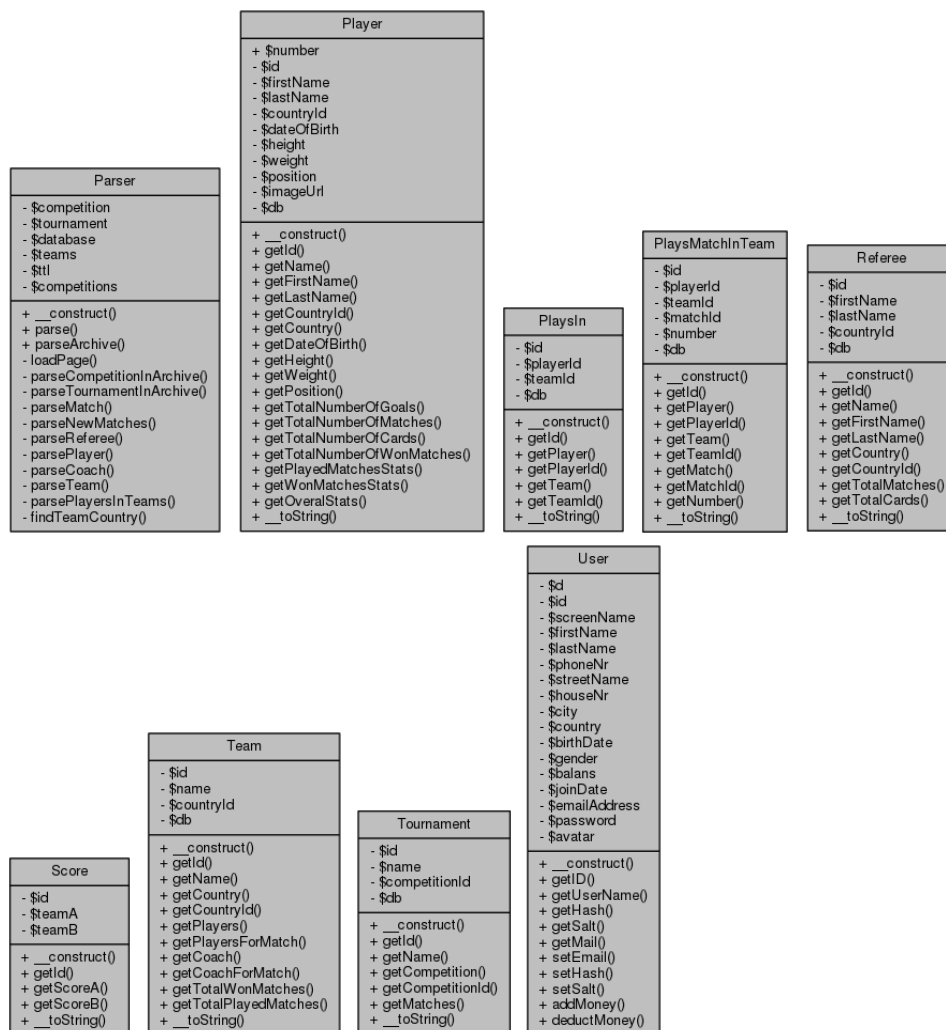
## 2 Design

### 2.1 UML diagram



## Report Project Databases: Maart





## 2.2 Parser

De parser bestaat uit twee delen. De parseArchive functie zal, voor alle geselecteerde competities, alle wedstrijden van de afgelopen jaren gaan opzoeken en de data extraheren en in de database opslaan. Het is de bedoeling dat deze functie slechts eenmalig moet worden aangeroepen. Het andere deel, de crawler, zal regelmatig opnieuw worden uitgevoerd. Het zal de huidige competities in de gaten houden door nieuwe wedstrijden toe te voegen en de gegevens voor gespeelde wedstrijden te updaten.

Om het internetverkeer minder te belasten zal elke pagina die wordt binnen-

gehaald gecached worden. Voor het archief zullen deze pagina's in de cache blijven, voor andere pagina's zal de pagina na een bepaalde tijd opnieuw worden gedownload.

De urls van de competities zijn hardcoded, maar vanaf deze pagina's worden alle matches en teams door de parser ontdekt. Dus zelfs al zouden de urls wijzigen, dan zou dit haast geen werk kosten. Maar normaal moet de parser zonder problemen de komende jaren blijven werken.

Intern wordt gebruik gemaakt van de *PHP Simple HTML DOM Parser* om de inhoud van de pagina's te parsen.

Uiteraard worden niet enkel de matches geparsed, maar zal de parser ook informatie over de spelers, teams, coaches en zelfs scheidsrechters zoeken en opslaan in de database.

## 2.3 MVC

Om geen spaghetti te maken van de implementatie van de pagina's, hebben we geopteerd voor het *Model View Controller Pattern*. We routeren onze urls via *GluePHP* en *mod\_rewrite*. Apache geeft alle urls door aan *GluePHP*, welke a.h.v. gedefiniëerde reguliere expressies de url zal mappen op een bepaalde klasse. Het framework zal dan de GET of POST functie (afhanginge van het type request) uitvoeren op de nieuwe instantie van de juiste klasse. (In onze implementatie hebben we er echter voor gezorgd dat het framework dat object ook terugkeert, zodat we het verder kunnen gebruiken. Verder hebben we ook een catch-all controller toegevoegd aan het framework om error pagina's te kunnen genereren.)

Zo laten we *GluePHP* een instantie van een *Controller* aanmaken, die we bij het aanmaken de nodige datastructuren en informatie zal inladen. Op elke controller kunnen we dan een *template* functie oproepen, die het overeenkomstige thema gedeelte zal insluiten op de pagina.

**Controller** Onderstaand php fragment toont hoe we de juiste controller verkrijgen a.h.v. de reguliere expressies. We definiëren een array van urls die we mappen op een controller klasse.

```
$urls = array(  
    'error' => 'Controller\Error', //Catch all  
    INSTALL_DIR . 'player/(\d+)' => 'Controller\  
    Player',
```

```
INSTALL_DIR . 'competition/(\d+)' => '
    Controller\Competition',
INSTALL_DIR . 'match/(\d+)' => 'Controller\
    Match',
INSTALL_DIR . 'news' => 'Controller\News',
);

$controller = glue::stick($urls);
```

**Theme** Wanneer we de juiste controller hebben, kunnen we de correcte template bestanden includen voor die welbepaalde controller. (zie onderstaande code)

```
//Include the header template
include(dirname(__FILE__) . '/theme/header.php');

//Include the theme part
$controller->template();

//Include the footer template
include(dirname(__FILE__) . '/theme/footer.php');
```

## 3 Database

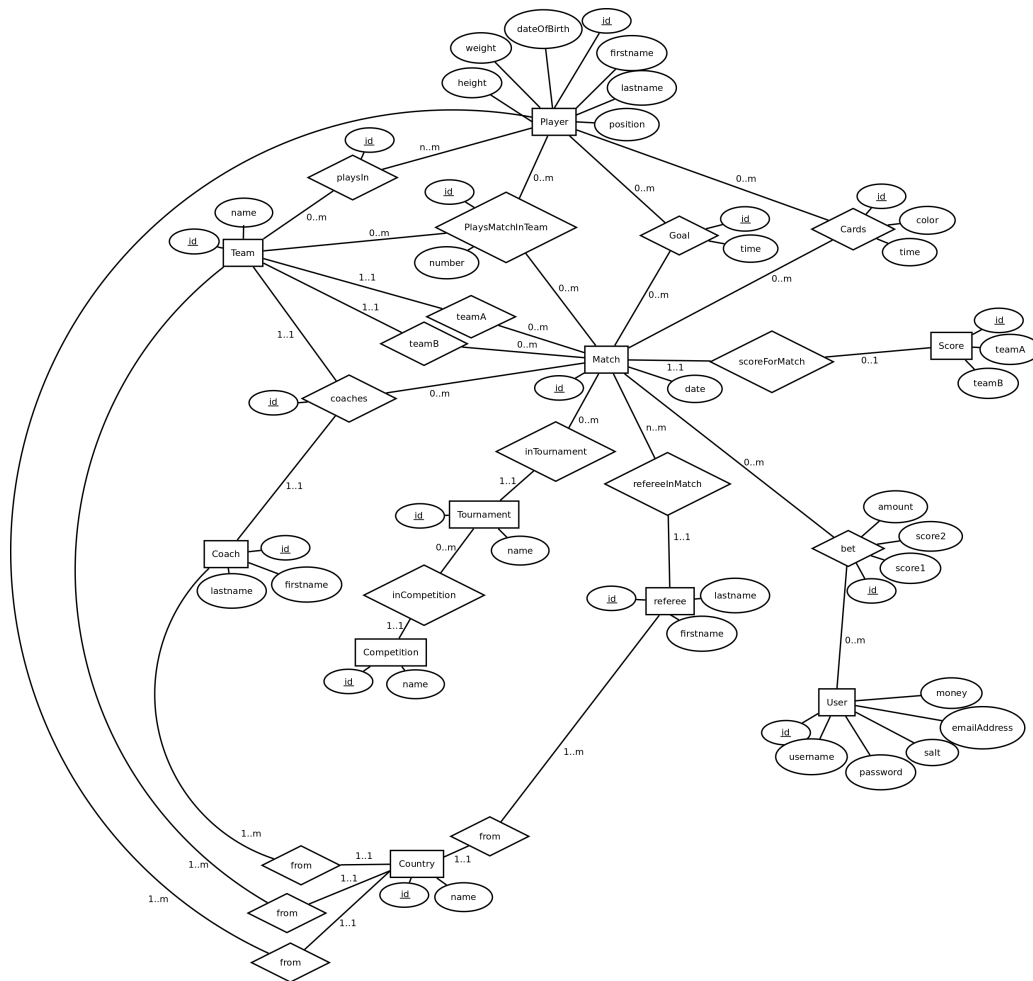
### 3.1 Schema (ERM-diagram)

Bij het aanvatten van een project met zo'n omvangrijke database structuur als hier, is het erg belangrijk om van in het begin een solide database structuur af te spreken. Hiervoor hebben we dan ook meteen onze eerste samenkomst gebruikt. We hebben goed nagedacht over welke zaken en hoe we deze met elkaar in verband zouden brengen (wat niet zo eenvoudig was gezien niemand van onze groep een echte voetbalkenner is).

Het heeft alleszins zijn vruchten afgeworpen gezien onze huidige (geïmplementeerde) database structuur erg weinig afwijkt van de eerste kribbel die we destijds op papier gezet hadden (met uitzondering van het nog niet geïmplementeerde deel).

In de ERM zijn we vertrokken vanuit de entiteiten Match, Player en Team. Hierop hebben we de rest van het 'voetbal' gedeelte van onze ERM gebouwd. Ten slotte hebben we nog een 'gebruikers' gedeelte toegevoegd.

*Gezien de omvang van de ERM kan het zijn dat deze apart moet geopend worden om deze voldoende leesbaar te kunnen bekijken. (bijlages/ERM.png)*



Figuur 1: ERM schema



## 4 User Interface

De user interface concentreert zich vooral op matches (met de voorspellingen en weddenschappen) en de spelers. Gebruikers navigeren vanaf de homepage door de competities, seizoenen (binnen zo'n competitie) en krijgen dan een overzichtspagina van een bepaalde wedstrijd. Op deze match pagina staat dan wat algemene informatie zoals de score, scheidsrechters, ..., geflankeerd door de teams met hun spelers (voor die bepaalde match) opgelijst. Gebruikers kunnen te allen tijde doorklikken op spelers, teams, coaches, competities, om zo telkens weer op een overzichtelijke pagina te komen waarop wat meer informatie staat.

**Spelers** Bij de spelers voorzien we de bezoeker van de nodige statistische data over een speler. Zo staan er het aantal gespeelde wedstrijden, gewonnen wedstrijden, kaarten en goals. Daarnaast staat er ook wat metadata over de spelers, zoals diens nationaliteit, positie op het veld, lengte, gewicht, etc. Onderaan de pagina kan de bezoeker een grafiek zien met de wedstrijden van het laatste jaar. Daaronder staat een overzichts grafiek die de tijdlijn van de speler voorstelt. Met daarop alle matches van zijn carrière. (Zie figuur 5 p. 12 voor een voorbeeld van een speler pagina.)

**Gebruikers** Gebruikers kunnen op de website reeds registreren, inloggen, accountinstellingen wijzigen en bets plaatsen. Het registratie-, login- en controle paneel is voorzien van voldoende checks die nagaan of de door de user opgegeven informatie correct is. Zo kan een user bijvoorbeeld niet registreren met een syntactisch ongeldig emailadres of een gebruikersnaam die reeds bestaat. We hebben getracht ons user systeem volgens huidige maatstaven erg veilig te maken. Wanneer een gebruiker registreert, dan wordt er een random salt gegenereerd en vervolgens wordt de sha256 hash berekend aan de hand van het gekozen wachtwoord, geconcateneerd met de salt. In de database worden de salt en het gehashte wachtwoord opgeslagen. Bij het inloggen wordt min of meer hetzelfde gedaan: de sha256 hash wordt wederom berekend aan de hand van de salt van de user (in de database) en het wachtwoord dat de user ingeeft. Als de berekende hash dan overeenkomt met de hash van de user in de database, dan was het wachtwoord correct en is de gebruiker ingelogd.

Een gebruiker kan reeds bets plaatsen op een score voor een bepaalde match en achteraf de reeds geplaatste bets bekijken. Dit systeem is echter nog vrij

ZeeJong Upcoming Events News Leagues Bets User Logout

### Change your account settings

Leave fields open if you don't want them to be changed.  
Fields marked (\*) are required.

Your new emailaddress is invalid.  
New passwords do not match.

Current Password (\*)

New Email Address

New Password

Please re-type New Password

[Update settings](#)

©2014 ZeeJong Betting System

Figuur 2: Schermafbeelding van het userConfigPanel

primitief waardoor de bets bijvoorbeeld bij afloop van een match niet berekend worden en gewoon 'blijven staan'. Het plaatsen van een bet gebeurt door naar de pagina van een bepaalde wedstrijd te gaan en op de 'bet' knop te drukken. Vervolgens landt de gebruiker op een pagina bij welke hij een bepaalde score en geldbedrag kan kiezen.

ZeeJong Upcoming Events News Leagues Bets User Logout

### Place a bet

Score Team 1

Score Team 2

Amount of money

[Place](#)

Information

Shakhtar Donetsk vs Werder Bremen

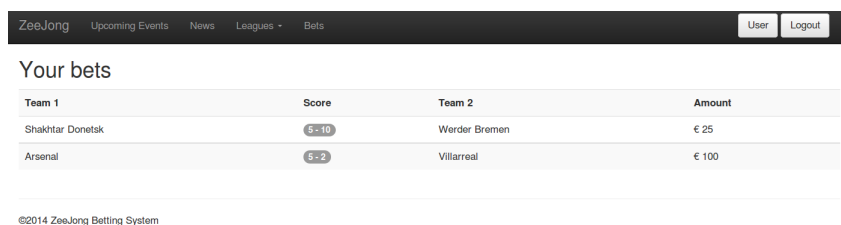
Date: 20-05-2009

Prognose: 0-1

©2014 ZeeJong Betting System

Figuur 3: Schermafbeelding van de place bet pagina

**Design** Om het design tot een goed einde te brengen hebben we gebruik gemaakt van het *Bootstrap* framework. Hierdoor hadden we een stevige basis om op te steunen bij het stylen van de pagina's. (Voor de installatie pagina



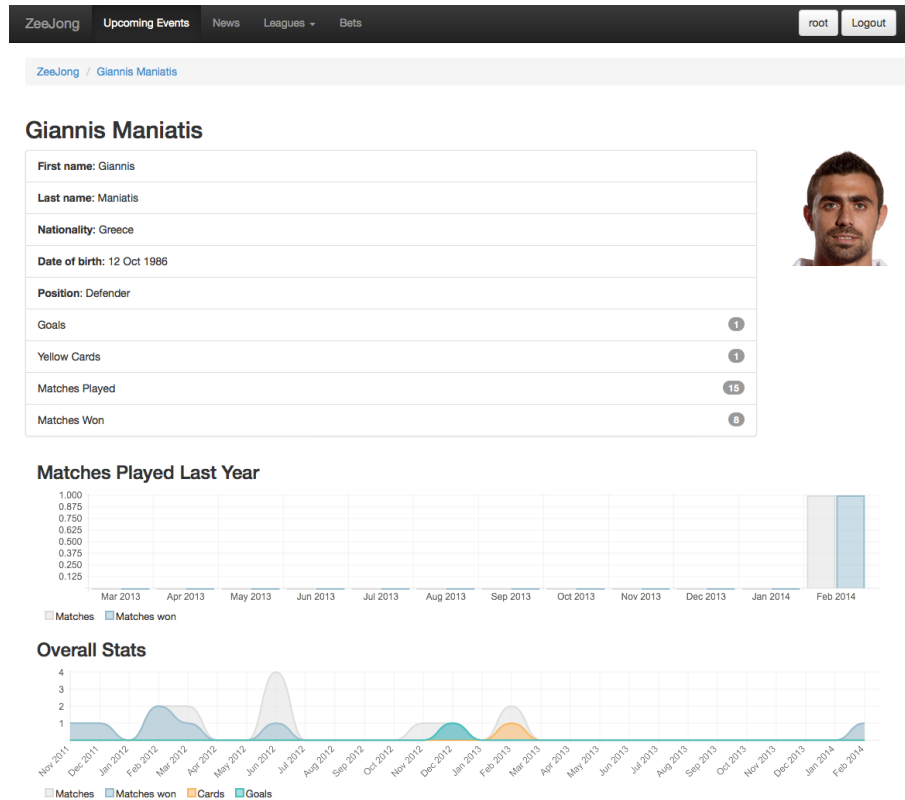
The screenshot shows a web interface for 'ZeeJong'. The top navigation bar includes links for 'Upcoming Events', 'News', 'Leagues', and 'Bets', along with 'User' and 'Logout' buttons. Below the navigation bar, the heading 'Your bets' is displayed. A table lists two bets with columns for 'Team 1', 'Score', 'Team 2', and 'Amount'. The first bet is for Shakhtar Donetsk vs Werder Bremen with a score of 5-10 and an amount of € 25. The second bet is for Arsenal vs Villarreal with a score of 5-2 and an amount of € 100. At the bottom, a copyright notice reads '©2014 ZeeJong Betting System'.

Team 1	Score	Team 2	Amount
Shakhtar Donetsk	5-10	Werder Bremen	€ 25
Arsenal	5-2	Villarreal	€ 100

©2014 ZeeJong Betting System

Figuur 4: Schermafbbeelding van de bets pagina

hebben we echter de minimalistische framework *Pure CSS* gebruikt.) De grafieken worden in javascript gerenderd op de html5 canvas d.m.v. de javascript module *ChartJS*. Intern hebben we slechts één framework gebruikt, namelijk *GluePHP*. Dit framework zet urls om in klasse objecten op basis van reguliere expressies. Zo konden we met Apache's *mod\_rewrite* onze urls afhandelen.



Figuur 5: Schermafbeelding van de speler pagina

## 5 Extra Functionaliteit

### 5.1 Selector class

**Selector** Om te voorkomen dat er zeer veel verschillende queries geschreven moeten is er een wrapper geschreven die deze queries automatisch genereert. Momenteel werkt deze wrapper alleen voor niet al te ingewikkelde select queries.

Bijvoorbeeld: We willen weten hoeveel goals een speler heeft gemaakt:

```
$sel = new Selector('Goal'); // Selecteer Goal  
column
```

```
$sel->filter ([[ 'playerId', '=', $playerId ]]); //  
    Filter op de speler  
$sel->count(); // Tel het aantal resultaten.
```

De selector kan dan aan de database worden gegeven die de sql query zal omzetten in een prepared statement en uitvoeren. In dit geval zal de volgende query gegenereerd worden:

```
SELECT COUNT(*) FROM 'Goal' WHERE ('playerId'=?)
```

De gebruiker krijgt dan als resultaat een associatieve array terug met de values die geselecteerd zijn.

Hier nog enkele voorbeelden:

**Voorbeeld 1** Selecteer de spelers in een bepaald team

```
$sel = new Selector('PlaysIn');  
$sel->filter ([[ 'teamId', '=', $teamId ]]);  
$sel->join('Player', 'playerId', 'id');  
$sel->select('Player.*');
```

**Voorbeeld 2** Tel hoeveel matches een team gespeeld heeft

```
$sel = new Selector('Match');  
$sel->filter(  
    [  
        // We moeten 1 van de teams zijn in deze match  
        ['teamA', '=', $teamId],  
        ['teamB', '=', $teamId]  
    ]  
);
```

```
// Als de scoreId niet NULL is hebben we gespeeld
$sel->filter ([[ 'scoreId', 'IS_NOT_NULL', '' ]]);
$sel->count();
```

## 5.2 Unit Test Framework

Om ons ervan te vergewissen dat de database functies volledig werken, hebben we unit tests voorzien. Voor vele talen vind je dan zeer snel een zeer eenvoudig framework. Dit was echter niet meteen snel te vinden voor PHP. Daarom hebben we zelf een test framework geschreven, waarmee het schrijven van unit tests veel gemakkelijk moet gaan.

Een test *case* wordt gedeclareerd door simpelweg een PHP klasse te declareren. Een test *section* binnen zulk een *case* is dan gewoon een functie met daarin enkele uitvoerbare tests. Voor het uitvoeren van de tests zijn er enkele *REQUIRE* functies, zoals *require\_true*, *require\_false*, *require\_equal*, *require\_notequal* en *require\_exception*.

Een voorbeeld van een test bestand ziet er dan als volgt uit:

```
class Example extends UnitTest {

    public function AnotherTest() {

        $this->REQUIRE_EQUAL($var, $this->
            testVarA);
        $this->REQUIRE_NOTEQUAL('one', 'two'
            );
        $this->REQUIRE_TRUE('one' == 'one');
        $this->REQUIRE_FALSE('one' == 'two')
            ;

    }

    public function ExceptionTest() {

        $this->BEGIN_REQUIRE_EXCEPTION();
```

```
        //A statement between BEGIN_REQUIRE
        and
        //END_REQUIRE should throw an
        exception
        //in order to make the performed
        test 'passed'.

        divide(2, 0);

        $this->END_REQUIRE_EXCEPTION();

    }

}
```

De gebruiker van het test framework plaatst alle te uitvoeren tests in de *test* map, en het framework handelt de rest af. Je krijgt uiteraard een overzichtelijke html output bij het runnen van de tests (Zie figuur 6 p. 16).

### 5.3 RSS nieuwsfeed

Om de gebruikers meer interessante inhoud op de website te bieden, hebben we onze site voorzien van een *News* pagina, waarop we met RSS de laatste voetbal nieuwtjes binnehalen. Omdat zelf een RSS parser schrijven geen gemakkelijke klus is, gebruiken we *SimplePie RSS*, een nieuwer alternatief voor *Maggie RSS*.

19 tests in 3 test cases 4 failures		
Example		
Assignment		OK
AnotherTest		OK
ExceptionTest		ERROR
Failure on line 58		
<pre>\$this-&gt;BEGIN_REQUIRE_EXCEPTION();  //A statement between BEGIN_REQUIRE and //END_REQUIRE should throw an exception //in order to make the performed test 'passed'.  divide(2, 0);  \$this-&gt;END_REQUIRE_EXCEPTION();</pre>		
String		
Word		OK
Sentence		OK
SentenceWithMoreThanTwoWords		ERROR
Failure on line 19 \$this->REQUIRE_EQUAL('this is a sentence', 'which will not be equal to this part');		
AnotherSentence		OK
LoremIpsum		OK
DolerSit		OK

Figuur 6: HTML uitvoer van unit tests

## 6 Planning

Voor de volgende deadline verwachten we om het wedden op matches volledig geïmplementeerd te hebben. Daarnaast zou er ook een administratie pagina moeten komen voor dit geheel, waar de admins hun betting site in de gaten kunnen houden, en handmatig wedstrijden toevoegen of wijzigen. Tot slot zou de prognose ook accurater kunnen worden.

Zoals we reeds deden, zullen we ook nu weer elke donderdag afspreken om afspraken te maken en issues te bespreken.



## 7 Appendix

### 7.1 Volledige Queries

#### 7.1.1 Tabellen

**Bet** Bevat alle weddenschappen met de volgende informatie: match, score ploeg a, score ploeg b, tijdstip van weddenschap, de gebruiker, hoeveel geld er gewed is. De foreign keys zijn de gebruiker die de bet heeft gemaakt en de match waarop de bet is gezet.

```
CREATE TABLE IF NOT EXISTS 'Bet' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'matchId' int(11) NOT NULL,  
  'score1' int(2) DEFAULT NULL,  
  'score2' int(2) DEFAULT NULL,  
  'time' int(3) NOT NULL,  
  'userId' int(11) NOT NULL,  
  'amount' int(11) NOT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'matchId_2' ('matchId', 'score1', 'score2',  
    'userId'),  
  KEY 'userId' ('userId'),  
  KEY 'matchId' ('matchId')  
) AUTO_INCREMENT=1 ;
```

**Cards** Deze tabel bevat alle kaarten die spelers krijgen gedurende een wedstrijd. De tabel heeft dus als attributen de speler, match, kleur (geel/rood) en de tijd (d.i. het aantal verstreken minuten sinds het begin van de match)

```
CREATE TABLE IF NOT EXISTS 'Cards' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'playerId' int(11) NOT NULL,  
  'matchId' int(11) NOT NULL,  
  'color' tinyint(1) NOT NULL,  
  'time' int(3) NOT NULL,  
  PRIMARY KEY ('id'),
```

```
UNIQUE KEY 'playerId_3' ('playerId', 'matchId', 'time'),
KEY 'playerId' ('playerId', 'matchId'),
KEY 'playerId_2' ('playerId'),
KEY 'matchId' ('matchId')
) AUTO_INCREMENT=1 ;
```

**Coach** Bevat alle coaches met de volgende informatie: naam, land. Het land word gerefereerd door een foreign key.

```
CREATE TABLE IF NOT EXISTS 'Coach' (
'id' int(11) NOT NULL AUTO_INCREMENT,
'firstname' varchar(50) NOT NULL,
'lastname' varchar(50) NOT NULL,
'country' int(11) NOT NULL,
PRIMARY KEY ('id'),
UNIQUE KEY 'firstname' ('firstname', 'lastname', 'country'),
KEY 'country' ('country'),
KEY 'country_2' ('country')
) AUTO_INCREMENT=1 ;
```

**Coaches** Houdt bij wie de coach van een team is voor een bepaalde wedstrijd. Het bevat dus de volgende informatie: coach, team, match. Coach, team en match zijn foreign keys.

```
CREATE TABLE IF NOT EXISTS 'Coaches' (
'id' int(11) NOT NULL AUTO_INCREMENT,
'coachId' int(11) NOT NULL,
'teamId' int(11) NOT NULL,
'matchId' int(11) NOT NULL,
PRIMARY KEY ('id'),
```

```
    UNIQUE KEY 'coachId_3' ('coachId', 'teamId', 'matchId'),  
    KEY 'coachId' ('coachId', 'teamId'),  
    KEY 'matchId' ('matchId'),  
    KEY 'teamId' ('teamId'),  
    KEY 'coachId_2' ('coachId')  
  ) AUTO_INCREMENT=1 ;
```

**Competition** Bevat de namen van de competities. Bijvoorbeeld: World Cup, UEFA Europa League, ...

```
CREATE TABLE IF NOT EXISTS 'Competition' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'name' varchar(50) NOT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'name' ('name')  
  ) AUTO_INCREMENT=1 ;
```

**Country** Bevat de namen van landen.

```
CREATE TABLE IF NOT EXISTS 'Country' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'name' varchar(50) NOT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'name' ('name')  
  ) AUTO_INCREMENT=1 ;
```

**Goal** Bevat de volgende gegevens over een goal: speler, match, tijdstip. De speler en de match zijn de foreign keys.

```
CREATE TABLE IF NOT EXISTS 'Goal' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'playerId' int(11) NOT NULL,  
  'matchId' int(11) NOT NULL,  
  'time' int(3) NOT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'playerId_3' ('playerId', 'matchId',  
    'time'),  
  KEY 'playerId' ('playerId', 'matchId'),  
  KEY 'matchId' ('matchId'),  
  KEY 'playerId_2' ('playerId')  
) AUTO_INCREMENT=1 ;
```

**Match** Bevat de volgende informatie over een match: de 2 teams, het tournament, de scheidsrechter, datum, de uitslag.

De teams, het tournament, de scheidsrechter en de uitslag zijn foreign keys.

```
CREATE TABLE IF NOT EXISTS 'Match' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'teamA' int(11) NOT NULL,  
  'teamB' int(11) NOT NULL,  
  'tournamentId' int(11) NOT NULL,  
  'refereeId' int(11) DEFAULT NULL,  
  'date' int(11) NOT NULL,  
  'scoreId' int(11) DEFAULT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'teamA_2' ('teamA', 'teamB',  
    'tournamentId', 'date'),  
  KEY 'teamA' ('teamA', 'teamB'),  
  KEY 'tournamentId' ('tournamentId', 'refereeId'),  
  KEY 'scoreId' ('scoreId'),  
  KEY 'refereeId' ('refereeId'),  
  KEY 'tournamentId_2' ('tournamentId'),  
  KEY 'teamB' ('teamB')  
) AUTO_INCREMENT=1 ;
```

**Player** Bevat de volgende gegevens over een speler: naam, land, geboortedatum, lengte, gewicht, positie(aanvaller,verdediger,...)  
Het land is een foreign key.

```
CREATE TABLE IF NOT EXISTS 'Player' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'firstname' varchar(50) NOT NULL,  
  'lastname' varchar(50) NOT NULL,  
  'country' int(11) NOT NULL,  
  'dateOfBirth' int(11) DEFAULT NULL,  
  'height' int(3) DEFAULT NULL,  
  'weight' int(3) DEFAULT NULL,  
  'position' varchar(50) DEFAULT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'firstname' ('firstname', 'lastname',  
    country', 'dateOfBirth'),  
  KEY 'country' ('country')  
) AUTO_INCREMENT=1 ;
```

**PlaysIn** Linkt de spelers met hun huidige team. De foreign keys zijn dus de speler en het team.

Een speler heeft 1 huidige team, dit kan dus eventueel als extra informatie bij de speler worden opgeslagen. Hierdoor zal deze tabel overbodig worden.

```
CREATE TABLE IF NOT EXISTS 'PlaysIn' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'playerId' int(11) NOT NULL,  
  'teamId' int(11) NOT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'playerId_2' ('playerId', 'teamId'),  
  KEY 'playerId' ('playerId'),  
  KEY 'teamId' ('teamId')  
) AUTO_INCREMENT=1 ;
```

**PlaysMatchInTeam** Houd bij welke spelers voor welk team speelden in een bepaalde match. De foreign keys zijn de speler, het team en de match.

```
CREATE TABLE IF NOT EXISTS 'PlaysMatchInTeam' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'playerId' int(11) NOT NULL,  
  'number' int(11) NOT NULL,  
  'teamId' int(11) NOT NULL,  
  'matchId' int(11) NOT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'playerId_2' ('playerId', 'teamId', '  
    matchId'),  
  KEY 'playerId' ('playerId', 'teamId', 'matchId'),  
  KEY 'teamId' ('teamId'),  
  KEY 'matchId' ('matchId')  
) AUTO_INCREMENT=1 ;
```

**Referee** Bevat de volgende informatie over een scheidsrechter: naam, land. Het land is een foreign key.

```
CREATE TABLE IF NOT EXISTS 'Referee' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'firstname' varchar(50) NOT NULL,  
  'lastname' varchar(50) NOT NULL,  
  'countryId' int(11) NOT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'firstname' ('firstname', 'lastname', '  
    countryId'),  
  KEY 'countryId' ('countryId')  
) AUTO_INCREMENT=1 ;
```

**Score** Bevat de uitslagen, de score van team 1 en van team 2.

```
CREATE TABLE IF NOT EXISTS 'Score' (  
    'id' int(11) NOT NULL AUTO_INCREMENT,  
    'teamA' int(11) DEFAULT NULL,  
    'teamB' int(11) DEFAULT NULL,  
    PRIMARY KEY ('id'),  
    UNIQUE KEY 'teamA' ('teamA', 'teamB')  
) AUTO_INCREMENT=1 ;
```

**Team** Bevat de naam en het land van een team.  
Het land is de foreign key.

```
CREATE TABLE IF NOT EXISTS 'Team' (  
    'id' int(11) NOT NULL AUTO_INCREMENT,  
    'name' varchar(50) NOT NULL,  
    'country' int(11) NOT NULL,  
    PRIMARY KEY ('id'),  
    UNIQUE KEY 'name' ('name', 'country'),  
    KEY 'country' ('country')  
) AUTO_INCREMENT=1 ;
```

**Tournament** Bevat de naam en bij welke competitie het hoort.  
De competitie is de foreign key.

```
CREATE TABLE IF NOT EXISTS 'Tournament' (  
    'id' int(11) NOT NULL AUTO_INCREMENT,  
    'name' varchar(50) NOT NULL,  
    'competitionId' int(11) NOT NULL,  
    PRIMARY KEY ('id'),  
    UNIQUE KEY 'name_2' ('name', 'competitionId'),  
    KEY 'competitionId' ('competitionId'),  
    KEY 'name' ('name', 'competitionId')  
) AUTO_INCREMENT=1 ;
```

**User** Bevat informatie over de geregistreerde gebruikers(username, password, email, money).

```
CREATE TABLE IF NOT EXISTS 'User' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'username' varchar(50) NOT NULL,  
  'password' varchar(128) NOT NULL,  
  'salt' varchar(128) NOT NULL,  
  'emailAddress' varchar(50) NOT NULL,  
  'money' int(11) DEFAULT '0',  
  PRIMARY KEY ('id'),  
  UNIQUE KEY 'username' ('username')  
) AUTO_INCREMENT=1 ;
```

### 7.1.2 Constraints

```
—  
— Constraints for table 'Bet'  
—  
ALTER TABLE 'Bet'  
  ADD CONSTRAINT 'Bet_ibfk_2' FOREIGN KEY ('matchId'  
    ') REFERENCES 'Match' ('id'),  
  ADD CONSTRAINT 'Bet_ibfk_1' FOREIGN KEY ('userId')  
    REFERENCES 'User' ('id');  
  
—  
— Constraints for table 'Cards'  
—  
ALTER TABLE 'Cards'  
  ADD CONSTRAINT 'Cards_ibfk_2' FOREIGN KEY ('  
    matchId') REFERENCES 'Match' ('id'),  
  ADD CONSTRAINT 'Cards_ibfk_1' FOREIGN KEY ('  
    playerId') REFERENCES 'Player' ('id');  
  
—  
— Constraints for table 'Coach'
```



```

--
ALTER TABLE 'Coach'
  ADD CONSTRAINT 'Coach_ibfk_1' FOREIGN KEY ('
    country') REFERENCES 'Country' ('id');

--
-- Constraints for table 'Coaches'
--
ALTER TABLE 'Coaches'
  ADD CONSTRAINT 'Coaches_ibfk_3' FOREIGN KEY ('
    matchId') REFERENCES 'Match' ('id'),
  ADD CONSTRAINT 'Coaches_ibfk_1' FOREIGN KEY ('
    coachId') REFERENCES 'Coach' ('id'),
  ADD CONSTRAINT 'Coaches_ibfk_2' FOREIGN KEY ('
    teamId') REFERENCES 'Team' ('id');

--
-- Constraints for table 'Goal'
--
ALTER TABLE 'Goal'
  ADD CONSTRAINT 'Goal_ibfk_2' FOREIGN KEY ('matchId
    ') REFERENCES 'Match' ('id'),
  ADD CONSTRAINT 'Goal_ibfk_1' FOREIGN KEY ('
    playerId') REFERENCES 'Player' ('id');

--
-- Constraints for table 'Match'
--
ALTER TABLE 'Match'
  ADD CONSTRAINT 'Match_ibfk_5' FOREIGN KEY ('
    scoreId') REFERENCES 'Score' ('id'),
  ADD CONSTRAINT 'Match_ibfk_1' FOREIGN KEY ('teamA
    ') REFERENCES 'Team' ('id'),
  ADD CONSTRAINT 'Match_ibfk_2' FOREIGN KEY ('teamB
    ') REFERENCES 'Team' ('id'),
  ADD CONSTRAINT 'Match_ibfk_3' FOREIGN KEY ('
    tournamentId') REFERENCES 'Tournament' ('id'),
```

```
ADD CONSTRAINT 'Match_ibfk_4' FOREIGN KEY ('
    refereeId') REFERENCES 'Referee' ('id');

--
-- Constraints for table 'Player'
--
ALTER TABLE 'Player'
    ADD CONSTRAINT 'Player_ibfk_1' FOREIGN KEY ('
        country') REFERENCES 'Country' ('id');

--
-- Constraints for table 'PlaysIn'
--
ALTER TABLE 'PlaysIn'
    ADD CONSTRAINT 'PlaysIn_ibfk_2' FOREIGN KEY ('
        teamId') REFERENCES 'Team' ('id'),
    ADD CONSTRAINT 'PlaysIn_ibfk_1' FOREIGN KEY ('
        playerId') REFERENCES 'Player' ('id');

--
-- Constraints for table 'PlaysMatchInTeam'
--
ALTER TABLE 'PlaysMatchInTeam'
    ADD CONSTRAINT 'PlaysMatchInTeam_ibfk_3' FOREIGN
        KEY ('matchId') REFERENCES 'Match' ('id'),
    ADD CONSTRAINT 'PlaysMatchInTeam_ibfk_1' FOREIGN
        KEY ('playerId') REFERENCES 'Player' ('id'),
    ADD CONSTRAINT 'PlaysMatchInTeam_ibfk_2' FOREIGN
        KEY ('teamId') REFERENCES 'Team' ('id');

--
-- Constraints for table 'Referee'
--
ALTER TABLE 'Referee'
    ADD CONSTRAINT 'Referee_ibfk_1' FOREIGN KEY ('
        countryId') REFERENCES 'Country' ('id');
```

```
-- Constraints for table 'Team'
--
ALTER TABLE 'Team'
  ADD CONSTRAINT 'Team_ibfk_1' FOREIGN KEY ('country'
    'id') REFERENCES 'Country' ('id');

--
-- Constraints for table 'Tournament'
--
ALTER TABLE 'Tournament'
  ADD CONSTRAINT 'Tournament_ibfk_1' FOREIGN KEY ('
    competitionId') REFERENCES 'Competition' ('id')
  ;
```

### 7.1.3 Queries

**Getters: ID** Alle getters via id worden automatisch gegenereerd door een selector omdat deze allemaal een zeer gelijkaardige vorm hebben. Hier enkele voorbeelden, de id wordt telkens gegeven:

```
SELECT *
FROM Country
[WHERE id = ?];
```

```
SELECT *
FROM Match
[WHERE id = ?];
```

```
SELECT *
FROM Player
[WHERE id = ?];
```

**Getters: Attributen** Alle getters via attributen worden automatisch gegenereerd door een selector omdat deze allemaal een zeer gelijkaardige vorm hebben. Hier enkele voorbeelden, alle bepalende attributen, op de id na, worden telkens gegeven:

```
SELECT *  
FROM Country  
[WHERE name = ?];
```

```
SELECT *  
FROM Match  
[WHERE teamA = ? AND  
teamB = ? AND  
date = ? AND  
tournamentId = ?];
```

```
SELECT *  
FROM Player  
[WHERE firstname = ? AND  
lastname = ? AND  
country = ?];
```

**Inserts** Ook alle insert queries worden automatisch gegenereerd door een selector omdat deze allemaal een zeer gelijkaardige vorm hebben. Hier enkele voorbeelden, alle attributen, op de id na, worden telkens gegeven:

```
INSERT INTO Country (name)  
VALUES (?);
```

```
INSERT INTO 'Match' (teamA, teamB, tournamentId,
                    refereeId, date, scoreId)
VALUES (?, ?, ?, ?, ?, ?);
```

```
INSERT INTO Player (firstname, lastname,
                   country, dateOfBirth, height, weight,
                   position)
VALUES (?, ?, ?, ?, ?, ?, ?);
```

*Naast de simpele get en insert queries, blijven er nog enkele ingewikkeldere queries over die we handmatig hebben geschreven.*

**Gewonnen wedstrijden van speler binnen interval** Deze query vraagt de hoeveelheid matches waarbij de gegeven speler voor het winnende team speelde binnen een gegeven tijdspanne op.

```
SELECT COUNT(*)
FROM 'PlaysMatchInTeam'
[JOIN 'Match' ON 'Match'.id = matchId]
[JOIN 'Score' ON 'Score'.id = scoreId]
[WHERE playerId = ? AND
'Match'.date > ? AND
'Match'.date < ? AND
((teamId = 'Match'.teamA AND 'Score'.teamA > '
Score'.teamB) OR
(teamId = 'Match'.teamB AND 'Score'.teamB > '
Score'.teamA))];
```

**Gewonnen wedstrijden van speler** Deze query vraagt de hoeveelheid matches waarbij de gegeven speler in totaal voor het winnende team speelde.

```
SELECT COUNT(*)
FROM 'PlaysMatchInTeam'
[JOIN 'Match' ON 'Match'.id = matchId]
[JOIN 'Score' ON 'Score'.id = scoreId]
[WHERE playerId = ? AND
  ((teamId = 'Match'.teamA AND 'Score'.teamA > '
    Score'.teamB) OR
  (teamId = 'Match'.teamB AND 'Score'.teamB > '
    Score'.teamA))];
```

**Gewonnen wedstrijden van team** Deze query vraagt de hoeveelheid matches die het gegeven team in totaal gewonnen heeft op.

```
SELECT COUNT(*)
FROM 'Team'
[JOIN 'Match' ON 'Match'.teamA = 'Team'.id OR '
  Match'.teamB = 'Team'.id]
[JOIN 'Score' ON 'Match'.scoreId = 'Score'.id]
[WHERE 'Team'.id = ? AND
  (( 'Match'.teamA = 'Team'.id AND 'Score'.teamA >
    'Score'.teamB) OR
  ( 'Match'.teamB = 'Team'.id AND 'Score'.teamB > '
    Score'.teamA)) ];
```

**Gespeelde wedstrijden** Deze query vraagt de hoeveelheid matches dat de gegeven speler in totaal gespeeld heeft op.

```
SELECT COUNT(*)
FROM 'PlaysMatchInTeam' AS p
[INNER JOIN 'Match' AS m ON p.matchId = m.id]
[WHERE p.playerId = ? AND
  m.date > ? AND
  m.date < ?];
```

**Kaarten binnen interval** Deze query vraagt de hoeveelheid strafkaarten dat de gegeven speler heeft gekregen binnen een bepaalde tijdsperiode op.

```
SELECT COUNT(*)
FROM 'Cards'
[JOIN 'Match' ON 'Match'.id = matchId]
[WHERE playerId = ? AND
'Match'.date > ? AND
'Match'.date < ?];
```

**Goals binnen interval** Deze query vraagt de hoeveelheid goals dat de gegeven speler gescoord heeft binnen een bepaalde tijdsperiode op.

```
SELECT COUNT(*)
FROM 'Goal'
[JOIN 'Match' ON 'Match'.id = matchId]
[WHERE playerId = ? AND
'Match'.date > ? AND
'Match'.date < ?];
```

**Meeste goals** Deze query vraagt een gegeven aantal spelers op met de meeste goals.

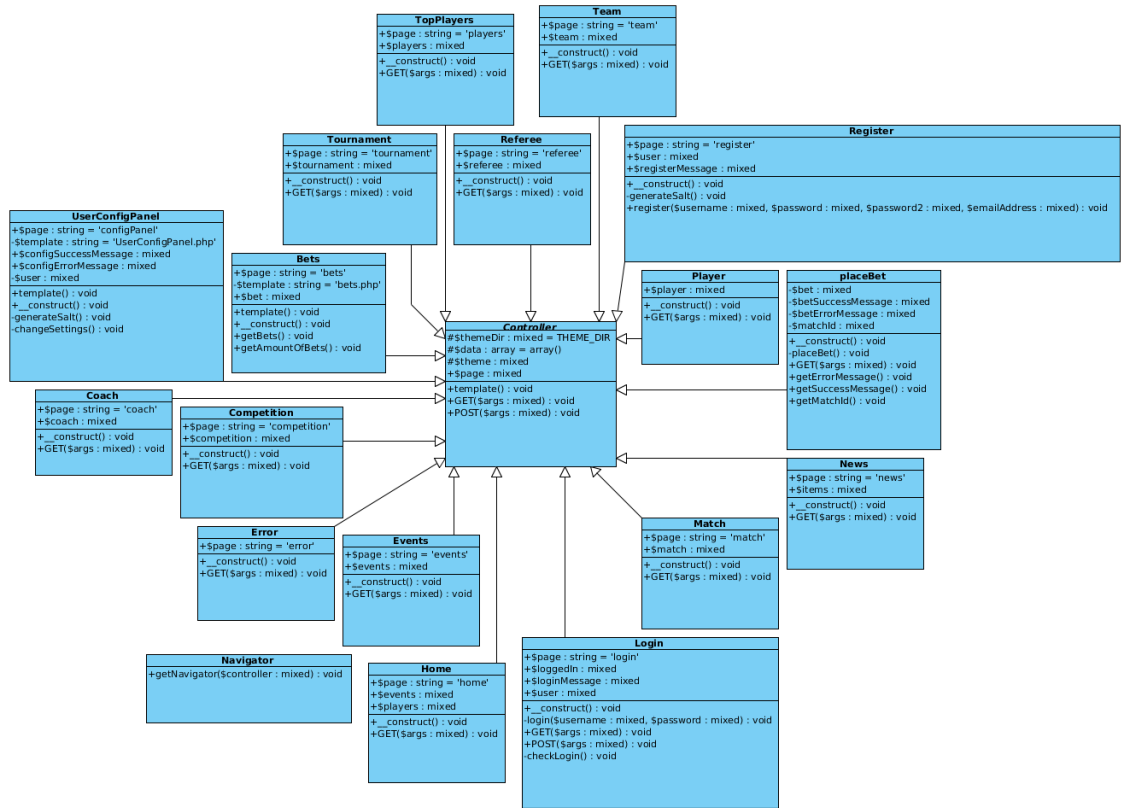
```
SELECT Player.*, COUNT(*) AS 'playedMatches'
FROM 'Player'
[JOIN 'PlaysMatchInTeam' ON 'PlaysMatchInTeam'.
playerId = 'Player.id']
[GROUP BY 'Player.id']
[ORDER BY 'playedMatches' [DESC]]
LIMIT ?, ?;
```

## 7.2 Complexe UML





## Report Project Databases: Maart



## Report Project Databases: Maart

```
Database
$link : mixed
$statements : array = array()
$statements2 : array = array()
+ _construct($db_host : mixed = DB_HOST, $db_user : mixed = DB_USER, $db_password : mixed = DB_PASS, $db_database : mixed = DB_NAME) : void
+ __destruct() : void
+ select($selector : mixed) : void
+ getStatement2($query : mixed) : void
+ getStatement($query : mixed) : void
+ getUserBets($id : mixed) : void
+ addBet($matchId : mixed, $score1 : mixed, $score2 : mixed, $userId : mixed, $amount : mixed) : void
+ getMatchFromBet($id : mixed) : void
+ getScoreFromBet($id : mixed) : void
+ getScoreFromBet($id : mixed) : void
+ getMoneyFromBet($id : mixed) : void
+ getUserName($id : mixed) : void
+ getUserPasswordHash($id : mixed) : void
+ getUserPasswordSalt($id : mixed) : void
+ getUserMail($id : mixed) : void
+ setUserMail($id : mixed, $emailAddress : mixed) : void
+ setUserSalt($id : mixed, $salt : mixed) : void
+ setUserHash($id : mixed, $hash : mixed) : void
+ doesUserNameExist($username : mixed) : void
+ doesUserExist($id : mixed) : void
+ getUser($username : mixed) : void
+ getMoney($id : mixed) : void
+ setMoney($id : mixed, $amount : mixed) : void
+ registerUser($username : mixed, $salt : mixed, $hashedPassword : mixed, $emailAddress : mixed) : void
+ getCountry($name : mixed) : void
+ getCountryById($countryId : mixed) : void
+ addCountry($name : mixed) : void
+ getCompetitionById($id : mixed) : void
+ checkCompetitionExists($id : mixed) : void
+ getCompetition($name : mixed) : void
+ addCompetition($name : mixed) : void
+ addTournament($name : mixed, $competitionId : mixed) : void
+ getTournament($name : mixed, $competitionId : mixed) : void
+ checkTournamentExists($id : mixed) : void
+ getTournamentById($id : mixed) : void
+ addReferee($firstName : mixed, $lastName : mixed, $countryId : mixed) : void
+ getReferee($firstName : mixed, $lastName : mixed, $countryId : mixed) : void
+ checkRefereeExists($id : mixed) : void
+ getRefereeById($id : mixed) : void
+ addCoach($firstName : mixed, $lastName : mixed, $countryId : mixed) : void
+ getCoach($firstName : mixed, $lastName : mixed, $countryId : mixed) : void
+ checkCoachExists($id : mixed) : void
+ getCoachById($id : mixed) : void
+ addCoaches($coachId : mixed, $steamId : mixed, $matchId : mixed) : void
+ getCoaches($coachId : mixed, $steamId : mixed, $matchId : mixed) : void
+ addTeam($name : mixed, $countryId : mixed) : void
+ getTeam($name : mixed, $countryId : mixed) : void
+ getTeamById($id : mixed) : void
+ checkTeamExists($id : mixed) : void
+ addPlayer($firstName : mixed, $lastName : mixed, $countryId : mixed, $dateOfBirth : mixed, $height : mixed, $weight : mixed, $position : mixed) : void
+ getPlayer($firstName : mixed, $lastName : mixed, $countryId : mixed) : void
+ checkPlayerExists($id : mixed) : void
+ getPlayerById($id : mixed) : void
+ doesMatchExist($id : mixed) : void
+ addGoal($playerId : mixed, $time : mixed, $matchId : mixed) : void
+ getGoal($playerId : mixed, $time : mixed, $matchId : mixed) : void
+ getGoalById($id : mixed) : void
+ addMatch($steamA : mixed, $steamB : mixed, $scoreA : mixed, $scoreB : mixed, $refereeId : mixed, $date : mixed, $tournamentId : mixed) : void
+ getMatch($steamA : mixed, $steamB : mixed, $date : mixed, $tournamentId : mixed) : void
+ removeMatch($id : mixed) : void
+ getMatchById($id : mixed) : void
+ checkMatchExists($id : mixed) : void
+ addPlayerToMatch($playerId : mixed, $matchId : mixed, $steamId : mixed, $number : mixed) : void
+ getPlaysMatchInTeam($playerId : mixed, $matchId : mixed, $steamId : mixed, $number : mixed) : void
+ getTeamInMatch($steamId : mixed, $matchId : mixed) : void
+ addPlayerToTeam($playerId : mixed, $steamId : mixed) : void
+ getPlaysIn($playerId : mixed, $steamId : mixed) : void
+ getPlayersInTeam($steamId : mixed) : void
+ removePlayersFromTeam($steamId : mixed) : void
+ addFoulCard($playerId : mixed, $matchId : mixed, $time : mixed, $color : mixed) : void
+ getFoulCard($playerId : mixed, $matchId : mixed, $time : mixed, $color : mixed) : void
+ getFoulCardById($id : mixed) : void
+ addScore($steamA : mixed, $steamB : mixed) : void
+ getScore($steamA : mixed, $steamB : mixed) : void
+ getScoreById($id : mixed) : void
+ getTotalNumberOfGoals($playerId : mixed) : void
+ getTotalNumberOfMatches($playerId : mixed) : void
+ getTotalNumberOfPlayerMatchesInterval($playerId : mixed, $min : mixed, $max : mixed) : void
+ getTotalMatchesWorByPlayerInterval($playerId : mixed, $min : mixed, $max : mixed) : void
+ getTotalCardsOfPlayerInterval($playerId : mixed, $min : mixed, $max : mixed) : void
+ getGoalsOfPlayerInterval($playerId : mixed, $min : mixed, $max : mixed) : void
+ getPlayersWithMostMatches($start : mixed, $end : mixed) : void
+ getTotalNumberOfCards($playerId : mixed) : void
+ getCoachForTeamAndMatch($steamId : mixed, $matchId : mixed) : void
+ getCoachForTeam($steamId : mixed) : void
+ getTournamentsInCompetition($competitionId : mixed) : void
+ getMatchesInTournament($tournamentId : mixed) : void
+ resultToCompetitions($result : mixed) : void
+ resultToTeams($result : mixed) : void
+ resultToReferees($result : mixed) : void
+ resultToMatches($result : mixed) : void
+ resultToGoals($result : mixed) : void
+ resultToCoaches($result : mixed) : void
+ resultToCoaches($result : mixed) : void
+ resultToPlayers($result : mixed) : void
+ resultToScores($result : mixed) : void
+ resultToCountries($result : mixed) : void
+ resultToCards($result : mixed) : void
+ resultToPlaysMatchInTeams($result : mixed) : void
+ resultToUsers($result : mixed) : void
+ getallCompetitions() : void
+ getTotalMatchesWorByPlayer($playerId : mixed) : void
+ getTotalMatchesWorByTeam($steamId : mixed) : void
+ getTotalMatchesPlayedByTeam($steamId : mixed) : void
+ getMatchDateBorder($playerId : mixed, $first : mixed) : void
+ getTotalCardsInMatch($matchId : mixed) : void
+ getUpcommingEvents($limit : mixed) : void
```