frame=tb, language=SQL, aboveskip=3mm, belowskip=3mm, showstringspaces=false, columns=flexible, basicstyle=, numbers=none, numberstyle=, keywordstyle=, commentstyle=, stringstyle=, breaklines=true, breakatwhitespace=true tabsize=3

Groep A
Rapport 1

19 maart 2014

0 Enkele opmerkingen over dit rapport

We hebben ervoor gekozen om verder te bouwen op het rapport van de vorige keer. Grote delen van de tekst zijn dus dezelfde als een maand geleden. Op die manier behandelt ook dit rapport alle functionaliteit van de website, en is het niet nodig om het vroegere rapport erbij te leggen. Volledig veranderde en nieuwe (sub)secties zijn 'Status', 'Taakverdeling', 'Voorspellingen van gebruikers', 'Extra functionaliteit' en 'Planning'. Licht aangepaste (sub)secties zijn 'Design' (zonder subsecties) en 'Database'.

1 Status

Er is heel wat data toegevoegd aan de database; die bevat momenteel honderden teams en duizenden spelers. De database moet uiteraard nog steeds verder groeien. Daarnaast is ook de mogelijkheid om als ingelogde gebruiker voorspellingen uit te brengen toegevoegd. Heel wat bugs zijn gefixt. De site staat ondertussen ook online (www.coachcenter.net), maar na een wissel in servers zijn de meeste pagina's onbereikbaar geworden. Het is dus nog steeds aangewezen de lokale versie te gebruiken. Verder zijn er geen grote features toegevoegd; het project lag na de vorige presentatie een tijdje stil door een combinatie van een aantal plotse andere taken en drukke vakanties van de groepsleden.

2 Taakverdeling

2.1 Stijn

Verzamelde heel wat meer data, vond nieuwe bronnen voor data. Refactorde grote delen van de code met oog op snelheid, efficiëntie en modulariteit. Betreft het design, is het nu een stuk eenvoudiger om aanpassingen door te voeren (vb. extra data van een speler parsen).

2.2 Kristof

Regelde een server om onze website online te krijgen, en ging aan het werk om de website online ook effectief aan de praat te krijgen. Fixte ook verschillende bugs.

2.3 Tom

Werkte aan de implementatie van het voorspellingsalgoritme.

2.4 Jakob

Werkte aan de interface om gebruikers voorspellingen uit te laten voeren.

2.5 Ruben

Zorgde voor specifieke bugfixes.

3 Design

De meesten van ons groepje hadden totaal geen ervaring met het bouwen van websites. In het begin van het project zijn de meesten dus begonnen met het doornemen van allerlei tutorials. Dit nam redelijk wat tijd in beslag, waardoor het eigenlijke project met wat vertraging begon. In eerste geval werkten we aan de backend van de website: opstellen van databaseschema, bouwen van loginsysteem, ... We waren eerst van plan om onze data te verkrijgen via de OpenFooty API. Ongeveer een week na het versturen van onze API key request begonnen we te vrezen dat we deze niet meer gingen krijgen, wat inderdaad nog steeds niet gebeurd is. We moesten dus op zoek gaan naar andere manieren om onze data te vergaren. De enige andere optie leek het gebruik van een crawler. Niemand van ons had enige ervaring met het schrijven/gebruiken van crawlers, en de crawler bleek een struikelblok. Nu werkt de crawler goed, en kunnen we de crawler vertrouwen om goeie informatie te verzamelen. Aan de implementatie van het voorspellingsalgoritme wordt gewerkt, maar dit is nog niet in de website geïntegreerd. De meeste, maar niet alle pagina's zijn ondertussen aangemaakt.

3.1 Voorspellingsalgoritme

Het voorspellingssysteem hebben wij opgesplitst in 2 essentiële delen, namelijk het voorspellen van een winnaar van een wedstrijd, en het bepalen van de te verwachten score. Factoren die wij belangrijk vonden om rekening mee te houden in het voorspellen van dit alles zijn vooral onderlinge resultaten van eerder gespeelde wedstrijden, en gemiddeldes van andere wedstrijden tegen andere ploegen. Ook wordt altijd rekening gehouden met de locatie waar

gespeeld wordt (uit of thuis).

Het voorspellen van de winnaar is natuurlijk de makkelijkste van de twee. We beginnen simpelweg door te kijken naar eerder gespeelde wedstrijden tussen deze ploegen. We werken met een puntensysteem waarbij een ploeg punten krijgt wanneer het een wedstrijd wint. Het aantal punten die ze krijgen, hangt af van de impact die we willen dat het heeft op het gehele resultaat. Hierbinnen maken we ook nog eens het onderscheid of de set-up uit/thuis dezelfde is of omgekeerd. Het eerste geval is natuurlijk het belangrijkste aangezien dit een identieke case is. Daarom zal een ploeg ook de meeste punten worden toegekend voor het winnen van zulke wedstrijden. Daarna gaan we kijken naar algemene resultaten, maar weer rekening houdende met het al dan niet uit/thuis spelen. We kijken naar alle wedstrijden die de thuisploeg thuis speelde, en de uitploeg uit speelde. Voor elke gewonnen match krijgen ze ieder weer een bepaald aantal punten (afhankelijk van de impact die deze factor moet hebben). Vervolgens kijken we naar de wedstrijden waar de thuisploeg uit speelde, en de uitploeg thuis. Wederom krijgen ze voor iedere gewonnen match een bepaald aantal punten maar weer minder dan in het vorige geval. Tot slot gaan we de punten vergelijken en op basis daarvan kunnen we de kans bepalen dat een bepaalde ploeg wint.

Het bepalen van een score is iets complexer. Dit volgt een gelijkaardig systeem als hierboven, maar met een belangrijk verschil. Hier werken we niet met het optellen van punten. We gaan altijd werken met gemiddeldes. Zo zullen we wederom eerst kijken naar identieke wedstrijden en van elke ploeg het gemiddelde aantal goals bewaren. Dan gaan we kijken naar wedstrijden met dezelfde ploegen maar op een andere locatie en wederom berekenen we het gemiddeld aantal goals gescoord door de teams. Dan gaan we weer kijken naar de algemene resultaten waarbij we wel wederom een onderscheid maken tussen uit- en thuis spelen. We berekenen dus het gemiddelde aantal goals dat de thuisploeg thuis scoort, en het gemiddelde aantal goals dat de thuisploeg uit scoort. Voor de uitploeg natuurlijk net hetzelfde. Nu zitten we met 4 gemiddeldes die we individueel vermenigvuldigen met een factor (mate van impact) en daarna optellen en delen door 4. Door dit af te ronden hebben we nu een aantal goals bepaald voor iedere ploeg. Maar hier liep het soms nog mis. Afhankelijk van de data kon de kans dat ploeg A won 90% zijn maar de score berekenen in het voordeel van ploeg B. Daarom checken we op het einde de kans op winst van een ploeg en passen de score indien nodig aan zodat deze uitkomt met de verwachte uitslag van de wedstrijd.

Ter uitbreiding zouden we graag nog meer data bepaald laten worden door het voorspellingsysteem. Dit zou dan eventuele spelers die scoren, kaarten, minuten van de goals, enzovoort zijn. Hiervoor moeten we echter eerst grondig de data die we ter beschikking hebben analyseren, en zo een geschikt

algoritme hiervoor uitdenken dat hoogstwaarschijnlijk in dezelfde trend zal liggen als die hierboven.

3.2 Voorspellingen van gebruikers

Via een formulier kunnen gebruikers voorspellingen uitbrengen. Dit formulier kan zowel bereikt worden via het gebruikersmenu bovenaan elke pagina, of via een knop op de pagina's van matchen. In het tweede geval worden teams en datum uiteraard al voor de gebruiker ingevuld. De gebruiker kan verder voorspellingen uitbrengen over de eindscore, wie het eerste doelpunt scoort, hoeveel gele kaarten elk team krijgt en hoeveel rode kaarten elk team krijgt. Dit kan later makkelijk uitgebreid worden. Voorspellen van eindscore is verplicht, de rest kan de gebruiker naar eigen wil blanco laten. Indien geen goals gescoord worden, dient er uiteraard niet opgegeven te worden wie het eerste doelpunt scoort. Om te kunnen bepalen wie de beste voorspellingen uitbracht, krijgt elke speler een puntentotaal afhankelijk van zijn voorspellingen. Dit zal afhankelijk zijn van de correctheid van voorspellingen en het aantal uitgebrachte voorspellingen. Merk op dat hoe meer dingen een gebruiker voorspelt, hoe meer punten er voor die match te verdienen zijn, maar hoe minder punten er per afzonderlijke voorspelling te verdienen zijn. Op deze manier is het invullen van extra voorspellingen een risk-rewardsituatie. We willen vermijden dat iedereen zich verplicht voelt alles in te vullen omdat ze anders geen kans maken, maar ook dat niemand extra voorspellingen invult omdat het nooit de moeite loont. Om de precieze afwegingen te bepalen zullen we nog wat moeten experimenteren. Het is momenteel mogelijk om voorspellingen uit te brengen, gemaakte voorspellingen te bekijken en de correctheid van voorspellingen van ondertussen gespeelde wedstrijden na te gaan, maar puntentellingen en vergelijkingen met andere gebruikers gebeuren nog niet. Ook worden voorspellingen van het systeem nog niet getoond.
Momenteel is het mogelijk om voorspellingen uit te voeren op reeds gespeelde matchen. In de uiteindelijke versie van de site zal dit natuurlijk niet meer nodig zijn, maar nu kan dit nog, met oog op demo'en van functionaliteit zonder een halve dag te moeten wachten en bugfixen. Momenteel worden nog geen punten geteld op basis van voorspellingen, en we zullen de gemaakte voorspellingen nog resetten eens dat wel gebeurt. Zo geeft deze optie geen oneerlijk voordeel aan early adopters van onze site.
Gemaakte voorspellingen worden opgeslagen in een specifieke tabel, 'bet', in de database. Concreet krijgt elke voorspelling een unieke id, en worden id van match en gebruiker en alle gemaakte voorspellingen bijgehouden.

4 Database

Onze database bestaat 12 tabellen met voetbalgerelateerde data en 2 usergerelateerde tabellen.

1. 'continent': Tabel voor werelddelen. Bevat een id en een naam.
2. 'country': Tabel voor landen. Bevat een id, een naam, de id van het werelddeel waarin het land ligt, en een afkorting voor de naam van het land. Die afkorting zal gebruikt worden op de website.
3. 'player': Tabel voor voetbalspelers. Bevat een id, een naam en een boolean die aangeeft of de speler geblesseerd is.
4. 'coach': Tabel voor voetbalcoaches. Bevat een id en een naam.
5. 'team': Tabel voor voetbalteams. Bevat een id, een naam, een id van het land van het team, een id van de huidige coach van het team en de FIFA score.
6. 'competition': Tabel voor voetbalcompetities. Bevat een id en een naam.
7. 'match': Tabel voor voetbalmatches. Bevat id's van thuis- en uitteam, id van de competitie en een datum.
8. 'playerPerTeam': Tabel die spelers en teams met elkaar linkt. Bevat id's van een speler en een team.
9. 'playerPerMatch': Tabel die spelers en matches met elkaar linkt. Bevat id's van een speler en een match en de tijden waarop de speler op het veld kwam en van het veld ging.
10. 'teamPerCompetition': Tabel die teams en competities met elkaar linkt. bevat id's van een team en een competitie.
11. 'goal': Tabel voor doelpunten. Bevat id van match waarin doelpunt gescoord is, tijdstip waarop, id van de speler die het doelpunt scoorde, id van het team waarnaar het punt ging en een boolean die aangeeft of het doelpunt tijdens de penaltyfase gescoord werd.
12. 'cards': Tabel voor gele en rode kaarten. Bevat een id, een id van de speler die de kaart kreeg, een id van de match waarin de kaart gegeven werd, de kleur van de kaart en de tijd waarop de kaart gegeven werd.
13. 'user': Meer uitleg bij sectie over gebruikers.
14. 'bet': Meer uitleg bij sectie over voorspellingen

4.1 ER-diagramma

Zie laatste pagina

4.2 Relational model

continent(id, name)
country(id, name, continent_id, abbreviation)
player(id, name, injured)
coach(id, name)
team(id, name, country_id, coach_id, fifascore)
competition(id, name)
match(id, hometeam_id, awayteam_id, competition_id, date)
playerPerTeam(player_id, team_id)
playerPerMatch(player_id, match_id, intime, outtime)
playerPerCompetition(team_id, competition_id)
goal(match_id, time, player_id, team_id, penaltyphase)
cards(id, player_id, color, time)
user(id, username, firstname, lastname, email, password, country, session_id, registrationcode)
bet(id, user_id, match_id, hometeam_score, awayteam_score, first_goal, hometeam_yellows, hometeam_reds, awayteam_yellows, awayteam_reds)

4.3 Constraints

– Constraints for table ‘cards‘ – ALTER TABLE ‘cards‘ ADD CONSTRAINT ‘match‘ FOREIGN KEY (‘match$_id$‘)$REFERENCES$‘$match$‘(‘id‘)$ONDELETENOACTIONONU$

– – Constraints for table ‘country‘ – ALTER TABLE ‘country‘ ADD CONSTRAINT ‘continent‘ FOREIGN KEY (‘continent$_id$‘)$REFERENCES$‘$continent$‘(‘id‘);

– – Constraints for table ‘goal‘ – ALTER TABLE ‘goal‘ ADD CONSTRAINT ‘goal$_match$‘$FOREIGNKEY$(‘$match_id$‘)$REFERENCES$‘$match$‘(‘id‘)$ONDELETENC$

– – Constraints for table ‘match‘ – ALTER TABLE ‘match‘ ADD CONSTRAINT ‘awayteam‘ FOREIGN KEY (‘awayteam$_id$‘)$REFERENCES$‘$team$‘(‘id‘)$ONDELETEC$

– – Constraints for table ‘playerPerMatch‘ – ALTER TABLE ‘playerPerMatch‘ ADD CONSTRAINT ‘player$_per_match$‘$FOREIGNKEY$(‘$match_id$‘)$REFERENCES$‘$match$

– – Constraints for table ‘playerPerTeam‘ – ALTER TABLE ‘playerPerTeam‘ ADD CONSTRAINT ‘player$_per_team$‘$FOREIGNKEY$(‘$team_id$‘)$REFERENCES$‘$team$‘(

– – Constraints for table ‘team‘ – ALTER TABLE ‘team‘ ADD CONSTRAINT ‘team$_coach$‘$FOREIGNKEY$(‘$coach_id$‘)$REFERENCES$‘$coach$‘(‘id‘)$ONDELETENO$

– – Constraints for table ‘teamPerCompetition‘ – ALTER TABLE ‘teamPerCompetition‘ ADD CONSTRAINT ‘tpc$_competition$‘$FOREIGNKEY$(‘$competition_id$‘)$REFERE$

– – Constraints for table ‘bet‘ – ALTER TABLE ‘bet‘ ADD CONSTRAINT ‘matchID‘ FOREIGN KEY (‘match$_id$‘)$REFERENCES$‘$match$‘(‘id‘)$ONDELETECAS$

5 User interface

Hoewel Laravel beschikt over een volledig uitgewerkt loginsysteem, besloten we ons eigen systeem te schrijven. Een dergelijk systeem gebruikt een database voor het opslaan van users, dus is het niet meer dan logisch dat we dit in een vak met betrekking tot databases zelf gaan programmeren. In de database bestaat een tabel 'user', met als fields 'id', 'username', 'firstname', 'lastname', 'email', 'password', 'country', 'session_id' en 'registrationcode'. De id is een auto incrementing integer die dient als key voor elke entry van de tabel. We voorzien een username, aangezien dit gebruikers een zekere vorm van anonimiteit biedt op onze website. Daarnaast is het een makkelijke manier om gebruikers op de website een unieke benaming te geven. Voornaam en familienaam zijn apart opgeslagen, zodat we in bijvoorbeeld emails gebruikers niet altijd met hun volledige naam niet hoeven aan te spreken. Het paswoord is uiteraard gehasht opgeslagen. Het hashen gebeurt via Laravel, aan de hand van bcrypt. Bcrypt is gebaseerd op het Blowfishalgoritme en heeft een salt ingebouwd, wat accounts beschermt tegen aanvallen gebruik makende van rainbow tables. Laravel biedt een functie voor het vergelijken van een ongehasht en gehasht paswoord. We vragen gebruikers ook in welk land ze wonen, zo kunnen we bijvoorbeeld bepaalde competities en nieuwsberichten een prominentere plek op de website geven afhankelijk van de gebruiker. De session_id wordt gebruikt om bij te houden of een gebruiker ingelogd is. We plaatsen een tijdelijke cookie met hetzelfde id bij de gebruiker, en kunnen dit zo nagaan. Ten slotte wordt de registratiecode gebruikt bij het nagaan van de validiteit van het emailadres van een gebruiker. De registratiecode wordt gemaild naar de gebruiker, en het account wordt pas geactiveerd wanneer deze code ingegeven wordt. Merk op dat deze functionaliteit momenteel nog niet actief is, gezien we nog niet beschikken over een mailserver.

Voor de validatie van input bij registreren (zijn verplichte velden ingevuld, staat bij emailadres wel een emailadres, is tweemaal hetzelfde paswoord ingetypt, ...) is de validatie van Laravel gebruikt. Dergelijke validatiecode ziet er heel wat beter uit voor de programmeur dan via een hoop simpele if-statements en reguliere expressies. We gebruiken de prepared statements van Laravel om sql-injecties tegen te gaan. Verder gebruiken we de Laravel Query Builder voor makkelijker schrijven van SQL queries niet, dit gebeurt zoals opgegeven met ruwe SQL queries.

Op elke webpagina verschijnt voor een niet-ingelogde gebruiker een loginknop. Het loginmenu wordt dan over de huidige pagina weergegeven. Daar is ook een "forgot password"-functionaliteit voorzien. De gebruiker geeft of zijn username of zijn emailadres in, en er wordt een email gestuurd waarmee het paswoord gereset kan worden. Wederom, deze functionaliteit is nog niet

actief. Bij een ingelogde gebruiker wordt bij elke bezochte pagina de login cookie gerefresht, zodat een gebruiker bij langdurige sessies op de site niet om de zoveel tijd opnieuw hoeft in te loggen. Later zal een ingelogde gebruiker ook een knop te zien krijgen die linkt naar zijn persoonlijk controlepaneel, maar dit moet nog geïmplementeerd worden. Aangezien alle functionaliteit die weggelegd is voor ingelogde gebruikers gebaseerd is op voorspellingen uitbrengen, wat momenteel nog niet geïmplementeerd is, ziet de website er verder voor ingelogde en niet-ingelogde gebruikers hetzelfde uit.

6 Webpagina's

De gemaakte webpagina's zijn in het Laravel PHP Framework gemaakt. Dit is een framework dat MVC hoog in het vaandel draagt en dat proberen we dus ook te respecteren. Elke pagina (View) heeft een controller tot zijn beschikking. Deze dient als tussenstuk tussen het model van de pagina waar de werkelijke queries gebeuren. Over het algemeen is dit wat er gebeurt:

Er wordt een pagina opgevraagd in de url. (Bv. /public/team?id=1). In de route-file wordt gedefinieerd welke controller er opgeroepen moet worden. Deze controller zal informatie opvragen van het relevante model en zal deze info doorspelen naar de view van de pagina.

Aanvankelijk was het werken met het framework niet eenvoudig maar eens het wat duidelijker werd zagen we de waarde ervan in. Het design hebben we grotendeels te danken aan Twitter Bootstrap. Een website ziet er zo relatief gemakkelijk goed uit.

De webpagina's zijn nog volop in development alsook kan het design hier en daar ook nog wel veranderen.

6.1 Notifications

Ook is er in ons systeem een uitgebreid notificatiesysteem aanwezig. Wanneer een gebruiker ingelogd is, zal er bovenaan naast het user menu een nieuw item tevoorschijn komen (icoon: ster). Op het moment dat er een notificatie is voor de gebruiker wordt die hier getoond. Zonder op het item te klikken kan je al zien hoeveel ongelezen notificaties je hebt door het cijfer in superscript dat naast het sterretje staat. Als de gebruiker hierop klikt komt er een dropdown menu tevoorschijn met de meest recente notificaties. Deze notificaties linken naar de relevante pagina.
Om dit mogelijk te maken is er een notificatiesysteem geschreven dat gemakkelijk uitbreidbaar is. Elke notificatie heeft een specifiek formaat: er is altijd

een actor, een subject, een object en een type. Aan de hand van deze informatie kan er dan een gepaste boodschap worden getoond.

1. Actor: de uitvoerende partij
2. Subject: de persoon voor wie de notificatie bestemd is
3. Object: de specifieke informatie over de notificatie
4. Type: om duidelijk te maken welk voor type notificatie we hebben (Bijvoorbeeld een usergroup invite)

6.1.1 Implementatie

Elke notificatie komt dus overeen met een rij van de notifications table. Wanneer een notificatie wordt opgeslagen (Notifications::saveNotification) komt er niet veel bij zien: gewoon letterlijk de kolommen invullen. Bij het opvragen van een notificatie (Notifications::getNotification) komt er wat meer bij kijken: in het resultaat wordt de informatie van het object erbij gezet. Wanneer er op de notificatie wordt geklikt wordt de redirect methode van de NotificationController opgeroepen die naargelang het type notificatie de gebruiker naar de juiste pagina redirect.

6.1.2 Soorten notifications

Usergroup invites Een gebruiker nodigt een andere gebruiker uit om een usergroup te joinen.

1. Actor: de gebruiker die de persoon uitnodigt
2. Subject: de uitgenodigde gebruiker
3. Object: de userGroupInvite

4. Type: 1, Notifications::INVITE$_{U}SER_{G}ROUP$ De notificatie zal linken naar de gebruiker zijn profiel waar die de invitatie zal kunnen accepteren of weigeren.

   Remind user bets Het systeem herinnert de gebruikers om te betten op matchen waar ze dat nog niet op hebben gedaan. Omdat het systeem dit verstuurt zullen we de actor en subject dezelfde personen nemen.

   1. Actor: de gebruiker
   2. Subject: de gebruiker
   3. Object: NULL (Geen verdere info nodig)

4. Type: 2, Notifications::REMIND$_{U}SER_{B}ETS$ De notificatie zal linken naar de upcoming matches page, waar de gebruiker gemakkelijk kan zien op welke matches hij/zij nog niet op heeft gebet.

   Nieuwe discussie in usergroup Wanneer een gebruiker een nieuwe discussie start in een usergroup zullen alle members van die groep een notificatie hierover krijgen.

   1. Actor: de gebruiker die de discussie heeft aangemaakt
   2. Subject: een gebruiker die lid is van de groep
   3. Object: de discussie

4. Type: 3, Notifications::NEW$_{D}ISCUSSION$ De notificatie zal linken naar de discussie.

   Nieuwe reply in discussie Als een gebruiker een bericht post in de discussie zullen alle deelnemers van die discussie hierover genotified worden.

   1. Actor: de gebruiker die een nieuwe reply heeft gemaakt
   2. Subject: deelnemer aan de discussie
   3. Object: discussie

4. Type: 4, Notifications::NEW$_{M}ESSASGE$ De notificatie linkt naar de discussie.

6.2 Work in progress

6.2.1 Home page

Bestaat uit 3 tabellen, een news foto feed en een wereldkaart die de landen kleurt in functie van hun WK-ranking. De 3 tabellen zullen data bevatten over de laatst gespeelde matchen, de opkomende matchen en de WK-ranking. De foto newsfeed is een RSS feed van FIFA die we met SimplePie hebben geparsed.
Op de wereldkaart krijgt elk land een tint groen toegekend; hoe hoger hun WK-ranking, hoe donkerder het groen. Later zul je op elk apart land kunnen klikken om op de pagina van het nationale team terecht te komen. Merk op dat het Verenigd Koninkrijk wegens beperkingen van de gebruikte Google GeoCharts-API niet ingekleurd is; zij hebben vier "nationale" teams (Engeland, Wales, Noord-Ierland, Schotland). Om dit op te lossen, wordt bij klikken op het VK ingezoomd, waarna de vier delen apart in de juiste kleur verschijnen.

6.2.2 Search bar / search results

Rechtsbovenaan alle pagina's staat een searchbar waarin je een speler/match/team kunt invullen en die de beschikbare info presenteert. Als je op de links doorklikt word je doorgestuurd naar de desbetreffende pagina.

6.2.3 Player Page

Deze pagina bevat informatie over een bepaalde speler.
In de PlayerController wordt volgende info opgevraagd van de Player en Team models: $playerObj = Player :: getPlayer$(playerName)[0]; $playerTeam = Team :: getTeambyPlayerID$(playerObj-¿id)[0]; $playerText = Player :: getPlayerText$(playerName); $playerImageURL = Player :: getPlayerImageURL$(playerName); $goals = Player :: goals$(playerObj-¿id); $cards = Player :: cards$(playerObj-¿id);

Deze informatie wordt dan vervolgens gebruikt om weer te geven op de webpagina. Wat wel opgemerkt moet worden is dat momenteel de playerText en de playerImageURL van wikipedia worden gehaald en niet in onze database zitten. Misschien doen we dit beter wel?

6.2.4 Team Page

Deze pagina bevat informatie over een bepaald team, zoals de spelers, matchen e.d.

In de TeamController wordt de benodigde informatie uit de models gehaald. Dit gebeurt in meerdere verschillende functies. Hieronder een voorbeeld. public function index($teamID$)teamObj = Team::getTeamByID($teamID$)[0];teamImageURL = Team::getTeamImageURL($teamObj->name$);

return View::make('team.team', compact('teamObj', 'teamImageURL'))-¿with('title', $teamObj->name$);

Deze informatie wordt vervolgens gebruikt om weer te geven op de webpagina. De spelers en dergelijke worden opgevraagd in andere methoden van de TeamController.

6.2.5 Team Players page

Deze pagina is een onderdeel van de bijhorende Team pagina. Door middel van Ajax wordt dit op de pagina geladen zonder de pagina te hoeven herladen.
In de PlayersController wordt volgende info opgevraagd van het Team model:
public function index($playerName$)playerObj = Player::getPlayer($playerName$)[0];playerTeam = Team::getTeambyPlayerID($playerObj->id$)[0];playerText = Player::getPlayerText($playerName$) = Player::getPlayerImageURL($playerName$);goals = Player::goals($playerObj->id$);cards = Player::cards($playerObj->id$);

return View::make('player.player', compact('playerObj', 'playerTeam', 'playerText', 'playerImageURL', 'goals', 'cards'))-¿with('title', $playerName$);

6.2.6 Match page

De match page bevat informatie van een specifieke wedstrijd. Hierbij wordt er dus info getoond over de match: de eindstand, wie er gescoord heeft, wie er een kaart heeft gekregen en dergelijke. Uiteraard kan er weer op meerdere manieren doorgeklikt worden door op de speler/teams te klikken en dergelijke.

6.2.7 Betting pages

Het is momenteel enkel mogelijk om individueel pronostieken in te vullen; ondersteuning om in groep te spelen volgt nog. Er is een formulier om je pronostiek in te vullen en door te sturen en een pagina met een overzicht van alle gemaakte gokken. Later komt er onder andere nog een pagina waar je een 'gokwedstrijd' kunt aanmaken met andere mensen. Ongetwijfeld zullen er nog extra pagina's bijkomen.

6.3 Optimalisatie van reeds bestaande pages

Een deel van de reeds geschreven code is niet altijd even proper. Daarom zal een deel ook een beetje herschreven worden; een voorbeeld hiervan zijn de controllers. In plaats van allemaal verschillende argumenten mee te geven aan een view is het properder om alles eerst in een array te steken en die dan gewoon mee te geven. Dit maakt het makkelijker om later argumenten toe te voegen.

In concreto: Eerder dit: function index($matchID$)data['match'] = Match::get($matchID$);data['goalshometeam'] = Match::goals($matchID$,data['match']-¿hometeam$_id$);data['cardshometeam'] = Match::cards($matchID$,data['match']-¿hometeam$_id$);data['goalsawayteam'] = Match::goals($matchID$,data['match']-¿awayteam$_id$);data['cardsawayteam'] = Match::cards($matchID$,data['match']-¿awayteam$_id$);

return View::make('match.match',$data) -> with('title','Match');

in plaats van: public function index($playerName$)playerObj = Player::getPlayer($playerName$)[0] = Team::getTeambyPlayerID($playerObj$-> id)[0];playerText = Player::getPlayerText($playerName$) = Player::getPlayerImageURL($playerName$);goals = Player::goals($playerObj$-> id);cards = Player::cards($playerObj$-> id);

return View::make('player.player', compact('playerObj', 'playerTeam', 'playerText', 'playerImageURL', 'goals', 'cards'))-¿with('title', $playerName$);

7 Extra functionaliteit

De meeste extra functionaliteit die gesuggereerd wordt in de opgave is aanwezig of is ondersteuning voor ingebouwd. Er is heel wat data aanwezig over spelers, er is relevant nieuws te lezen, er is (ondersteuning voor) data uit andere competities, ...
Over verdere extra functionaliteiten hebben we nog niet echt nagedacht, het is nu in de eerste plaats belangrijk dat de verplichte functionaliteit er is. Daarna zou nog tijd over moeten zijn voor extra functionaliteit.

8 Planning

Het project heeft niet erg veel vorderingen geboekt sinds de laatste evaluatie. Om te vermijden dat dit nogmaals zou gebeuren, zullen we dit keer geen pauze invoeren na de evaluatie, maar blijven doorwerken. Zo zal het werk zich niet opstapelen bij de deadline, wat een ramp zou zijn gezien die in de examens valt. De taakverdeling verliep al goed, nu moeten we enkel zorgen dat het individuele werk ook effectief gebeurt. Bij de eerste evaluatie waren we wel al redelijk ver gevorderd in het project, wat betekent dat we nu toch geen onoverkomelijke achterstand hebben opgelopen. De resterende verplichte functionaliteit hebben we onder elkaar verdeeld, eens die aanwezig en stabiel is gaan we over op zo veel mogelijk extra functionaliteit (zolang het nuttig is voor de gebruiker)

8.1 Stijn

Mogelijk om e-mails te verzenden inbouwen, refactoring van code

8.2 Tom

Voorspellingen van systeem integreren met voorspellingen van user

8.3 Ruben

Ondersteuning voor groepen van gebruikers om samen te werken bij het voorspellen.

8.4 Jakob

Grafische weergave van data voor gebruikers implementeren

8.5 Kristof

Automatisch updaten van database implementeren

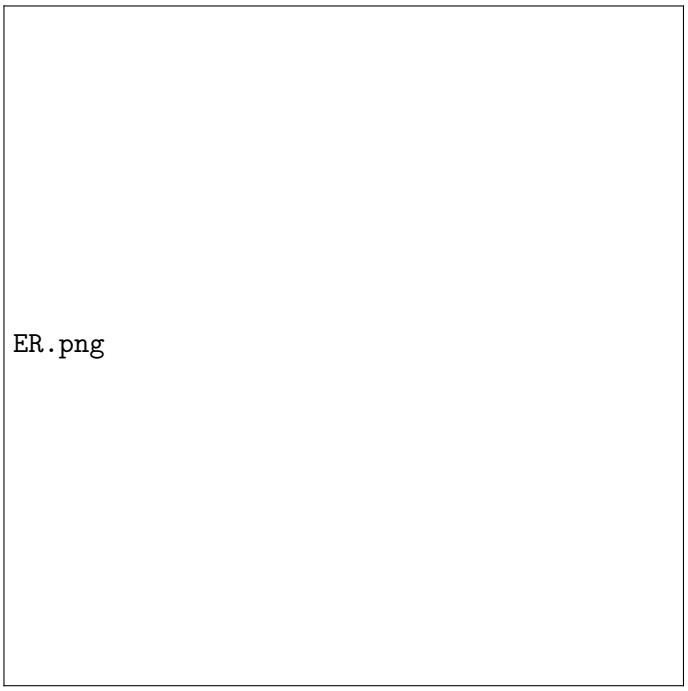

Figure 1: Onze database in ER-model