DELCOM INTERNATIONAL B.V.

Upgrade front-end

Technisch Ontwerp

Dave Wolters

Dit document bevat het technisch ontwerp. Het technisch ontwerp bevat de technische randvoorwaarden waaraan de te realiseren front-end moet voldoen.

Upgrade front-end

Technisch Ontwerp

Afstudeerperiode	02-04-2012 – 31-08-2012
Auteur	Dave Wolters Amperestraat 47 3553 CK Utrecht Tel. 06-14022329 davewolters@gmail.com
Studentnummer	1513626
Bedrijfsbegeleider	*Delcom International B.V.* Jan Duijnhouwer Dussendijk 14 4271 XL Dussen Tel. 0162-580258 jan@dmssolutions.eu
Docentbegeleider	Nini Salet *Hogeschool Utrecht*, Faculteit Natuur en Techniek Opleiding Information Engineering Nijenoord 1 3552 AS Utrecht 088-4818861 nini.salet@hu.nl
Versie	0.2 18-06-2012
Status	Afgerond

INLEIDING

Dit document bevat het technisch ontwerp (TO) voor de front-end webapplicatie van Delcom Management Systems. Het technisch ontwerp is opgesteld voorgaande de realisatie van de front-end.

Het TO beschrijft de technische randvoorwaarden waaraan de front-end moet voldoen. Het Functioneel Ontwerp heeft de functionaliteiten beschreven welke de front-end moet bevatten. Het Technisch Ontwerp gaat hierop verder door de technische randvoorwaarden te bepalen voor de front-end. In dit document komen de technieken, de opzet van de front-end en de database aan bod.

Dit technisch ontwerp moet antwoord geven op de volgende vraag: Hoe moet de vernieuwde front-end werken?

Het eerste hoofdstuk bevat de gebruikte technieken van de website. Hierin worden de programmeertalen, het framework en de browserondersteuning toegelicht.

Hoofdstuk 2 bevat de opzet van de website. Hierin worden de rechten van de gebruikers uitgewerkt, worden de templates besproken en wordt de architectuur van de front-end nader verklaard.

Hoofdstuk 3 gaat over de database. De database wordt gebruikt om alle data te bewaren en uit te lezen. Dit hoofdstuk bevat een databasemodel.

Dit document is opgesteld voor- en in samenwerking met de opdrachtgever.

VERSIEBEHEER

Versie	Datum	Auteur	Opmerking
0.1	13-06-2012	Dave Wolters	Eerste versie van het Technisch Ontwerp
0.2	18-06-2012	Dave Wolters	Tweede versie van het Technisch Ontwerp

INHOUDSOPGAVE

LIJST MET FIGUREN

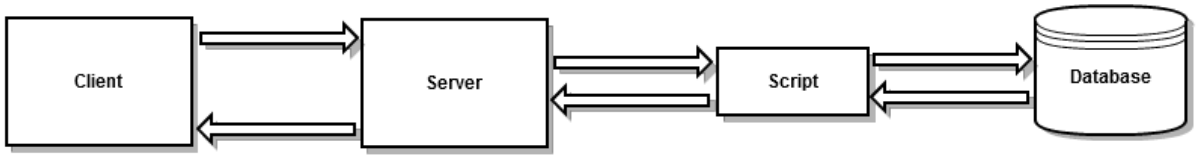
LIJST MET TABELLEN

1 DE TECHNIEKEN

In dit hoofdstuk worden de gebruikte technieken besproken. Dit wordt gedaan aan de hand van de programmeertalen en het framework. Afsluitend wordt er bepaald welke browsers en welke schermresoluties er ondersteund worden in de vernieuwde front-end en hoe dit bereikt wordt.

1.1 PROGRAMMEERTALEN

De front-end is een dynamische website. Bij een dynamische website is sprake van een website met een database-koppeling. Een dynamische website maakt gebruikt van client-serverarchitectuur (*zie figuur 1).*



FIGUUR 1 CLIENT-SERVER ARCHITECTUUR

Om de dynamische website tot stand te laten komen is een aantal programmeertalen nodig. Deze worden hieronder kort besproken.

HTML (Clientside)
Voor de opmaak van de webpagina's wordt gebruik gemaakt van HTML5. HTML5 maakt het mogelijk om tekst, afbeeldingen, multimedia weer te geven in een browser. Er is gekozen voor HTML5 omdat dit de laatste versie is van HTML. HTML5 biedt meer functionaliteiten dan HTML4.01. Zo is de compatibiliteit met verschillende platformen (desktop, mobiel, tablet) verbetert. Daarnaast is HTML5 met laden sneller t.o.v. HTML 4.01.

CSS (Clientside)
HTML beschrijft de inhoud van de site, met CSS de vorm (Groenendaal, 2009). Feitelijk houdt het in dat CSS de vormgeving van een website bepaald. Door gebruik te maken van losse CSS-stijlbladen kan een volledige website vormgegeven worden door één CSS-bestand. Er is gekozen om gebruik te maken van CSS3. CSS3 is de opvolger van CSS2 en heeft meer functionaliteiten tot zijn beschikking. Zo zijn er minder afbeeldingen nodig omdat CSS3 nu meer ondersteuning beidt op gebied van lettertypes, teksteffecten, patronen, schaduwrandjes en animaties. Voorheen werden deze gebieden opgevangen door javascript of met afbeeldingen maar met CSS3 is dit niet altijd meer nodig.

PHP (Serverside)
PHP is een serverside scripttaal waarmee krachtige dynamische webtoepassingen kunnen worden gerealiseerd, zoals webwinkels, forums, gastenboeken, nieuwssites, intelligente websites, gepersonaliseerde sites, voorraadlijsten, productenoverzichten, enzovoort (Groenendaal, 2009).
Er is voor PHP gekozen omdat PHP gratis is, platform-onafhankelijk, breed gedragen wordt door de community en geschikt is om te communiceren met databases door middel van MySQL.

Javascript (Clientside en serverside)
Javascript wordt gebruikt om webpagina's dynamischer en interactiever te maken. Voor de front-end wordt er alleen gebruik gemaakt van clientside-javascript. Hiermee wordt het mogelijk om delen van de website meer interactie mee te geven. Zo is het mogelijk om elementen een animatie of klikhandeling mee te geven. Een javascript-library die gebruikt gaat worden is JQuery.

AJAX (Clientside)
AJAX staat voor Asynchronous Javascript and XML en werd op 18 februari 2005 door Jesse James Garrett geïntroduceerd in zijn artikel 'Ajax: A New Approach to Web Applications' (Groenendaal, 2009). AJAX is zelf geen taal maar een techniek die gebruikt maakt van XHTML, CSS, DOM, XML, XSLT, XMLHttpRequest en Javascript. AJAX wordt gebruik om delen op de webpagina's te kunnen verversen zonder dat de gehele pagina hoeft te worden herladen.

1.2 HET FRAMEWORK

Vanuit de analyse is bekend geworden dat de opdrachtgever de techniek wil vernieuwen van de front-end (zie *hoofdstuk 2.2 Doelstellingen uit het Functioneel Ontwerp)*. Daarbij kwam aan het licht dat een MVC-model de meest ideale architectuur was doordat het logica, opmaak en stijl volledig scheidt van elkaar. Dit zorgt ervoor dat onderhoud makkelijker wordt en de mogelijkheid ontstaat om iedere klant een eigen vormgeving aan te beiden.

Om het MVC-model eenvoudig te hanteren is er gekozen voor een framework. Een framework is de basisstructuur voor het bouwen van web-applicaties. Een framework heeft een community en professionals die zich bezig houden met de ontwikkeling van het framework. Hierdoor is de code getest en verbeterd door vele mensen. Tevens zorgt een framework ervoor dat er sneller ontwikkeld kan worden. De programmeur hoeft zich niet bezig te houden met het grondwerk en kan meteen met de functionaliteiten aan de slag. Tevens verplicht het framework de programmeur om aan de hand van het gebruikte model te programmeren.

CodeIgniter
Voor de front-end ontwikkeling is er gekozen voor Codeigniter. (CodeIgniter: an open source Web Application Framework that helps you write PHP programs, 2012) Er zijn vele frameworks maar Codeigniter blinkt uit in de beschikbare documentatie. Dit zorgt ervoor dat Codeigniter eenvoudig te leren valt. Tevens is er een grote community waardoor eventuele vragen snel beantwoord kunnen worden via forums. Voor de afstudeerder was dit een van de belangrijkste overwegingen om te kiezen voor Codeigniter gezien zijn geringe ervaring met frameworks.

Naast de beschikbare documentatie moest het framework ook gemakkelijk te installeren zijn. Delcom hanteert een systeem van duplicaten. Door middel van kopiëren van de website worden nieuwe klanten aangesloten op het systeem. De server is niet altijd in beheer van Delcom, daardoor moet er geen serverinstelling gewijzigd hoeven te worden. Codeigniter heeft geen serverwijzigingen nodig om te werken.

Daarnaast is Codeigniter snel in het laden van pagina's, klein van formaat (2.1mb) en kost het niets.

Nadelen zijn er ook, zo is Codeigniter van zichzelf nogal compact in functionaliteiten. Dit gat wordt opgevuld door library's van derden, waardoor er een grotere afhankelijkheid ontstaat van derden. Tevens is Codeigniter minder strikt in het opdwingen van een programmeerwijze, waardoor verschillende programmeurs verschillende programmeerstijlen hanteren. Met overdracht moet hiermee rekening gehouden worden door gebruik te maken van veel commentaar in de code.

1.3 ONDERSTEUNING WEBBROWSERS EN SCHERMRESOLUTIES

Een andere doelstelling (zie *hoofdstuk 2.2 Doelstellingen uit het Functioneel Ontwerp)* uit de analyse was het ondersteunen van mobiele platformen, zoals tablets en smartphones. Om niet voor elk platform een nieuwe website te bouwen is ervoor gekozen om een *responsive webdesign* te gebruiken. Responsive webdesign houdt in dat de website zich aanpast naar de resolutie van de gebruiker d.m.v. CSS. Responsive webdesigns zijn niet de heilige graal in het ontwikkelen van websites, want mobiele gebruikers hebben andere eisen aan een website dan desktop gebruikers. De voor- en nadelen van responsive webdesigns zijn als volgt (Rdz, 2011):

Voordelen	Nadelen
Eén website voor alle platformen	Langere ontwikkelperiode
Eén URL en één ervaring voor alle platformen	Cross-browser compatibiliteit
Nooit een horizontale scrol balk	Laadtijden voor mobiel
Afbeeldingen en media worden getoond op een correcte manier	Gebruikerseisen verschillen per platform
Mobiel internet gebruik groeit explosief.	Advertenties zijn moeilijk verkleinbaar

TABEL 1 VOOR- EN NADELEN RESPONSIVE WEBDESIGN

Geboden ondersteuning per platform voor de front-end:

Platform	Resolutie	Browsers
Mobiel	800 x 480	Safari mobile, Dolphin, Opera
Tablet	1024 x 600	Safari mobile, Dolphin HD, Opera
Desktop	1280 x 1024	Firefox 4+, Google Chrome 11+, Safari 5+, Internet Explorer 8+

TABEL 2 GEBODEN ONDERSTEUNING PER PLATFORM

2 OPZET FRONT-END

In dit hoofdstuk wordt de opzet van de front-end besproken. Dit gebeurt aan de hand van de gebruikersrechten, de globale architectuur van de front-end, de bestandsstructuur en de templates.

2.1 DE GEBRUIKERSRECHTEN

De front-end bevat een inlogfunctionaliteit waardoor de gebruikers gescheiden worden in twee groepen, sporters en personal trainers. Elk van de groepen hebben eigen pagina's, hieronder wordt weergegeven, in *tabel 3 en 4,* welke rechten de gebruikers hebben. Deze tabellen komen uit *hoofdstuk 4.2 en 4.3* van het *Functioneel Ontwerp*.

CRUD staat voor create, read, update en delete. De dikgedrukte, onderstreepte letters komen terug in de applicatie. Ofwel voor de functie 'Bezoeken' wordt alleen de read toepassing uitgevoerd.

De Sporters

Functionaliteit	**Webpagina's**
Inschrijven	Inschrijfpagina
Inloggen	Homepagina (landingpage)
Profiel (**R U**)	Profielpagina (R, U), MyPage (R)
Reserveren (**C R D**)	Reserveringspagina (C, R, D), Totaal overzicht reserveringen (R, D)
Transacties (**C R**)	Betalingspagina (C), Totaal overzicht betalingen (R)
Abonnementen (**C R U D**)	Abonnementenpagina (C, R, U, D)
Trainingsschema's (**R**)	Prestatiepagina (R)
Prestaties (**R**)	Prestatiepagina (R)
Bezoeken (**R**)	Bezoekpagina (R)

TABEL 3 WEBPAGINA'S SPORTERS

De personal trainers

Functionaliteit	**Webpagina's**
Profiel (**R U**)	Profielpagina (R, U), MyPage (R)
Trainingsschema's (**C R U D**)	Trainingsschemapagina (C, R, U, D)
Prestaties (**C R U**)	Prestatiepagina (C, R, U, D), MyPage (R)
Cadeaubon (**C R D**)	Cadeaubonpagina (C, R, D)
Bezoekers (**R**)	Bezoekerspagina (R), MyPage (R)
Lesvaardigheden (**R U**)	Lesvaardigheid-pagina (R, U)
Medewerker uren (**R**)	Medewerker uren-pagina (R)

TABEL 4 WEBPAGINA'S PERSONAL TRAINERS

2.2 DE ARCHITECTUUR

In paragraaf 1.1 zijn de programmeertalen al aan bod gekomen en in *paragraaf 1.2* is het framework behandeld. Maar hoe komen deze twee elementen samen? Dit wordt verklaard in deze paragraaf. Eerst wordt het MVC-model toegelicht en vervolgens wordt deze samengevoegd met de programmeertalen in een overzicht om een beeld te vormen van de architectuur van de applicatie.

2.2.1 HET MVC-MODEL

Model-view-controller *(of MVC) is een ontwerppatroon ("design pattern") dat het ontwerp van complexe toepassingen opdeelt in drie eenheden met verschillende verantwoordelijkheden: datamodel (model), datapresentatie (view) en applicatielogica (controller).* (Model-view-controller-model, 2012)

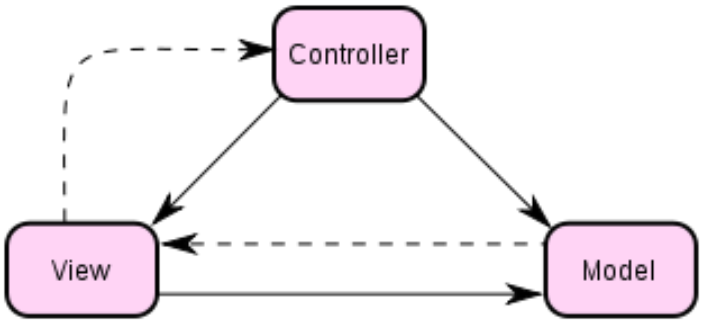
Het scheiden van de code bevordert de leesbaarheid en herbruikbaarheid van code. Hierdoor wordt de website makkelijker aanpasbaar voor zowel de klanten als voor Delcom. Hieronder wordt elke onderdeel van het MVC kort toegelicht:

Model
Definieert de representatie van de informatie waarmee de applicatie werkt. Aan ruwe gegevens wordt betekenis gegeven door relaties tussen data en logica toe te voegen. De daadwerkelijke opslag van data wordt gedaan met behulp van een persistent opslagmedium, zoals een database. De applicatie zal gegevens die gebruikt worden in het model, ophalen en wegschrijven van en naar de dataopslag via een datalaag. De datalaag is niet per se een onderdeel van het MVC-patroon.

View
Informatie wordt weergegeven via de View. Userinterface-elementen zullen gedefinieerd zijn in dit onderdeel. De view doet geen verwerking (zoals berekeningen, controles, ...) van de gegevens die getoond worden.

Controller
De controller verwerkt en reageert op events, die meestal het gevolg zijn van handelingen van de gebruiker.

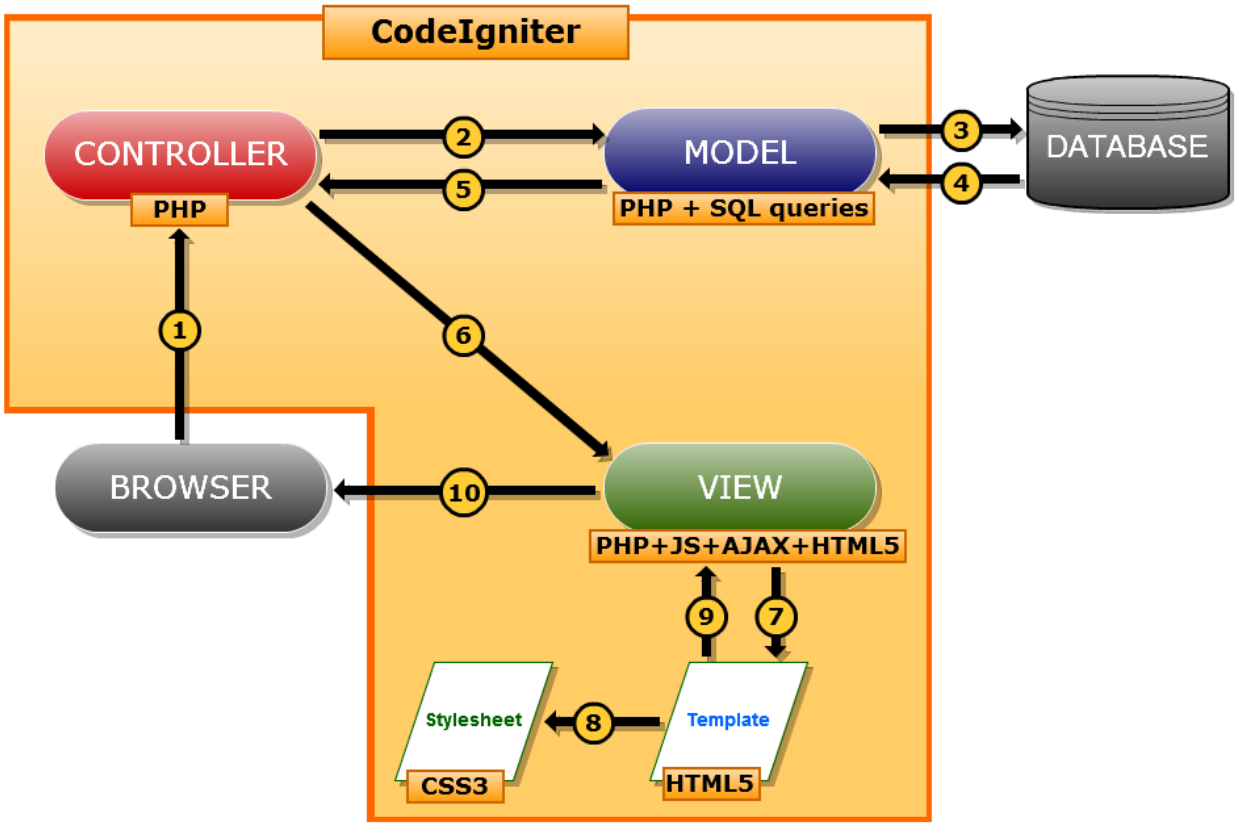
(Model-view-controller-model - Wikipedia, 2012)



FIGUUR 2 HET MVC-MODEL

2.2.2 OVERZICHT ARCHITECTUUR

In *hoofdstuk 2.2.1* werd duidelijk dat het MVC-model gehanteerd zou worden. Nu de programmeertalen en framework ook bekend zijn, kan er een overzicht getoond worden van de opzet van de front-end. In het overzicht wordt duidelijk hoe de elementen samenkomen en welke stappen er door het systeem genomen worden. Het oranje vlak bevat het gehele Codeigniter framework en de database is een MySQL database *(zie Figuur 3).*



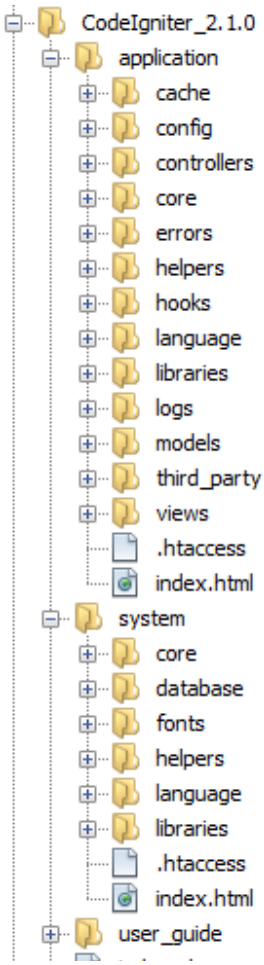
FIGUUR 3 OVERZICHT ARCHITECTUUR

Uitleg van de doorlopen stappen in Figuur 3

1 De browser stuurt een verzoek naar de Controller.
2 De controller maakt verbinding met het model en geeft een opdracht.
3 Het model interpreteert de opdracht en stuurt een query naar de database.
4 De database verwerkt de query en stuurt het antwoord terug naar het model.
5 Het model ontvangt het antwoord en verwerkt de data naar de controller.
6 De controller stuurt de data naar de view.
7 De view haalt de template op voor de opmaak van de data.
8 De template haalt de bijbehorende stijlsheet op voor de vorm van de opmaak.
9 De view bouwt een pagina op met de bijbehorende data, template en stijlsheet.
10 De view presenteert de pagina aan de browser.

2.3 DE BESTANDSSTRUCTUUR

Het MVC-model heeft dus drie eenheden, namelijk model, view, controller. Elk van deze eenheden heeft een eigen map binnen het Codeigniter framework. Het Codeigniter framework hanteert de volgende indeling voor de bestanden (*Zie Figuur 4).* Hieronder worden de belangrijkste mappen kort besproken.



FIGUUR 4 CODEIGNITER BESTANDSSTRUCTUUR

De application map bevat de belangrijkste bestanden van de front-end.

In de config map worden de configuratie-bestanden opgeslagen van de database, de autoload libaries, routers en helpers.

In de controllers map worden de controller-bestanden van de functionaliteiten opgeslagen. Deze verzorgen het verkeer tussen de models en de views. Elke controller-class verlengt de standaard CI_controller class.

In de models map worden alle model-bestanden opgeslagen. De models verzorgen de communicatie tussen de webapplicatie en de database. Elke model-class verlengt de standaard CI_model class.

In de views map worden alle view-bestanden opgeslagen. De views presenteren de data.

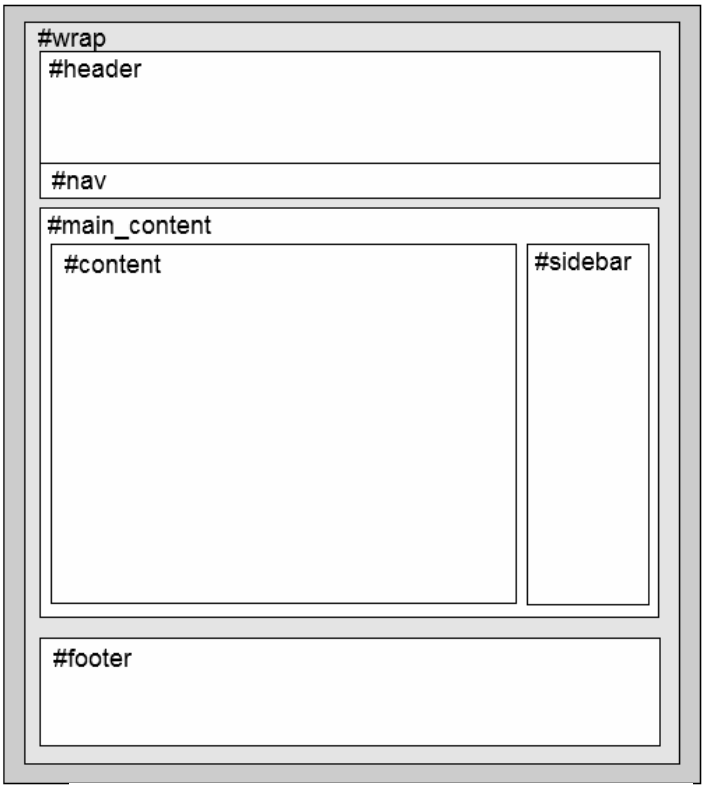
In de libararies map worden alle externe-libaries opgeslagen. Op deze manier blijven de libaries gescheiden van de applicatie.

De system map bevat de kern van Codeigniter. Hierin hoeven geen veranderingen plaats te vinden. Deze map is niet publiekelijk toegankelijk om zo de integriteit van de kern van Codeigniter te waarborgen

De user_guide map bevat uitgebreide documentatie over CodeIgniter.

2.4 DE TEMPLATE

De template zorgt voor de opmaak van de pagina's. Als basis voor de template wordt de *HTML5 boilerplate* (HTML5 Boilerplate - Een robuuste bakplaat voor HTML5-lekkers., 2012) gebruikt. Een boilerplate zorgt ervoor dat een pagina er in elke browser hetzelfde uit ziet. Dit wordt gedaan d.m.v. CSS3, HTML5 en Javascript. Hierdoor is het mogelijk om HTML5 te gebruiken zonder dat er compatibiliteitproblemen ontstaan met Internet Explorer 6. Tevens zijn er een aantal best-practises opgenomen in de code. Zo wordt bijvoorbeeld de code automatisch gecomprimeerd voor het verbeteren van de laadsnelheid. De boilerplate zorgt ervoor dat de afstudeerder minder tijd kwijt is aan bug fixing en eerder kan beginnen met het bouwen van de vormgeving en functionaliteiten.



FIGUUR 5 TEMPLATE

De nieuwe HTML pagina lay-out is als volgt;

- Wrap
  - Header
    - Navigatie
  - Main_content
    - Content
    - Sidebar
  - Footer

Figuur 5 laat zien hoe de pagina lay-out er in een wireframe uit ziet.

3 DE DATABASE

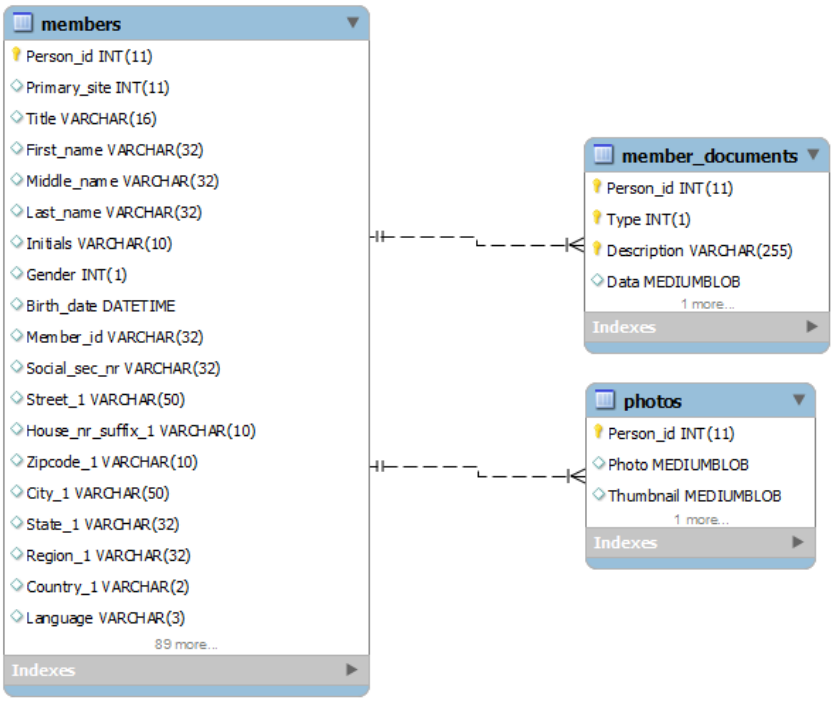
In dit hoofdstuk wordt de database besproken. Dit wordt gedaan aan de hand van een databasemodel. De database wordt niet ontworpen door de afstudeerder en de afstudeerder mag ook geen veranderingen in database aanbrengen. Behalve in de tabel web_options waarin alle websites opties worden opgeslagen van het admin-panel van de front-end.

3.1 HET DATABASEMODEL

Aangezien de database niet ontworpen hoeft te worden zal deze alleen beschreven worden. Alleen de belangrijkste tabellen van de website worden hier uitgelicht. In totaal zijn er 77 tabellen, waardoor een volledig databasemodel onoverzichtelijk zou zijn.

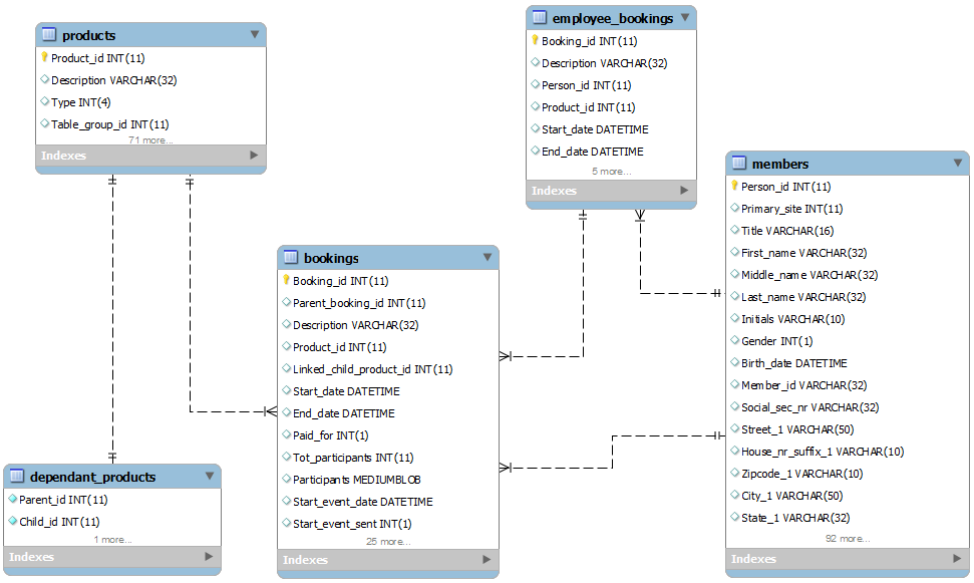
Sporters en personal trainers

Figuur 6 toont de klassen die betrekking hebben tot de sporters en personal trainers. Sporters en personal trainers worden opgeslagen in dezelfde tabel, namelijk **Members**. Aan de hand van de kolom *Trainer* wordt bepaald of een member een sporter of een medewerker is. Tevens heeft de **Members** klasse twee subklassen namelijk **Members_documents** en **Photo's** om de documenten en foto's van de gebruikers te bewaren.



FIGUUR 6 DATABASEMODEL: SPORTERS EN PERSONAL TRAINERS

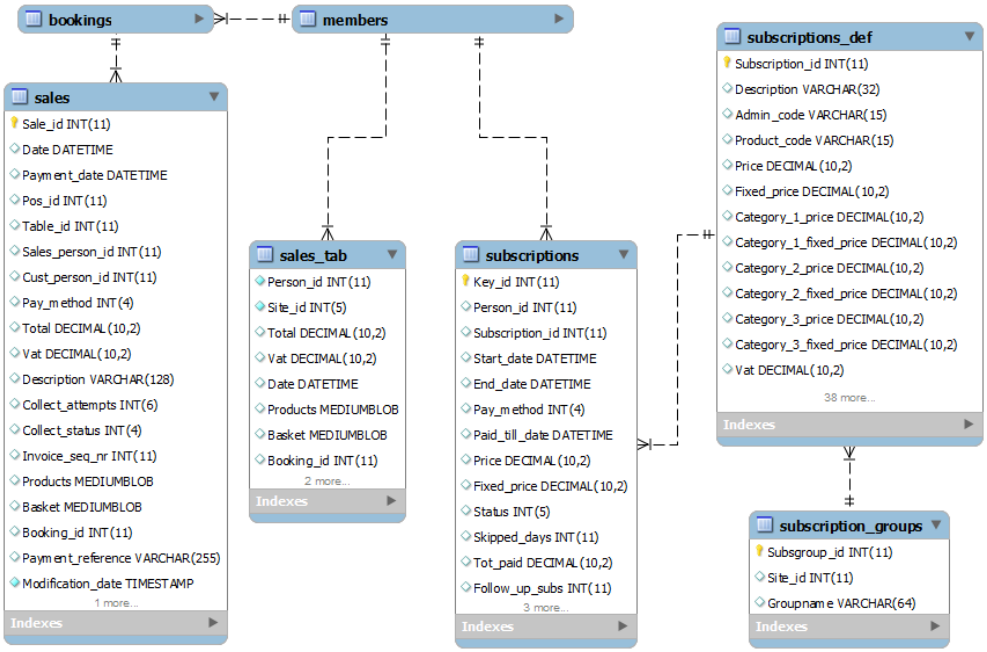
Reserveren



FIGUUR 7 DATABASEMODEL: RESERVEREN

Figuur 7 toont de klassen die betrekking hebben tot de reserveringen. Reserveringen worden opgeslagen in de klasse **Bookings**. **Bookings** maakt gebruik van **Members** om de deelnemers op te slaan en indien er een medewerker nodig is voor de activiteit wordt deze d.m.v. **Employee_bookings** opgeslagen. Dit is nodig aangezien members ook de medewerkers bevat, d.m.v. de kolom *Trainer* wordt in **Members** bepaald of een member een sporter of een medewerker is.

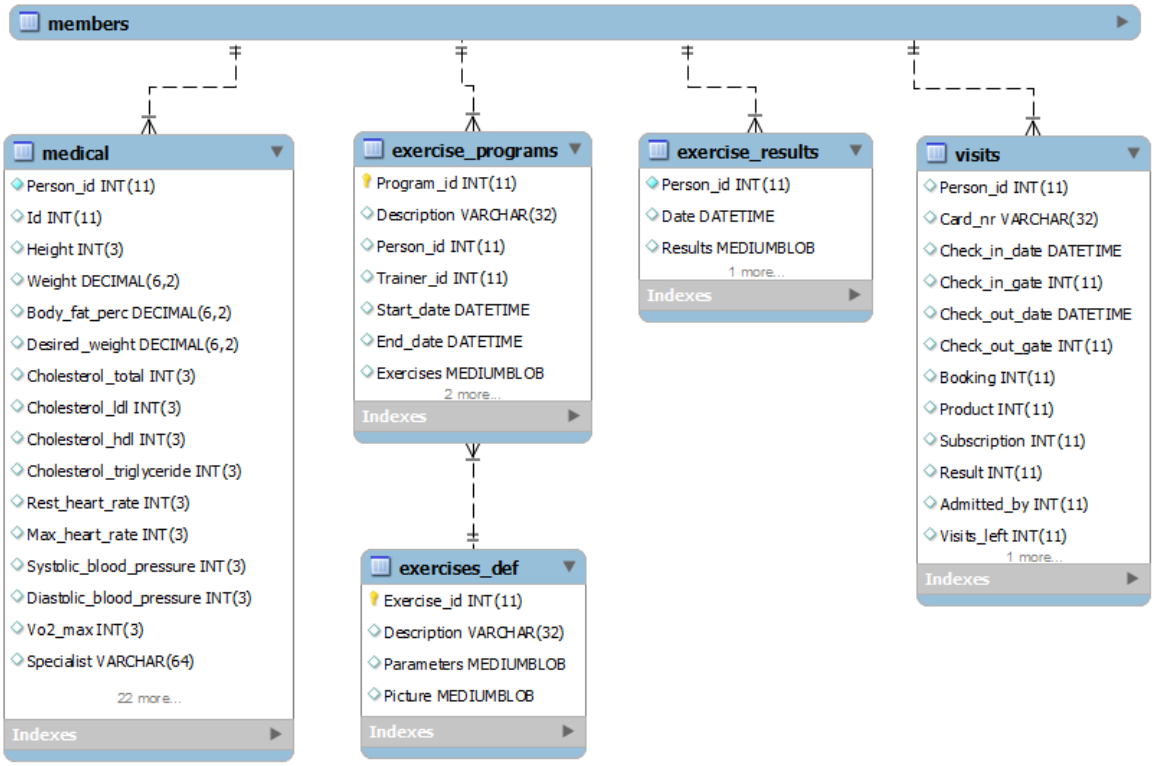
Abonnementen



FIGUUR 8 DATABASEMODEL: ABONNEMENTEN

Figuur 8 toont de klassen die betrekking hebben tot de abonnementen. Abonnementen bestaan uit één hoofdklasse en twee subklassen namelijk **Subscriptions**, **Subscriptions_def** en **Subscription_groups**. In **Subscriptions** worden de abonnementen gekoppeld aan de klanten. De betaalgegevens van de abonnementen worden bij de abonnementen opgeslagen in **Subscriptions**. De betalingen van reserveringen (**Bookings**) worden opgeslagen in **Sales**.

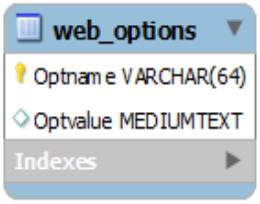
Prestaties en Trainingen



FIGUUR 9 DATABASEMODEL: PRESTATIES EN TRAININGEN

Figuur 9 toont de klassen die betrekking hebben tot de prestaties en trainingen van members. De bezoeken van een Member worden opgeslagen in de **Visits** tabel samen met de gegevens van het bezoek. **Medical** houdt alle medische gegevens bij van een member. **Exercise_programs** bestaat uit **Exercises_def** opgeslagen in een MEDIUMBLOB. De resultaten van de oefeningen van een Member worden opgeslagen in **Exercise_results**.

De website heeft een eigen tabel (**Web_options,** *zie figuur 10*) om gegevens in op te slaan. Dit kunnen CRM-gegevens zijn maar ook de betalingsopties worden in deze tabel opgeslagen. De tabel **Web_options** heeft twee velden namelijk *Optname* en *Optvalue*. De afstudeerder heeft deze tabel tot zijn beschikking.



FIGUUR 10 DATABASEMODEL: WEBOPTIES

CONCLUSIE

Het technische ontwerp geeft antwoord op de vraag: Hoe moet de vernieuwde front-end werken?

Om deze vraag te beantwoorden is dit document uiteengezet in technieken, opzet front-end en de database. Op deze manier worden alle aspecten belicht van de front-end op architectuurniveau, maar niet op functionaliteitenniveau.

De techniek is verdeeld in programmeertalen en framework. De volgende programmeertalen worden gebruikt:

- HTML5
- CSS3
- PHP
- Javascript
- AJAX

Als framework is er gekozen voor CodeIgniter. Er is voor CodeIgniter gekozen omdat dit framework het best gedocumenteerd is, geen serverinstellingen vereist en breed gedragen wordt door de community. Tevens hanteert CodeIgniter het MVC-model.

Als ontwerpmodel is gekozen voor het **M**odel-**V**iew-**C**ontroller-model. Het voordeel van een MVC-model is dat het datamodel (model), datapresentatie (view) en applicatielogica (controller) van elkaar gescheiden zijn. Hierdoor kunnen er verschillende presentaties zijn van de data zonder dat het datamodel aangepast hoeft te worden. Tevens wordt het, door het strikt scheiden van code en gebruik maken van CSS3, mogelijk om klanten een eigen 'look 'n feel' aan te bieden. Iets wat in de huidige front-end niet mogelijk is.

Naast een framework is er ook gekozen om een templatesysteem te hanteren. De template zorgt voor de opmaak van de pagina's. Als basis voor de template wordt de *HTML5 boilerplate* gebruikt. Een boilerplate zorgt ervoor dat een pagina er in elke browser hetzelfde uit ziet. Dit wordt gedaan d.m.v. CSS3, HTML5 en Javascript.

Het datamodel van het MVC-model zit aangesloten op een MySQL database. De database slaat alle gegevens op en biedt deze ook aan. Deze database bestaat uit 77 tabellen waarvan de afstudeerder er één tot zijn beschikking krijgt. De overige tabellen worden gebruikt in de lokale applicatie en website en kunnen niet zomaar gewijzigd worden.

Het overzicht van *paragraaf 2.2.2* laat zien hoe al deze elementen samenkomen en het geheel vormen van de front-end applicatie.

BRONNENLIJST

CodeIgniter: an open source Web Application Framework that helps you write PHP programs. (2012). Opgeroepen op Mei 01, 2012, van CodeIgniter: http://codeigniter.com/

Model-view-controller-model. (2012, mei 3). Opgeroepen op mei 20, 2012, van Wikipedia: http://nl.wikipedia.org/wiki/Model-view-controller-model

Groenendaal, H. v. (2009). *Webdesign Van Concept Tot Realisatie.* Den Haag: Academic Service.

Rdz, P. (2011, augustus 12). *The Pros and Cons of Responsive Web Design*. Opgeroepen op juni 1, 2012, van The Pam: http://thepam.blogspot.nl/2011/08/pros-and-cons-of-responsive-web-design.html