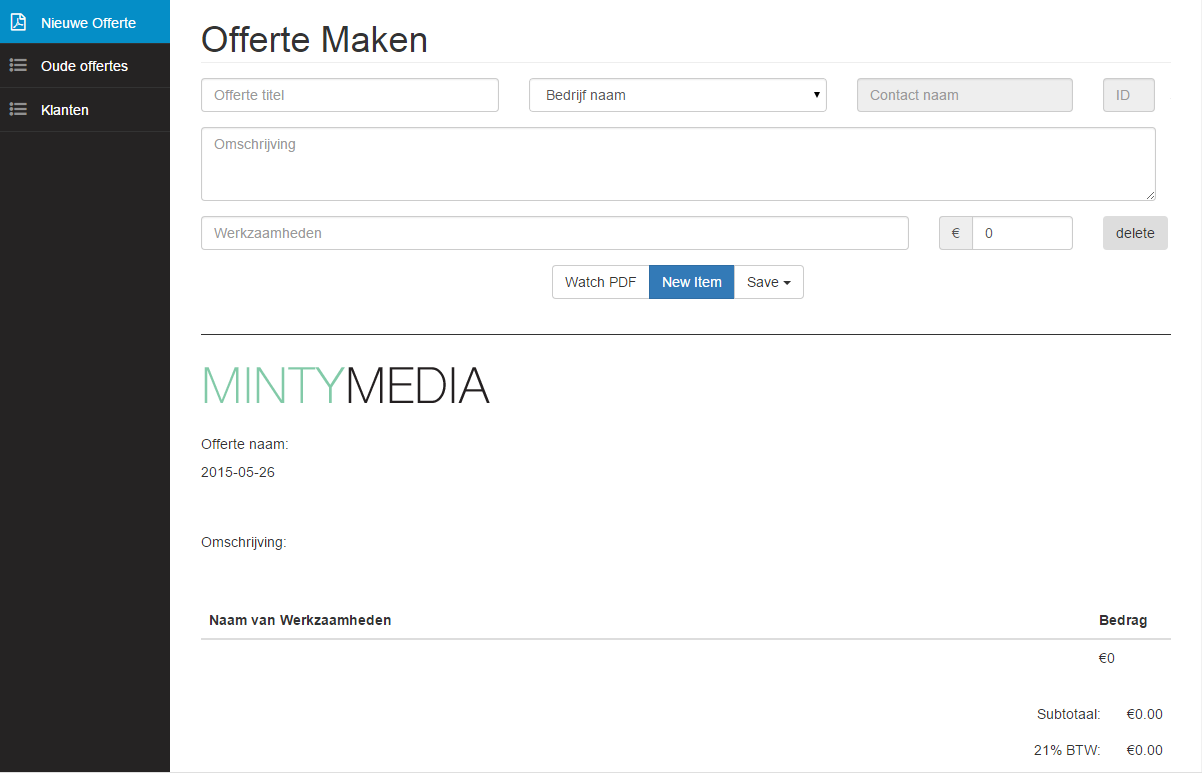
PVB verslag

het offerte systeem



Naam: Geoffrey van Driessel

Klas: A3B

Opleiding: ICT Applicatie ontwikkelaar

Stage: Minty Media

Project: Het offerte systeem

Inhoudsopgave

[Inleiding 3](#_Toc420407270)

[Plan van aanpak 4](#_Toc420407271)

[Brackets 4](#_Toc420407272)

[PHP Designer 5](#_Toc420407273)

[XAMPP 6](#_Toc420407274)

[Frameworks 6](#_Toc420407275)

[Eisen en wensen 6](#_Toc420407276)

[Werkzaamheden 9](#_Toc420407277)

[Het draadmodel 9](#_Toc420407278)

[Het begin 11](#_Toc420407279)

[JSPDF 12](#_Toc420407280)

[JSPDF table 13](#_Toc420407281)

[Chrome Print 13](#_Toc420407282)

[De favicon 16](#_Toc420407283)

[PDF Titel 16](#_Toc420407284)

[Side Menu 18](#_Toc420407285)

[Github 20](#_Toc420407286)

[Database 21](#_Toc420407287)

[Back end 24](#_Toc420407288)

[Views & routing 27](#_Toc420407289)

[AngularJS request 29](#_Toc420407290)

[PHP Back end 31](#_Toc420407291)

[AngularJS verwerking 33](#_Toc420407292)

[NAS 35](#_Toc420407293)

[Test 36](#_Toc420407294)

[Conclusie 37](#_Toc420407295)

[Het resultaat 37](#_Toc420407296)

# Inleiding

Dit verslag gaat over het offerte systeem dat ik heb ontwikkeld bij mijn stagebedrijf ‘Minty Media’. In dit verslag worden alle werkzaamheden die ik heb verricht omtrent het offerte systeem besproken. Ook probeer ik alle kerntaken van de pvb te behandelen.

De opdracht kreeg ik van Job Dekker, mijn praktijkbegeleider. Hij wilde graag een nieuw offertesysteem voor in de browser waarmee hij eenvoudig gegevens in kan vullen en meteen de offerte als PDF kan opslaan of printen.

Hij gebruikte eerst een betaalde dienst genaamd [www.offertesturen.nl](http://www.offertesturen.nl), die dienst vond hij erg handig in het gebruik, maar bij die dienst moest hij per offerte/factuur 40 cent betalen; dat liep erg op. Ook zochten wij nog een project voor mij om zelfstandig aan te werken, dus kwamen wij op het idee om een eigen offertesysteem te bouwen.

Plan van aanpak

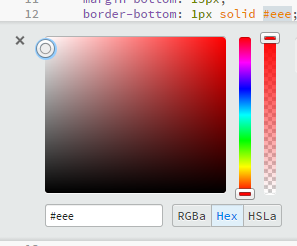
Ik begon met het creëren van een ontwikkelomgeving, ik gebruikte eerst mijn laptop maar ze boden een desktop aan om op te ontwikkelen. De computer staat beschikbaar voor werknemers van Minty Media. Er stond een clean Windows 8 installatie op, ik kon dus gelijk aan de slag en heb een paar handige tools geïnstalleerd.

* Brackets
* PHP Designer
* XAMPP

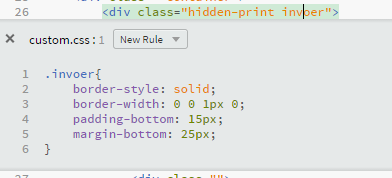
Voor dit project ben ik compleet zelf verantwoordelijk, ik ben verantwoordelijk voor het design, het interactieve gedeelte en de gegevens verzameling van de web applicatie.

### Brackets

Brackets is een moderne en opensource ‘tekst editor’ met handige features voornamelijk voor CSS en HTML. Het ondersteunt bijvoorbeeld ‘quik edit’ wat heel handig is als je bezig bent met kleuren bijvoorbeeld**.** Het is namelijk mogelijk om in de code een kleur te selecteren, dan de quik edit te openen en zo kan je heel handig een nieuwe kleur selecteren uit een soort kleurenpalet:



Ook ondersteunt het programma bijvoorbeeld live preview, dat spreekt een beetje voor zich, de functie opent het HTML bestand in de browser en geeft veranderingen in de code gelijk door aan de browser. Inline editors is een feature die ik persoonlijk niet zo veel gebruik maar het kan best handig zijn dit houdt in dat als je een element in je HTML zou selecteren dat je gelijk de CSS kan aanpassen met één druk op de knop:



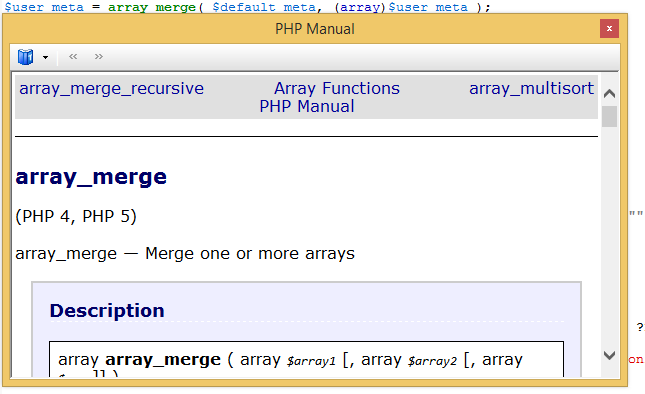
Support voor LESS en SCSS is ook aanwezig wat ook erg handig is, ik gebruik geen LESS en SCSS in dit project omdat ik het daar te klein voor vind. Ik gebruik Brackets voor het coderen met HTML, CSS, Javascript en soms zelfs PHP.

### PHP Designer

PHP Designer biedt een handige omgeving voor het werken met PHP. Het is PHP IDE en PHP editor in één, wat erg handig is met het testen van applicaties. Vooral voor grote websites gemaakt met PHP is dit een must-have. De PHP IDE houd eigenlijk in dat je de PHP code kan uitvoeren wat normaal gesproken niet mogelijk zou zijn zonder een server.

Ik gebruik PHP Designer alleen voor het aanpassen van PHP code omdat ik vind dat brackets veel handiger is voor CSS, HTML en Javascript.

PHP Designer heeft een paar heel handige features voor het editen van PHP code, bijvoorbeeld een goede en handige syntax high-lighting en documentatie over de standaard functies van PHP, *the PHP manual*. Je kan bijvoorbeeld, als je niet weet wat een functie doet, de functie selecteren en dan zie je op een pop-up schermpje een duidelijk uitleg met voorbeelden.



### XAMPP

XAMPP is een veel gebruikt programma, XAMPP creëert een lokale server ook wel een localhost genoemd. Zo kan je een server gebruiken in het lokale netwerk, dit is erg handig bij het testen en creëren van server-side applicaties. Zo kan eerst de applicatie lokaal ontwikkelt worden, en dan getest voordat het online staat.

Ook biedt het programma ondersteuning voor een database. Wat erg handig is in samenwerking met een server-side taal zoals PHP. Zo kan ik met XAMPP een SQL database opzetten en met PHP kan ik deze database aanspreken.

### Frameworks

Bij deze applicatie wil ik graag gebruik maken van een aantal frameworks waar ik al bekend mee ben. Deze frameworks gebruik ik om de product efficiency te verhogen.

Dit zijn een aantal frameworks die ik gebruik:

* [Bootstrap](http://getbootstrap.com/); Bootstrap is een CSS framework met veel voorgeschreven CSS klassen die men kan gebruiken. Het design ga ik voornamelijk maken met Bootstrap. Met Bootstrap timmer je in weinig tijd een moderne web applicatie in elkaar.
* [AngularJS](https://angularjs.org/); Angularjs is een nieuwe Javascript framework gemaakt door Google. Het is een lightweight en snelle framework en word gebruikt voor het maken van dynamische en interactieve web applicaties.
* [jsPDF](https://github.com/MrRio/jsPDF) en [jsPDFTablePlugin](https://github.com/Prashanth-Nelli/jsPdfTablePlugin); voor het omzetten van HTML naar een PDF bestand.

### Eisen en wensen

De eisen en wensen in globale lijn zijn als volgt: er moet een web applicatie gecreëerd worden die lokaal in het netwerk bereikbaar moet zijn. De web applicatie bevat een aantal pagina’s: nieuwe offerte, oude offertes en klanten.

Op de nieuwe offertepagina moet een formulier komen te staan waarbij offerte gegevens ingevuld kunnen worden. De gegevens worden verwerkt in een A4 die uitgeprint kan worden of opgeslagen als PDF door één druk op de knop.

Het formulier moet voor de volgende gegevens invulvelden hebben:

* Offerte titel
* Klant naam
* Omschrijving
* Datum
* Werkzaamheden + bedrag

Naast de invulvelden van de werkzaamheden en het bedrag komt er een knop om een nieuwe regel toe te voegen. De toegevoegde regel bevat twee lege invulvelden voor een nieuwe werkzaamheid en het bijbehorende bedrag. Ook zal er per werkzaamheid een verwijder knop achter de regel komen, zodat je gemakkelijk een regel kan verwijderen.

Alle bedrijfsnamen worden opgehaald uit een database, deze komen in een uitrol menu waaruit de gebruiker een bedrijf kan selecteren. Als de gebruiker een bedrijf heeft geselecteerd worden alle bedrijfsgegevens opgehaald en in de offerte verwerkt.

Er komt een download knop waarbij alle ingevulde gegevens worden verwerkt in A4 formaat en deze als PDF word gedownload. Er komt een ‘save’ knop, die slaat de offerte op in de database met alle ingevulde gegevens. Zo kan deze later weer opgehaald worden.

Onder aan de pagina komt het voorbeeld van de A4 met alle ingevulde gegevens.

De oude offertes pagina bevat een tabel/lijst met alle oude offertes, opgehaald uit een database. De tabel bevat een aantal gegevens van de offertes:

* Offerte ID
* Offerte naam
* Offerte omschrijving
* Offerte datum
* Bedrijfsnaam

Als laatste kolom in de tabel komt een knop om de offerte te bekijken. Als er op de bekijk knop wordt gedrukt dan worden alle gegevens van de offerte opgehaald. Daarna worden deze gegevens ingeladen in de nieuwe offerte pagina.

De klantenpagina bestaat ook uit een tabel/lijst waarbij alle klanten worden opgehaald uit de database met alle klantgegevens. Eventueel komt hier ook een toevoeg of verwijder knop bij. De tabel bevat de volgende klant gegevens:

* ID
* Contact naam
* bedrijfsnaam
* Adres
* Postcode
* Stad

Deze klantgegevens worden geïmporteerd uit het vorige offerte systeem: www.offertesturen.nl.

Er komt een modern en duidelijke navigatie balk aan de linker zijkant. Dit menu zal uit knoppen bestaan, de knoppen bestaan weer uit een icoon en een titel. Dit menu zal ervoor zorgen dat de gebruiker door de verschillende pagina’s kan navigeren. Dit menu zal handmatig gemaakt worden.

Het offertesysteem zal ook met een gegevensverzameling(database) gekoppeld worden. Zo kunnen bepaalde gegevens opgeslagen en hergebruikt worden. De database zal gebruikt worden voor:

* Het opslaan van offertes met alle ingevulde gegevens
* Het ophalen van offertes met alle ingevulde gegevens
* Het opslaan van klant gegevens
* Het ophalen van klant gegevens

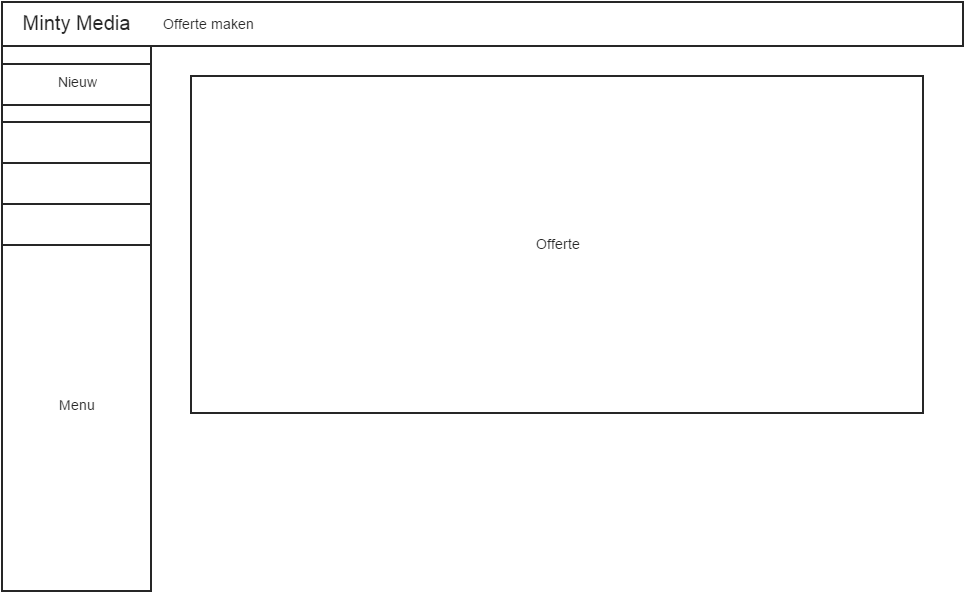
Ook zal de database moeten worden ingericht met de juist tabel structuur, de tabellen moeten bijvoorbeeld aan elkaar gekoppeld worden.

# Werkzaamheden

In dit hoofdstuk bespreek ik alle verrichte werkzaamheden omtrent het project ‘Offerte systeem’. Hierbij probeer ik zo duidelijk en discreet mogelijk te zijn.

### Het draadmodel

Na overleg met de ‘product-owner’(Job) gehad te hebben, ben ik een draadmodel gaan maken. Ik heb hem zo simpel en duidelijk mogelijk gehouden, na een aantal verbeteringen ziet het draadmodel er uiteindelijk zo uit:

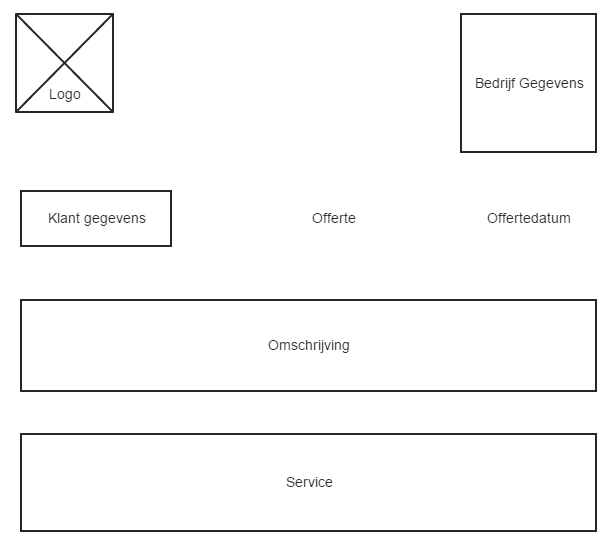


Dit is voor de web interface zelf, duidelijke blokken met wat waar moet. Het menu aan de zijkant met linkjes naar verschillende onderdelen van de web applicatie. En aan de bovenkant eventueel een soort balk met de titel van de pagina. En als content de offerte zelf.

De content bevat eigenlijk alles wat met de offerte te maken heeft, waaronder de invoer velden om de offerte te maken en het A4 voorbeeld van de offerte. Als de gebruiker op een link klikt in het menu zou alleen de content veranderen. In de content komen dus de verschillende web pagina’s.

Het uiteindelijke doel van de web applicatie is natuurlijk het maken van een PDF waarin de gemaakte offerte staat, die je zowel kan downloaden als printen.

De offerte wordt een PDF in A4 formaat waarbij al standaard gegevens in staan zoals het logo en de bedrijfsgegevens van de Job. Dat zijn gegevens die er altijd in moeten komen op een standaard plek dus die kan ik gewoon ‘hard-coded’ er in zetten.



Het draadmodel voor de uiteindelijk offerte die de web applicatie creëert.

Er kunnen natuurlijk nog altijd wijzigingen verricht word, er zal wel wat bij komen en/of geschoven worden met elementen. Het gaat voornamelijk om de basis van de offerte die duidelijk word met het draadmodel.

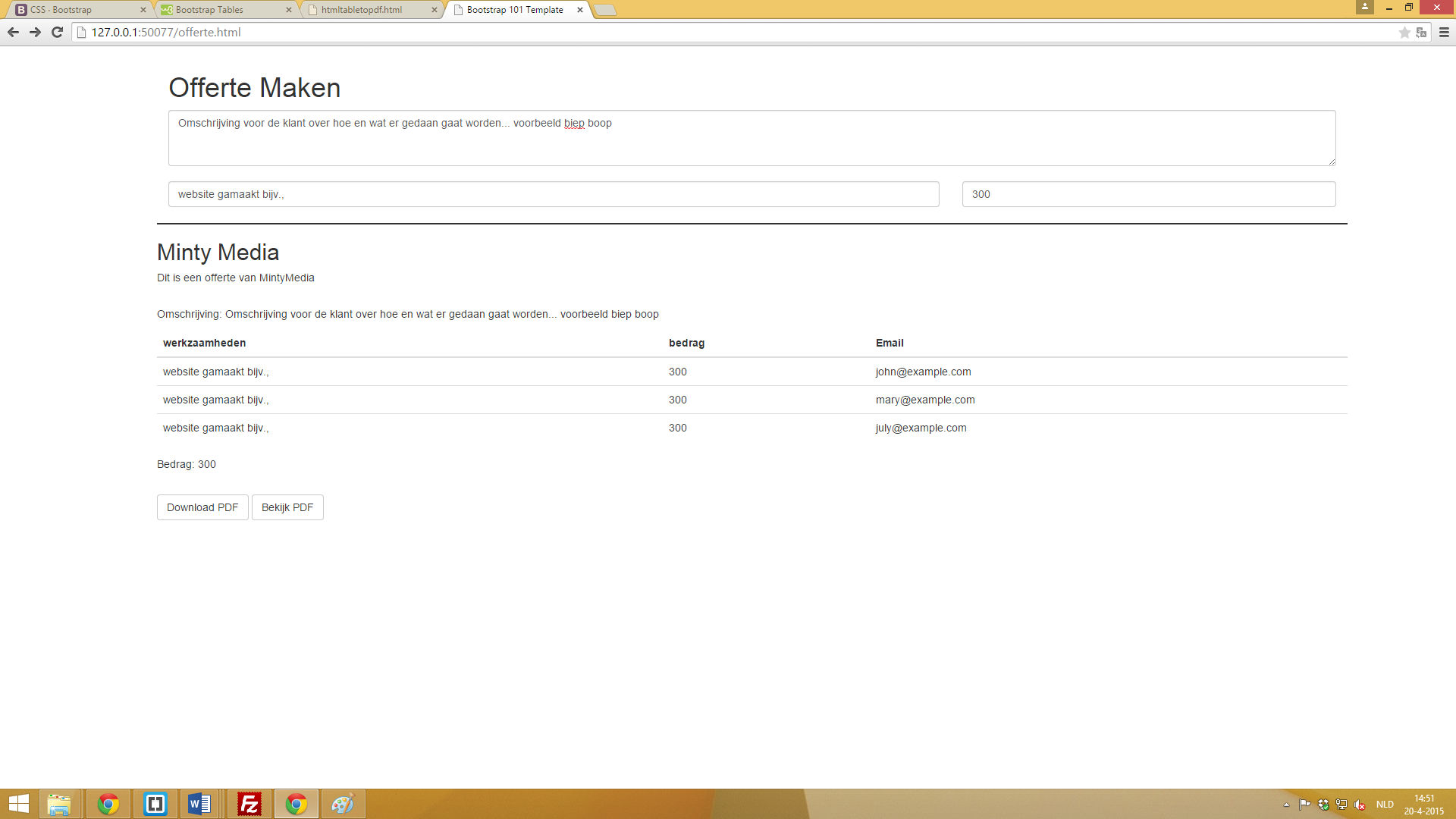
Nadat ik de web applicatie van het vorige offerte systeem had geanalyseerd leek het me handig om het in Javascript te maken. Ik had namelijk al eventjes met een Javascript framework genaamd “AngularJS” gewerkt en ik vond het erg goed bij elkaar passen. Zo is AngularJS bijvoorbeeld heel erg handig in het uitlezen van invul velden en dat sluit dus goed aan bij dit project. Daarnaast spreekt AngularJS me ook erg aan om dat ik het interessant vind en me er verder in wil verdiepen.

Toch twijfelde ik eerst nog tussen AngularJS en PHP omdat ik meer ervaring had met PHP. Ook is PHP erg handig met het uitlezen van invul velden en het en die gegevens te verwerken in een database.

Maar ik besloot toch om met AngularJS te beginnen, omdat AngularJS toch een aantal voordelen bood. Waaronder het client-side verwerken van de data in de invul velden, dit leg ik later verder uit. De rest van de voordelen worden later in dit verslag besproken.

### Het begin

Ik begon eerst met het maken van een web applicatie in Bootstrap en AngularJS, een simpel design en simpele functionaliteit. Met Bootstrap had ik efficiënt een web applicatie ontwikkeld het ziet er goed en modern uit met in verhouding weinig moeite. Ik ben begonnen met het ontwikkelen van een lege template die ik op het Bootstrap framework ontwikkeld hebt, op basis van dit framework ben ik de web applicatie gaan bouwen.

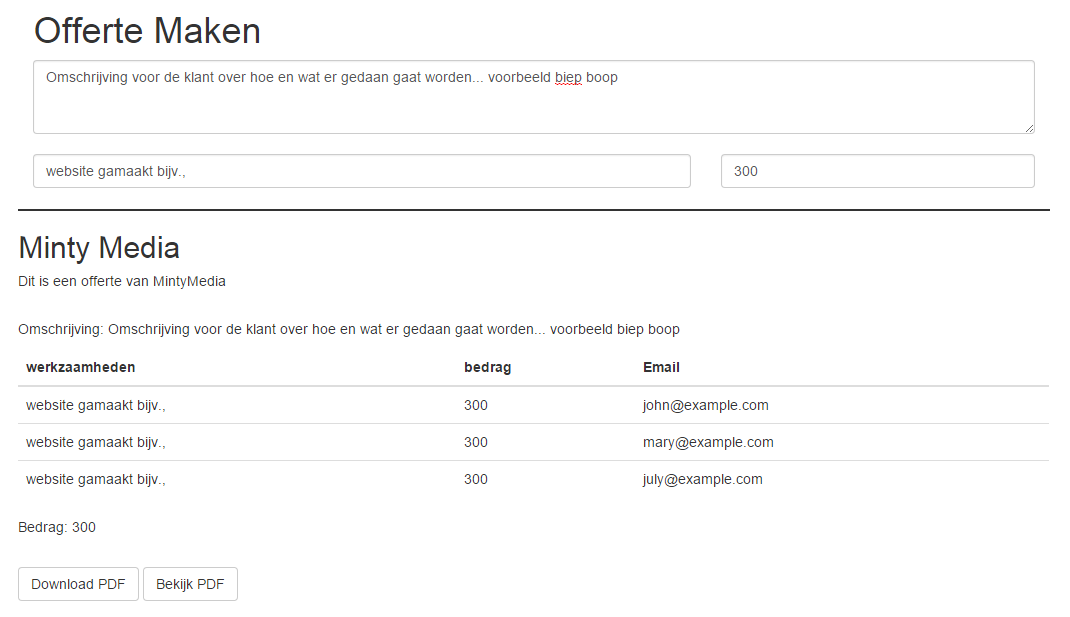


Na al wat werk gedaan te hebben is dit de demo web applicatie, de web applicatie past zich aan, aan de grote van het scherm, ook wel responsive genoemd.

Verder heeft het een hele simpele en moderne maar toch duidelijke opmaak. Bijna alle styling wordt door Boostrap gedaan, zoals de invul velden, tabel en de knoppen. Ook werkt bootstrap met ‘grids’, dat is een heel handig systeem om de grote en relaties van verschillende blokken in je web applicatie te beheren.

Klein voorbeeldje is bijvoorbeeld in het plaatje hier boven, onder de titel ‘Offerte maken’ staat een invul veld die de class ‘col-sm-12’ heeft. De ‘sm’ staat voor small screens en de 12 staat voor de breedte, met Bootraps grid systeem is de breedte eigenlijk in 12 blokjes verdeeld, col-sm-12 staat voor de hele breedte. Dus bijvoorbeeld col-sm-6 zou voor de helft staan, dan kan je 2 blokken van 6 naast elkaar zetten.

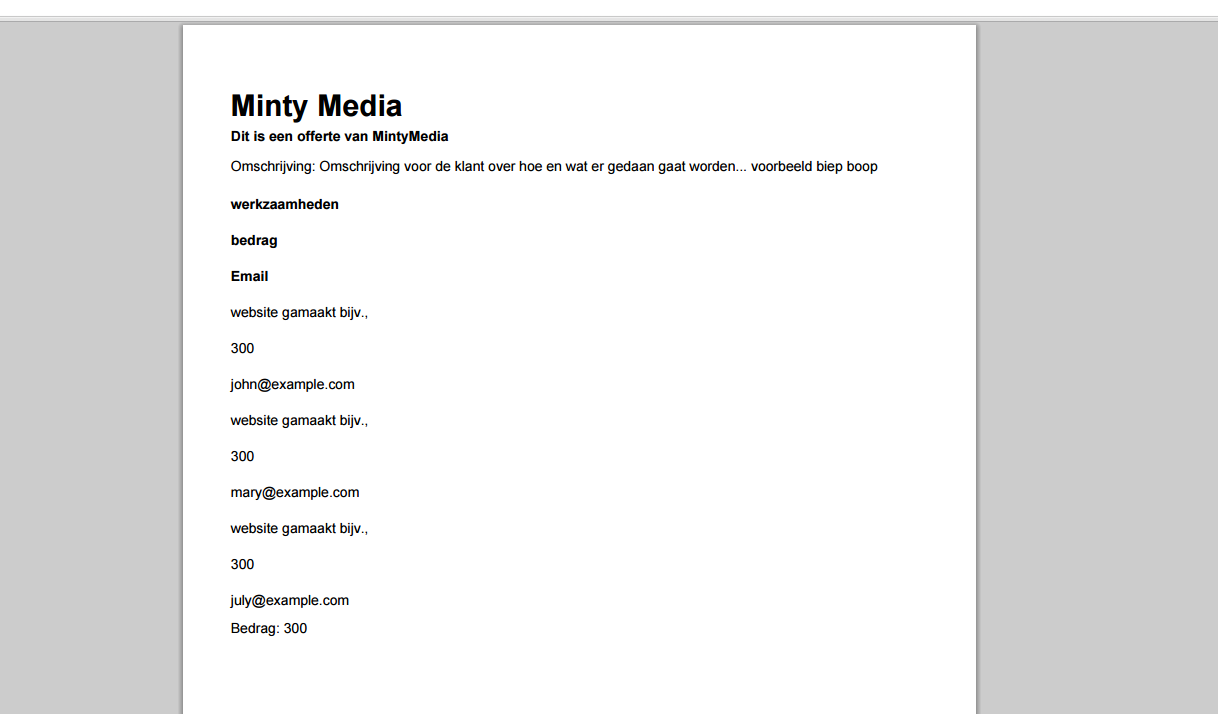
De twee invul velden onder de omschrijving hebben 9(de linker) en 3(de rechter) dus ook samen 12.



De tabel past zich aan naar wat je invult in de input, ik heb nu een paar voorbeeld teksten ingevuld en zoals je kunt zien word het gelijk overal ingevuld doormiddel van AngularJS wat echt super fijn werkt. Er staan ook twee knoppen onderaan, Download PDF en Bekijk PDF.

### JSPDF

Na flink wat google werk kwam ik JSPDF tegen, dat is een klein project gemaakt voor Javascript om PDF bestanden te maken. Ik ben na het design begonnen met het implementeren van JSPDF. Maar ik liep al snel tegen het probleem aan dat er geen documentatie is over de plug-in jsPDF, dat was wel vervelend en veel uitzoekwerk.

De ‘Bekijk PDF’ knop laat het PDF bestand zien hoe het er uit gaat zien met de ingevulde gegevens. En de Download knop download het PDF bestand naar de computer. 

De Bekijk PDF knop opent het PDF bestand in de browser met de ingevulde gegevens.

Wat er onder water gebeurt is eigenlijk dat JSPDF de HTML uit het vorige plaatje onder de streep letterlijk in een PDF bestand kopieert, helaas gaat dat niet zo goed met de tabel.

Dit is dus een probleem die ik tegen kwam en die mijn manier van aanpak dus zou veranderen. Het probleem was eigenlijk dat de JSPDF plug-in die dit mogelijk maakte nog in ontwikkeling is en qua lay-out kan je er nog weinig aan veranderen.

### JSPDF table

Toen kwam ik weer een andere plug-in tegen(een plug-in voor de andere plug-in) genaamd: ‘jsPDF.plugin.table’ die speciaal voor tabellen is gemaakt. Daar kwam dit uit:



Al iets beter, helaas was dit de standaard lay-out van een tabel en kon daar ook niet veel aan gepast worden en we vonden hem nog een beetje simpel dus ben toen verder gaan kijken of ik hem kon aanpassen, helaas kon dat niet. Was wel heel jammer om hier tegen aan te lopen omdat ik al best ver was met deze plug-in en ik had er al best veel tijd in gestopt.

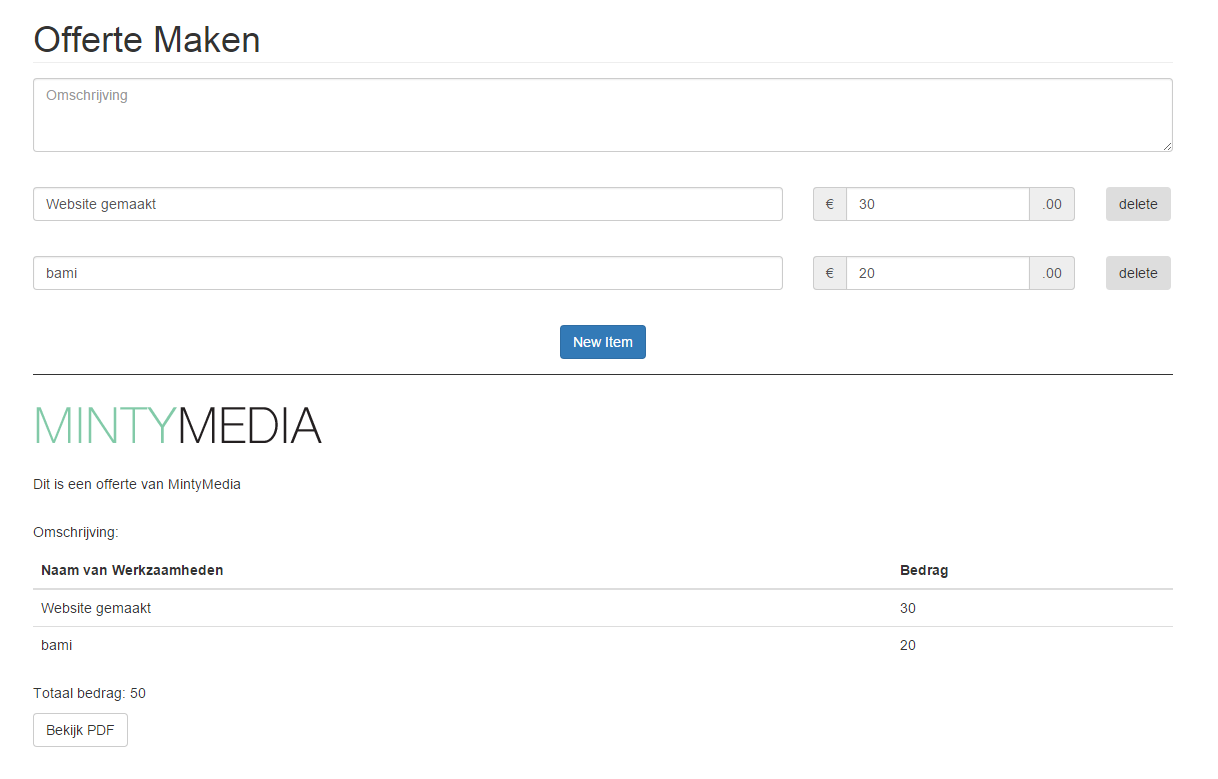
### Chrome Print

Maar ik had dus besloten, na flink wat zoek werk, om het plan van aanpak te veranderen. Job zelf kwam eigenlijk met het idee, je kan namelijk in Chrome op print pagina drukken en dan in plaats van een printer te selecteren op ‘als PDF opslaan’. Nu kreeg je de hele web applicatie als PDF te zien en dat was niet de bedoeling maar daar dacht ik wel een oplossing voor te hebben. Namelijk met CSS kan je per media type verschillende stijlen mee geven. Dus wat ik heb gedaan is voor de print type een aparte CSS class aan gemaakt. Genaamd ‘.verbergen’ en die verbergt het element waar de class op word gebruikt.



Wat hier eigenlijk gebeurt is dat de class alleen voor het media type print de volgende twee dingen doet, namelijk voor de body maakt hij font-size 13px dat is namelijk mooier op papier en bij .verberg class verbergt hij het hele element. Ik kwam er later achter dat bootstrap hier ook al rekening had mee gehouden; hidden-print is een standaard bootstrap class die precies het zelfde doet.

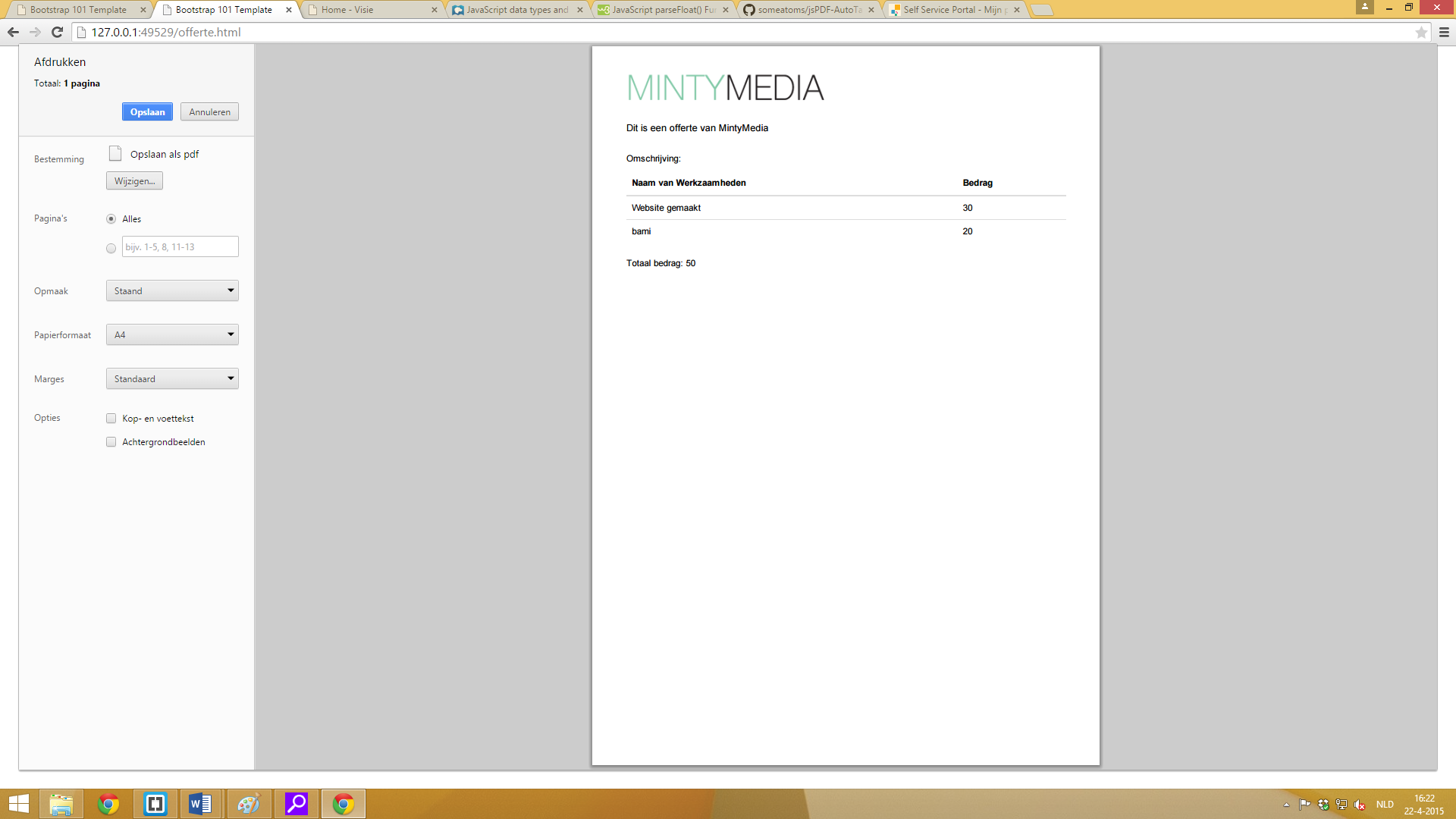
Deze methode heb ik gelijk toegepast aan het systeem, zo verschijnt het print scherm na het klikken op de Bekijk PDF knop. Ook heb ik het een en ander aangepast, zo heb ik het design verder ontwikkelt en een voorbeeld gedeelte gemaakt. Dit is te zien op de volgende afbeelding:



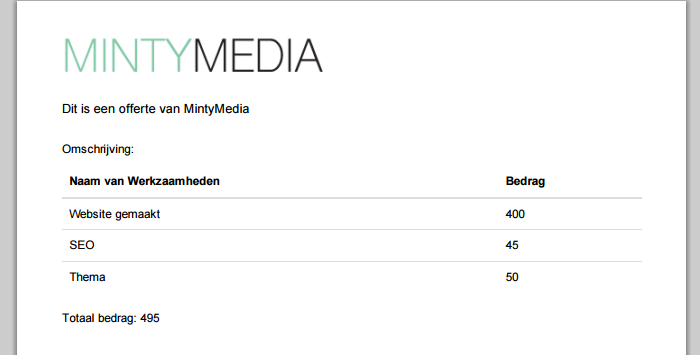
Boven aan staat het formulier met de invul velden die niet op het print/PDF bestand komen en onderaan staat de offerte hoe die er uit gaat zien.

Het bovenste gedeelte is compleet functioneel en interactief, gemaakt met AngularJS. Als er namelijk op de ‘New item’ knop wordt gedrukt word er zowel in het bovenste gedeelte een rij toegevoegd met invul velden en in het onderste gedeelte als tabel rij. Ook als er in het bovenste gedeelte gegevens worden ingevuld in de velden dan word dat gelijk in het onderste gedeelte verwerkt.

De verwijder knop doet precies het tegenovergestelde, namelijk die haalt zowel boven als onder een rij weg. De ‘Bekijk PDF’ knop opent het print scherm:



Zoals je kunt zien word alleen een klein gedeelte van de pagina laten zien. Zo kan ik het gewenste resultaat behalen namelijk een mooie offerte die kan worden opgeslagen en uitgeprint.

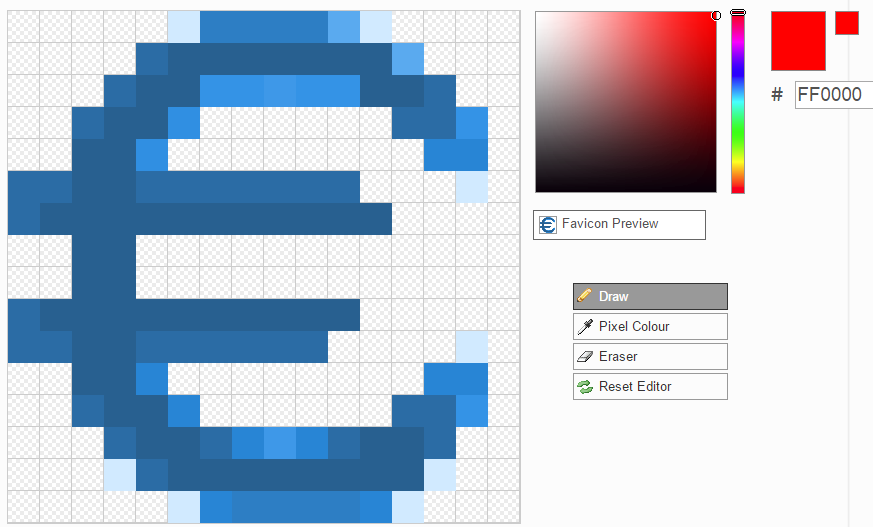


Zoals je kunt zien op de print demo wordt het design en de lay-out van de pagina behouden. In de demo heb ik een simpele tabel, logo en wat tekst neergezet en het begint al een beetje op een offerte te lijken. Dit heb ik aan Job laten zien en die dacht al aan een paar wijzigingen:

* **Footer**: Een standaard voettekst met gegevens; “Voor deze offerte geld zoveel dagen”
* **Totaal** **bedrag**: Rechtsonder drie rijtjes tekst; Subtotaal, BTW % en Totaal bedrag en daar achter het bedrag in getallen
* **Logo**: Een standaard logo van MintyMedia linksboven in (is al gebeurd)
* **Offerte**: Duidelijk kopje ‘Offerte’ ergens
* **Font** **size**: Iets kleinere Font grote wat mooier op papier staat
* **Bedrag**: Bedrag in decimalen

### De favicon

De favicon, afgekort van favoriet icoon, was iets wat ik zelf wel leuk vond om te doen. Heb me altijd afgevraagd hoe dat nou precies werkt qua resolutie enzovoort. Dus ging ik daar mee aan de slag, ik had het idee om zelf een icoontje te maken voor de gein. Online favicon maker was een soort pixel editor want het was 16 x 16 pixels.



Het idee was om een euro teken te maken met de kleur van typisch bootstrap blauw. Het was leuk om een keer te doen en het zag er uiteindelijk best goed uit, ook al zeg ik het zelf. Het was achter af veel simpeler om te doen, even een icoontje maken en hem invoegen.

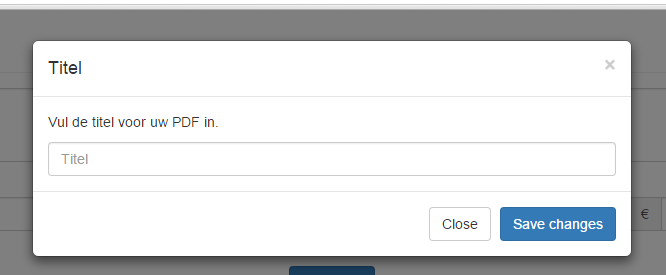


### PDF Titel

De titel om de PDF op te slaan kan je invoeren in het print scherm als je op downloaden klikt. Maar dan kan het systeem die gegeven nergens meer voor gebruiken maar job dacht er aan om de offertes ook gelijk in een Database op te slaan.

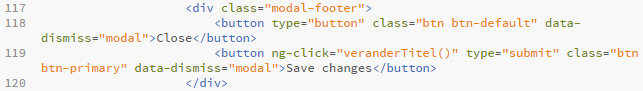
Dus toen kwam ik met het idee om dan daar ook een input box voor te maken en daar de titel in te vullen. Zo kan ik de titel gelijk door voeren in het systeem. Ik heb het nu zo gemaakt dat als je op ‘bekijk PDF’ klikt dat je dan een pop up schermpje te zien krijgt.

Hier kan men de gewenste titel invoeren, die wordt onderwater opgeslagen als AngularJS variabel. Ik kan daar dus alles mee doen wat ik wil, bijvoorbeeld als je op PDF opslaan klikt, voert die automatisch de titel van de web applicatie als titel voor het PDF bestand in. Wat ik dus heb gedaan is dat ik de titel van de web applicatie aanpas naar de opgegeven titel. Ik zal zo dieper op deze functie in gaan, het pop up scherm ziet er als volgt uit:



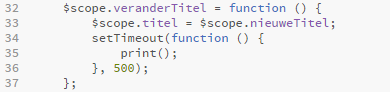
Ik zal iets dieper in gaan op deze functie. Ik zal bijvoorbeeld wat code laten zien die de functie aanroept, de functie zelf en waar de functie effect op heeft. Het is maar een kleine simpele functie dus niets ingewikkelds, het gaat om de functie ‘veranderTitel()’;

Waar de functie wordt aangeroepen:



De modal is het pop up scherm en de footer(voettekst) is het onderste gedeelte van de modal en word onderscheiden met een horizontale streep. In de afbeelding van de pop up kun je zien dat er twee knoppen staan in de voettekst, ‘Close ‘ en ‘Save changes’. De ‘Save changes’ knop heeft een attribuut genaamd ‘ng-click’, dat is een AngularJS ‘directive’ die ervoor kan zorgen dat als je op een element klikt een functie kan aanroepen. In dit geval is het element de button en als je daar op klikt word de functie ‘veranderTitel’ aangeroepen.

De AngularJS functie die wordt aangeroepen:



De eerste regel is standaard, daar wordt de functie aangemaakt. De tweede regel zet de variabel ‘titel’ op ‘nieuweTitel’. Wat daarna gebeurt leg ik later uit. Het is namelijk een bugfix, het gaat namelijk eigenlijk om de print functie. De print functie roept het standaard print scherm van de browser op. Die verschijnt dus nadat de titel is ingevoerd. Ik leg hierna uit wat er eigenlijk precies gebeurt.

Waar de functie invloed op heeft:

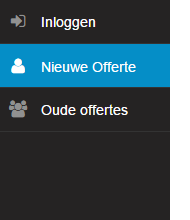


De title tag is een standaard HTML tag, het geeft de web applicatie een titel. Standaard als je in het print scherm zit en je klikt op ‘opslaan als PDF’ pakt hij de titel van de web applicatie als PDF titel. Op die manier kan ik dus de titel van de PDF manipuleren. De title tag/element heeft een attribuut genaamd ‘ng-bind’, dat is een AngularJS ‘directive’ die een variabel aan een element bind. In dit geval is het de variabel ‘titel’, en wat het dus eigenlijk doet is dat hij in de Javascript scope kijkt naar een variabel genaamd titel zoekt en daar de titel uit haalt.

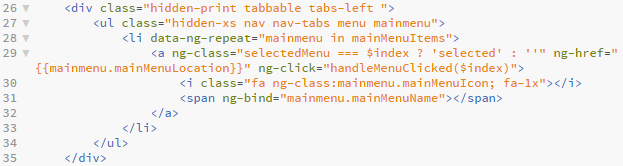
De bug fix: helaas als ik alleen de print functie gebruik in de veranderTitel functie, zonder de timout, werkte hij niet omdat het te snel gaat. De setTimeout functie zet de functie eigenlijk 0,5 seconden in de wacht. Als dat niet gebeurt dan springt hij van het pop-up scherm om een titel in te voeren gelijk naar het print scherm. Zonder de titel te veranderen.

### Side Menu

Na alle openstaande wijzigingen door te voeren ben ik verder gegaan met een sidebar. Bootstrap heeft helaas zelf geen standaard left sidebar dus heb dat van pureCSS gebruikt. Sluit verder wel goed aan bij bootstrap. Heb toen zelf custom kleuren voor de knoppen met hover en active etc. gebruikt.



Modern en flat design met simpele en duidelijke kleuren. De icoontjes komen van font-awesome. De huidige geselecteerde button regel ik met AngularJS



Dit is de HTML code voor het menu. Ik begin met een <div> element die een aantal stylesheet classen heeft. Bijvoorbeeld de hidden-print dat is een bootstrap class die het <div> element onzichtbaar maakt op een print scherm.

Het <ul> element, zie je vaak in de structuur van een menu, het is eigenlijk het parent element van <li> waar <ul> het hele menu is en <li> de aparte buttons.

In <li> staat het ng-repeat attribuut dat is een AngularJS directive. Ng-repeat herhaalt het hele element voor elke value(waarde) in een array. Wat er eigenlijk staat: <li herhaal-dit-element-voor-elke”waarde in array”>. Dus als je 3 waardes in een array hebt staan, krijg je 3 keer <li>…</li> te zien, met alle child element zoals <a>.

In dit voorbeeld is mainMenuItems de array en mainmenu de waarde uit de array die je kan gebruiken.

Het <a> element heeft een ng-class attribuut die kijkt of ‘selectedMenu’ gelijk is aan de $index. $index is een standaard waarde binnen de ng-repeat directive die bij houd bij welke waarde de ng-repeat is, als je er al 3 hebt staat $index op 3(eigenlijk 2, want hij begint bij 0).

En als die gelijk is dan krijgt het <a> element de class ‘selected’ en als het niet gelijk is dan ‘ ’, dus niets.

De ng-href is gewoon een href element die linkt naar een URL. Hij haalt de link uit de waarde van de array, dit wordt zo duidelijker als je de AngularJS code ziet. De ng-click directive heb ik hiervoor ook besproken, in dit voorbeeld wordt de functie “handleMenuClicked” opgeroepen. En die geeft de waarde van $index mee, hier ga ik zo dieper op in.

We zien ook een <i> element staan, die wordt gebruikt om er een icoontje van font-awesome aan te plakken. Als laatste staat er een <span> element met een ng-bind attribuut, die heb ik hiervoor al besproken. In dit voorbeeld koppelt hij de waarde van mainmenu.mainMenuName aan het span element, dit wordt dus gewoon tekst.

De AngularJS code gaat voor veel helderheid zorgen denk ik, dit is de mainMenuItems array:



Deze array heeft 3 objecten met elk 3 property; ‘mainMenuName’, ‘mainMenuIcon’ en ‘mainMenuLocation’. De ng-repeat directive gaat elk object uit de array langs en haalt daarbij de gegevens uit de property’s als die worden opgevraagd.



Hier wordt mainmenu het object uit mainMenuItems, hij blijft doorgaan tot dat alle objecten zijn geweest. Hierna kan ik bijvoorbeeld mainmenu.mainMenuName aanspreken want die property heeft die uit de array gehaald en in dit object geplaatst.

Je kan ook buiten de ng-repeat een array/object uitlezen; mainMenuItems[0].mainMenuName zou bijvoorbeeld ‘Inloggen’ retouren.

Aan de hand van het plaatje van de array kan je de gegevens invullen bij de HTML code onder de ng-repeat.

Als er op de knop wordt gedrukt wordt er een functie aangeroepen genaamd ‘handleMenuClicked’ die functie ziet er uit als volgend:



Devariabel selectedMenu wordt op de meegegeven index gezet. De variabel zit in een array genaamd parentVariables, die gebruik ik om alle variables in op te slaan die in meerdere controllers moeten worden gebruikt. Dus bijvoorbeeld de status van een menu moet in meerdere controllers kunnen worden gecheckt en aangepast.

### Github

Bij het ontwikkelen van dit project ben ik het al vaak tegengekomen, namelijk dat er vaak opensource software word gehost op Github. Nu leek het mij wel ook wel handig om Github te gebruiken, bijvoorbeeld voor versie control, met Github kan je elke update terugzien en kijken wat er precies is aangepast en verandert.

Ook staat met Github je project gelijk online, dit vond ik wel grappig want dan kan ik thuis bijvoorbeeld gewoon verder werken. Of een broer of vriend kan er even naar kijken waar je mee bezig bent.

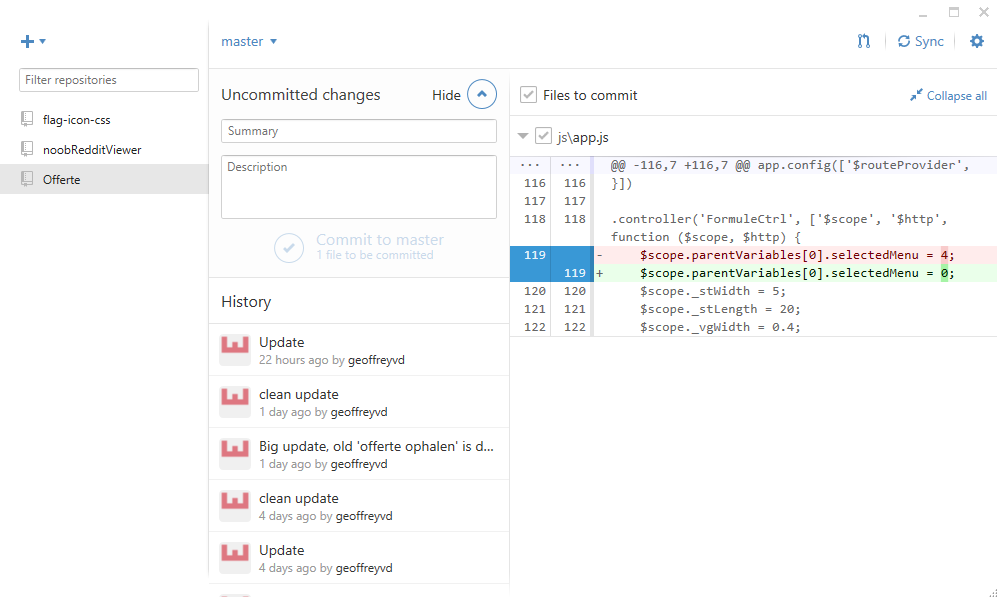
Dit is dus niet alleen grappig maar ook erg handig, je hebt namelijk altijd wel een soort Back up online staan waarbij iedereen je zou mee kunnen helpen met ontwikkelen. En het leek me altijd al eens leuk om zelf een project online te zetten op Github omdat ik het zo vaak tegen kwam.

Ik had dit meer voor mezelf gedaan zodat ik een beetje kon bijhouden wat ik allemaal had verricht en waar ik mee begon en dat ik kan terug zien hoe ik door het project ben gegaan. Ik had dan ook niet verwacht dat een random persoon mijn project zou bekijken/aanpassen.

Dit zou in een bedrijfsomgeving niet toepasselijk zijn omdat de source-code dan van het bedrijf zelf zijn.

Ik zou je dan ook willen vragen om mijn ‘[Github repository](https://github.com/geoffreyvd/Offerte)’ te bekijken. Hier kunt je de omschrijving van het project bekijken en het gehele project downloaden: <https://github.com/geoffreyvd/Offerte>

Hier staat de tijdlijn van het project met alle updates erbij, Inclusief alle veranderingen met commentaar en uitleg: <https://github.com/geoffreyvd/Offerte/commits/master>

De Github user interface

### Database

De web applicatie begint al ergens op te lijken, functioneel en grafisch gezien. De functionaliteit om een offerte te maken is er al, alleen nog niet compleet. En dus hadden we(Job en ik) het idee om er een database aan te koppelen waarbij de offertes kunnen worden opgeslagen en opnieuw bekeken en/of aangepast. Ook willen we oudere offertes opnieuw kunnen versturen.

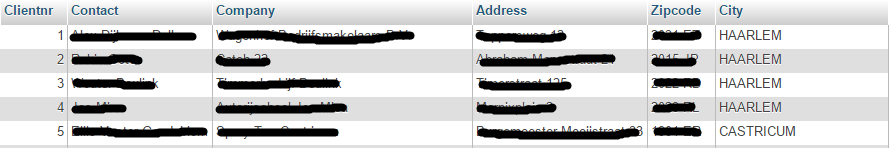
Ik ga ook een klant tabel maken waar ik alle klant gegevens kan opslaan, zodat als je een offerte wilt sturen alleen maar een klant hoeft te selecteren uit een lijstje en dat alle klant gegevens al worden opgehaald, en eventueel ingevuld in de offerte.

Ik wil dus 3 tabellen gaan maken; offertes, werkzaamheden en klanten.

De klanten tabel bevat alle informatie van de klanten met een unieke waarde(primary key); ‘Clientnr’, die wordt gebruikt om de klanten tabel aan de offertes tabel te koppelen. Elke rij in de klanten tabel bestaat uit één klant met elke kolom een ander persoonlijke gegeven. De klant gegevens stonden al in het oude offerte systeem dus die heb ik geïmporteerd.

Ik gebruik de algemene ‘code style’ en die hanteert een aantal regels voor de stijl van code, waaronder dat de kolom namen met een hoofdletter beginnen. Een ‘code style’ zorgt ervoor dat iedereen dezelfde richtlijnen gebruikt, bijvoorbeeld binnen een project is dit erg handig.

De klanten tabel ziet er zo uit:

 De gegevens zijn weggevaagd omdat het klantgegevens zijn van het bedrijf.

De ‘Clientnr’ kolom is een ‘auto\_increment’ dat betekent dat hij er automatisch een uniek getal neer zet, hij begint standaard bij 1 en doet er voor elke rij(row) eentje bij. Dat is erg handig want zo heb je altijd een unieke waarde per rij.

De werkzaamheden tabel bevat alle werkzaamheden met prijs per offerte. Dus per rij heb je een uniek ID; WerkzaamheidID met één werkzaamheid; WerkzaamheidTitel met een prijs; WerkzaamheidPrijs met de bijbehorende offerte via het offerte id; OfferteID.

 Zo ziet de werkzaamheden tabel eruit met wat voorbeeld gegevens.

De offerteID heb ik nodig om de werkzaamheden aan een offerte te koppelen, ik twijfelde eerst nog of ik misschien een hulptabel zou gaan maken om per werkzaamheidID een OfferteID te koppelen om het nog meer te normaliseren.

Dat werd alleen een stuk minder overzichtelijk en minder efficiënt om mee te werken omdat je dan alles dubbel zou moeten opslaan.

Ook zou je elke keer weer de opgeslagen auto\_increment moeten opvragen wat alleen maar voor een stuk meer traffic en gedoe zou zorgen. Dit vond ik de duidelijkere en makkelijkere oplossing om mee te werken.

De offertes tabel bevat per rij alle gegevens van één offerte, het is ook de tabel die de twee andere tabellen verbind.

Elke offerte heeft een uniek id die word opgeslagen in de OfferteID kolom. De offerte naam wordt opgeslagen onder de kolom OfferteNaam. De omschrijving van de offerte wordt opgeslagen onder OfferteOmschrijving en de datum onder OfferteDatum. De OfferteDatum is overigens een automatisch toegevoegde waarde, namelijk de dag dat de offerte in de database word gezet. KlantID bewaart het klant id van de klant van de gestuurde offerte. Zo kan ik met het klantID de klant gegevens ophalen uit de klanten tabel.

Zo ziet de offertes tabel er uit met voorbeeld gegevens.

Deze tabel koppelt eigenlijk de drie tabellen aan elkaar doormiddel van de OfferteID en de KlantID zo kan ik met een query alle data van één offerte uit alle 3 de tabellen halen. Een query voert een opdracht uit op de database, bijvoorbeeld het ophalen van data of het opslaan van data in de database. De volgende query haalt alle gegevens van een offerte op:

 Ik gebruik de ‘inner join’ syntax twee maal zo voeg ik de 3 tabellen aan elkaar. Met ‘on’ zet ik vast welke twee kolommen naar elkaar moeten kijken en hoe de tabellen gelinkt zijn.

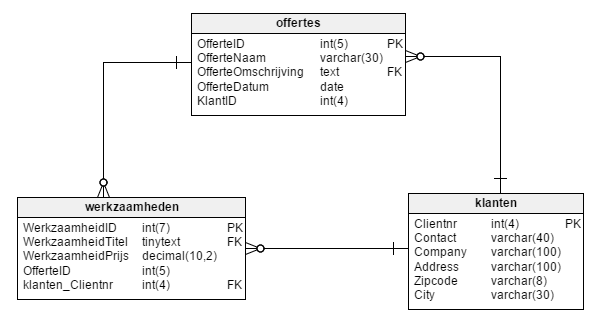
Met de ‘where’ syntax kan men een voorwaarde aan de query toevoegen, in dit geval is de voorwaarde “offertes.OfferteID = 22”. Dus de voorwaarde is dat OfferteID gelijk moet zijn aan 22. Dus ik selecteer één offerte bij zijn id want die is altijd uniek. Het eerste gedeelte, “Select \* From”, haalt de data op van alle kolomen van één tabel.

Het resultaat van de bovenstaande query:





Het resultaat was te groot om in 1 plaatje te houden, ze horen dus achter elkaar. Zoals je kunt zien worden alle gegevens opgehaald, vier maal zelfs, dit komt omdat er vier werkzaamheden zijn verricht en per werkzaamheid worden de gegevens opgehaald.

De structuur wordt nog wat duidelijker met de volgende afbeelding:

Eén klant kan meerdere offertes hebben, één offerte kan maar één klant hebben. Eén offerte kan meerdere werkzaamheden hebben, één werkzaamheid kan maar één offerte hebben. Eén klant kan meerdere werkzaamheden hebben, één werkzaamheid kan maar één klant hebben.

We kunnen met deze relatie bijvoorbeeld ook van één klant alle offertes en werkzaamheden ophalen, of van een bepaalde werkzaamheid de klant gegevens ophalen.

Ik host deze database op mijn localhost op xampp.

### Back end

De Front end(Presentation layer) is voorzien van functionaliteit doormiddel van AngularJS die handelingen beheert en invul velden uitleest. De data layer is er ook al met de database die klaar staat om te gebruiken.

Nu nog de Back end die de handelingen van de front end door kan voeren in de database. Ik had al veel ervaring met PHP dus dat klonk wel aantrekkelijk maar ik wilde eerst nog verder kijken of er niet iets beter samenwerkt met AngularJS, een soort api. En vond het wel leuk om iets nieuws te proberen. Toen kwam ik veel asp.net tegen en nodejs, helaas waren daar veel installaties voor nodig en ik moest wel een beetje opschieten met het project dus dat werd hem niet.

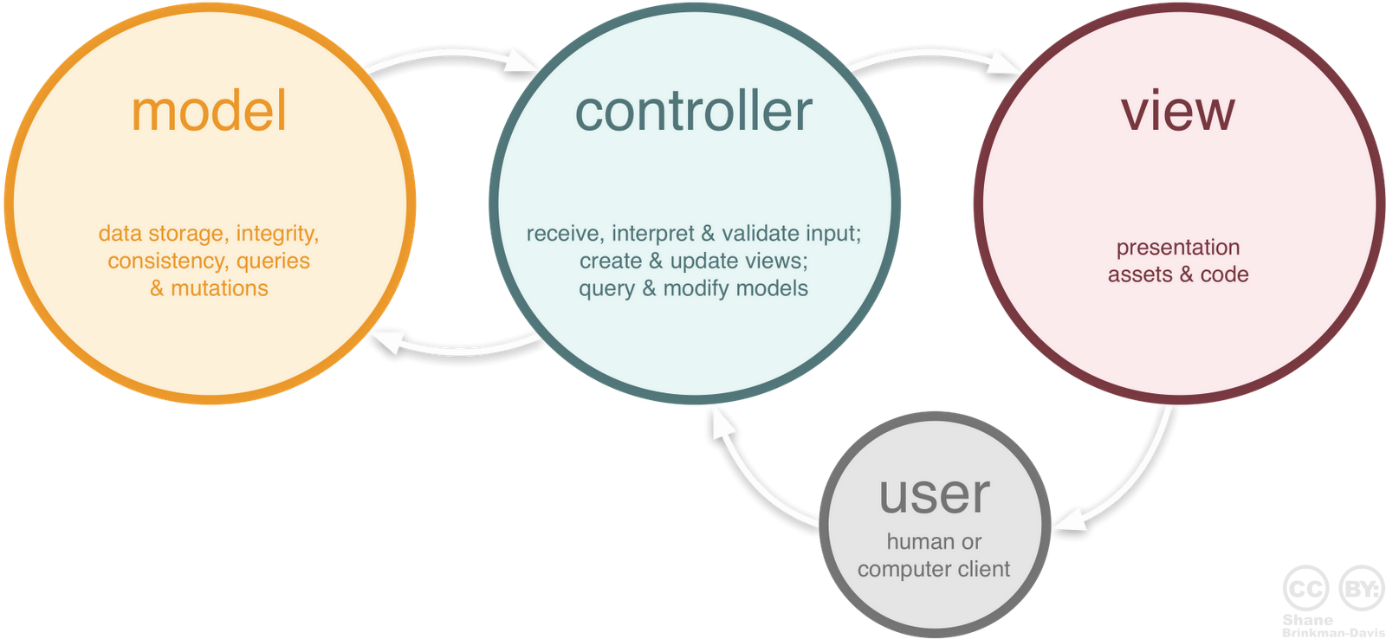
Toen ben ik gaan zoeken of ik misschien een PHP api kan maken. En daar waren ook al een paar frameworks voor, helaas zonder documentatie. Dus na veel pogingen om de bestaande frameworkds aan de praat te krijgen, wat uiteindelijk ook is gelukt, bezig geweest met het uitvogelen van hoe het precies werkt. Dat lukte helaas niet helemaal ook lukte het niet om de framework in mijn project te implementeren, dus heb toen maar de hoop opgegeven en besloten om zelf een api te maken.

Na veel research en kennis die ik al had, had ik gevonden wat ik nodig had. Met AngularJS kan een http request(*verzoek*) gestuurd worden naar een api, zo kan je met een ‘post’, data meegeven met een request. Zo stuur ik de request met json naar het PHP bestand die de request gaat behandelen. En dan eventueel wat terug stuurt met een soort return.

Zo kan met de request gegevens mee gestuurd worden om met PHP bijvoorbeeld die gegevens op te slaan in de database. Met de return kan ik in AngularJS de data uitlezen die word terug gestuurd, op die manier kan de data opgehaald worden. Als ik de twee combineer kan de data meegegeven worden om bijvoorbeeld een voorwaarde aan query te stellen en dan specifieke informatie op te halen.

Wat ik nu eigenlijk doe is dat ik PHP ook naar de data layer verplaatsen, en AngularJS de complete business layer. Dit is een nieuwe methode en mijn manier van aanpak omdat het een aantal voordelen heeft en ook omdat ik het erg interessant vind om te maken.

Zo ga ik dus in AngularJS de logica bouwen en met PHP haal ik alleen data uit de database en door gestuurd naar AngularJS er gebeurt verder geen logica in PHP. Het is dus een api(*Application Programming Interface*), dat is dus een interface die je kan aanspreken en die letterlijk alleen data ophaalt of verwerkt en dan een response heeft. Dit ontwikkelpatroon staat bekent als MVC(*model, view, controller*). Waarbij de model de data is, de controller de business logic(de logica) en de view is de Front end(de user interface).



Op deze manier spreek ik met AngularJS de PHP code aan die op zijn beurt weer een query naar de SQL database stuurt. Bijvoorbeeld als er op een knop wordt gedrukt registreert AngularJS dat, die kan dan naar PHP een request sturen om wat data op te halen. Die stuurt dan een query naar de database, die database heeft een reactie dat word terug gestuurd naar PHP. PHP stuurt het weer door naar AngularJS, die op zijn beurt weer de data in een tabel verwerkt.

Dit gebeurt allemaal onderwater, je merkt hier niets van. Dit zou allemaal ook mogelijk zijn om met PHP te maken, maar wat je dan krijgt is bij elke request een refresh. Dit komt omdat het server-side is, dus bij elke verandering op de pagina zal de hele pagina opnieuw opgevraagd en geladen moeten worden. AngularJS is een soort uitbreiding op Javascript en Javascript is gewoon clientside, dus ik kan alles live aanpassen naar hoe ik het willen zonder het verversen van de pagina.

Een aantal voordelen van deze ontwikkel methode:

* Het live updaten van de web applicatie (interactieve web applicatie voelt beter aan)
* Performance is beter omdat de pagina maar één keer word geladen en dan word elke keer kleine stukjes data opgehaald i.p.v. elke keer een verversing van de gehele web applicatie en dan nog die extra data
* Mogelijkheid om een SPA(single page application) te bouwen, helemaal geen verversingen en redirects(*verwijzingen*) meer
* Duidelijke onderscheidingen tussen Front end en Back end, dus overzichtelijke blokken code
* Een API staat los van alles dus kan hergebruikt worden in andere projecten
* De controller kan de data die hij ontvangt manipuleren, veel mogelijkheden
* Veel nieuwe dingen, interessant en leerzaam

Een aantal nadelen:

* Nieuwe aanpak, veel research kost tijd
* Het komt niet altijd goed uit dat de controller de data manipuleert, bijvoorbeeld als er een grote hoeveelheid data word opgevraagd, dan zou je met PHP een limit er op kunnen zetten maar dat is ook al logica. Dus nu word al die data opgestuurd en dan word er pas in de controller een limit opgezet.
* Veel verschillende frameworks die je moet researchen en opzetten binnen het project en dan kijken of het met elkaar over weg kan, kost ook veel tijd

Zo heeft het dus voordelen en nadelen, één nadeel was dat er meer data moet worden verstuurt. In dit project maakt dat niet zoveel uit omdat alles lokaal wordt gehost en word maar door één iemand gebruikt. Dus niet duizend man die de data opvraagt. Ook is die grote hoeveelheid data nog steeds minder groot dan elke keer de web applicatie opnieuw op te vragen.

Zo heb ik dit ook bij job neergelegd.

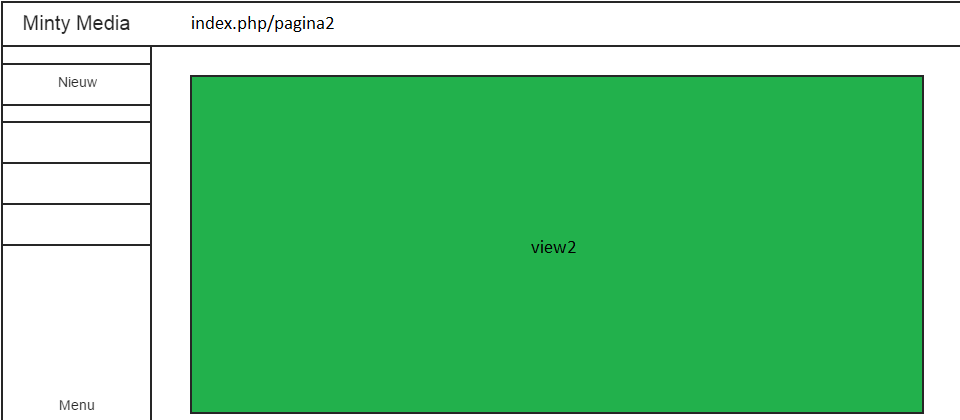
### Views & routing

Met AngularJS kan men ook views maken, dat zijn soort frames binnen de pagina. Die worden aangestuurd door de URL. Bijvoorbeeld als je index.PHP hebt en daarbinnen is de content de view. Dan kan je hem standaard de ‘default view’ meegeven bijvoorbeeld als je dan naar index.PHP/pagina2 gaat kan je de content veranderen naar view2. Zo blijft de hele web applicatie hetzelfde, bijvoorbeeld het menu en de voettekst zonder verversing van de pagina, en alleen de view verandert. Dit wordt ook wel een single page application genoemd.



Dit voorbeeld demonstreert de index.PHP pagina die de default view hanteert, de view is eigenlijk de content in dit voorbeeld en word aangegeven met rood. De default view kan een heel andere lay-out hebben dan elk andere view.

Zo kan een button in het menu bijvoorbeeld linken naar index.PHP/pagina2, AngularJS vangt dit op en verandert de view van ‘default view’ naar ‘view2’ en zo verandert de web applicatie interactief zonder het verversen van de pagina.



Dit voorbeeld demonstreert index.PHP/pagina2, ik zit eigenlijk nog op de index.PHP pagina alleen de view verandert. Zo word hier de view2 in de content geladen, een view bestaat uit een template en een controller. De template is een HTML bestand die zorgt voor de opmaak van de view, de controller is een AngularJS controller die alles regelt wat er in die view gebeurt.

Op deze manier wordt er voor gezorgd dat je binnen de web applicatie naar andere delen van de web applicatie kan zonder elke keer je pagina te verversen. Dus je menu en alles wat eigenlijk nooit verandert, staat buiten de view en die worden nooit opnieuw opgehaald. Alleen de view wordt opgehaald als hij wordt aangevraagd. Ook de URL verandert mee, zo lijkt het alsof je naar een andere pagina gaat alleen zonder iets te laden, dus het gaat snel en soepel.

In de code ziet dat er zo uit:



De functie routeProvider koppelt een view aan een route. In de eerste regel wordt de routeProvider geïnjecteerd in de module zo kan ik dit onderdeel gebruiken. Daarna wordt de functie aangeroepen en geconfigureerd. De ‘.when’ syntax is de voorwaarde waar de URL aan moet voldoen. Zo moet bij de eerste voldaan worden aan ‘/oudeoffertes’, dus hij kijkt wat er in de URL na de standaard index.PHP komt.

Als ik dus op een knop druk of in de URL het volgende typen: “index.PHP/oudeoffertes”, dan kom ik in deze route terecht. In dit geval wordt de titel verandert naar “Oudeoffertes” en de templateUrl haalt het HTML template voor de view op. De controller bepaalt welke controller de handelingen moeten verrichten in deze view.

Zo worden er nog een paar routes ingeladen, de laatste twee zijn voor de default views. Als laatste staat er de ‘.otherwise’, die is erg handig bijvoorbeeld als iemand een verkeerde URL heeft ingevoerd kan je dat opvangen en een bepaalde template laten zien.



Dit stukje code wordt ingeladen in de index.PHP, op die manier wordt de view aan de pagina gebonden. Wat er bijvoorbeeld gebeurt als we op index.PHP pagina komen, dan wordt de default view template hier in geladen. Dus dat HTML bestand wordt dan eigenlijk letterlijk hier in geplakt. Als we dan naar een andere route gaan dan word er een andere view ingeladen.



Zo begint de ‘nieuweofferte’ template, die wordt dus zo geladen in de content/container. En alle AngularJS bindingen zoals de ng-model, ng-bind en ng-click worden in de opgegeven controller verwerkt.

We hebben nu een single page applicatie.

### AngularJS request

De Front end structuur staat er nu en functioneel werkt het allemaal naar mijn wens. Nu wil ik dus de connectie met de Back end maken, en gegevens ophalen en verwerken. Voordat ik met PHP aan de slag kan moet ik eerst de connectie maken tussen AngularJS en PHP.

Ik wil met AngularJS een request sturen naar een PHP bestand, die vangt dat dan op, verwerkt het en stuurt een antwoord terug. Ik begin dus met AngularJS. Na veel zoek werk ben ik de AngularJS dependency ‘$http’ tegen gekomen, daarmee kan je http requests sturen. Een AngularJS dependency is een AngularJS service die niet standaard in de AngularJS framework is verwerkt, zo moet je een aparte dependency ‘injecten’ in een controller zodat je hem kan gebruiken. Dat ziet er zo uit:



Hier worden $scope en $http ingeladen in de controller zodat deze functies gebruikt kunnen worden.



Ik maak nu een request aan die als post word verstuurd naar de gewenste URL, in dit geval ‘PHP/getAllKlanten.PHP’. Ik kan ook data meegeven maar dat doen we in dit geval niet omdat ik alle klanten wil hebben en niet bijvoorbeeld 1 specifieke klant. De headers convert hem naar json zodat PHP er goed mee overweg kan.

De $http service werkt met promises, dat houd in dat een request altijd met een succes of een error wordt terug gestuurd. Zo kun je ze apart afhandelen, dit zien we ook terug in het plaatje.

‘Request.succes’ vangt de request op als het een succes is, hij schrijft in dit geval naar de console dat het een succes was en geeft alle data mee die terug is gestuurd. Ook word de meegestuurde data opgeslagen in de scope variabel ‘alleKlanten’.

Als we bijvoorbeeld een offerte willen opslaan moeten we wel data mee sturen naar PHP. Dit is te zien bij de functie uploadOfferte:



We sturen hier de titel, omschrijving, een array met werkzaamheden en de klantid mee.

### PHP Back end

Nu ga ik de Back end ontwerpen zodat ik de request kan opvangen, ik ga dit doen met PHP. Ik heb een localhost server waarop ik server-side kan werken met PHP.

Op de foto zie je het bestand postOfferte.PHP, dit bestand handelt de request van het de functie uploadOfferte, die functie stuurt ook data mee.

Op de foto is de code te zien, ik ga deze code behandelen. Aan het begin van de code wordt de data uitgelezen en geconverteerd, van JSON naar PHP variables. De include functie leest een ander PHP bestand uit, zo kan de code hergebruikt worden. Het functions.PHP bestand bevat een aantal functies die ik vaker gebruik, hier kom ik later op terug.

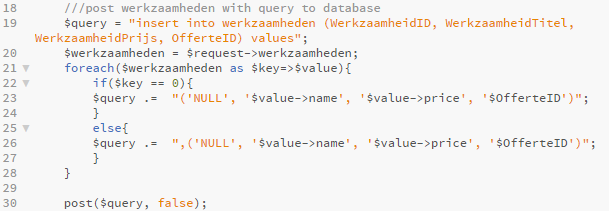
De $query variabel is een string die de opdracht bevat voor de database. De $titel variabel word opgehaald uit $request->titel, $request is een object met alle opgestuurde data. Het zelfde gebeurt bij $klantid en $omschrijving. De $datum wordt op de huidige datum gezet doormiddel van de date functie. Daarna worden al deze variabelen toegevoegd aan de query. $offerteID krijgt het resultaat van de functie post, die in de functions.PHP zit. Er worden ook twee parameters mee gegeven, namelijk de $query en ‘true’.

Op de foto is de code van functions.PHP te zien, dit bestand wordt alleen gebruikt voor de functies, het doet op zichzelf niets.

We gebruiken bij query de post functie, zo worden de twee parameters meegegeven, de query word uitgevoerd en verstuurd naar de database. Ook gaf ik als tweede parameter ‘true’ mee die word gebruikt in de if statement, ‘true’ voldoet aan de voorwaarde.

Als er aan de voorwaarde wordt voldaan dan word er een extra stukje code uitgevoerd: ‘return($mysqli->insert\_id);’. Dit stukje code zorgt ervoor dat de door SQL aangemaakte auto\_increment van de query word terug gestuurd(gereturnt).

De auto\_increment bevat het offerte id van de opgeslagen offerte, dit is een getal. Het resultaat wordt opgeslagen in $OfferteID. De offerte id heb ik nodig om de werkzaamheden te koppelen aan de offertes.

In de foto is de code te zien voor het versturen van de werkzaamheden naar de database.

De $query begint met hard-coded begin van de query, beschrijft in welke tabel de data moet worden opgeslagen en in welke kolommen. Daarna word voor elke werkzaamheid in de array werkzaamheden de naam en prijs van de werkzaamheid toegevoegd aan de query. Ook de offerte id word toegevoegd pet werkzaamheid.

De laatste roept de functie post op en daarmee word de query verstuurd.

Als er een response is van het PHP bestand dan kan AngularJs die verwerken, in dit geval stuurt PHP een soort status report terug, zo kan AngularJS dit naar de console weg schrijven.

### AngularJS verwerking

In PHP heb ik ook een bestand gemaakt genaamd getAllKlanten.php, dit bestand haalt alle klanten op uit de database. De klanten worden dan van PHP naar AngularJS gestuurd.

AngularJS ontvangt deze data op de volgende manier:

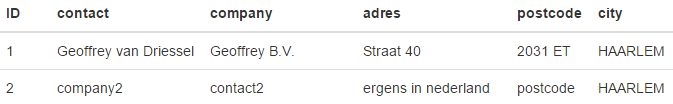


De klanten worden meegegeven in een array met alle klant gegevens. Deze array word opgehaald uit data.klanten en in alleKlanten opgeslagen. Nu heb ik dus de data uit de database gehaald en naar AngularJS gestuurd.

Op de klanten page komt een tabel met alle klanten en alle klant gegevens. Dit is voor Job erg handig zo kan hij bij al zijn klant gegevens met een druk op de knop. Ook worden de gegevens mooi in een tabel verwerkt. Dit gaat op de volgende manier:



De ng-repeat directive staat in de tr tag, de ng-repeat heb ik al een keer besproken dus daar zal ik verder niet op in gaan. De directive haalt in dit geval elke klant uit alleKlanten met alle gegevens en zet die in een tabel. Per kolom wordt een andere waarde uit het object gehaald, zoals te zien is op de bovenstaande afbeelding. Dit creëert het volgende:



Een Bootstrap tabel met de klantgegevens uit de database.

Deze methode kan ik bij meerdere onderdelen van de web applicatie toepassen. Zo haal ik bijvoorbeeld de oude offertes op uit de database met de zelfde methode. Ook deze schrijf ik weg in een tabel op de website.

Ook kan ik met deze methode gegevens weg schrijven naar de database. Dat doe ik bijvoorbeeld met het maken van een nieuwe offerte, zo kan de gebruiker de nieuwe offerte opslaan in de database.

Met deze methode heb ik dus een aantal onderdelen van de website gemaakt, namelijk:

* De nieuwe offerte pagina slaat de nieuwe offerte op in de database.
* De oude offertes pagina haalt alle oude offertes op uit de database en als er op de bekijk knop word gedrukt dan worden alle gegevens van één offerte opgehaald uit de database.
* De klanten pagina haalt alle klanten op uit de database.

### NAS

Nu het project zo goed als af is kan ik het systeem gaan implementeren in de gewenste omgeving. Zo wenst Job graag het systeem op de localhost van de NAS te hebben.

De NAS is al geïnstalleerd op het lokale netwerk, de Synology NAS heeft DSM als besturingssysteem draaien. De NAS heeft ook al een localhost draaien, dus ik kan PHP gebruiken. Voor de SQL interface gebruikt de NAS phpmyadmin, een erg handige gebruikers omgeving zo gebruik ik die zelf ook vaak.

De NAS is alleen binnen het lokale netwerk te raadplegen, dit is voor veiligheidsredenen.

Voor dat ik het project gaan implementeren ben ik nagegaan wat er allemaal geconfigureerd moet worden voordat het systeem werkt zoals verwacht.

De PHP code maakt een connectie met de database, daarvoor moet een database gebruikersnaam, wachtwoord, host en database naam mee gegeven worden. Als één van deze niet overeenkomt met die van de database dan kan de gehele gegevens verzameling niet uitgelezen worden.

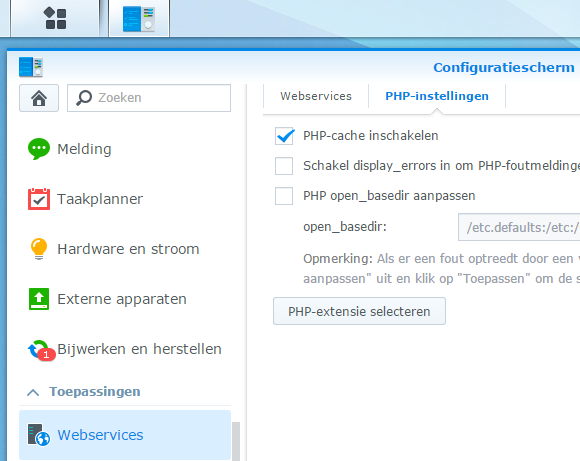
De database credentials(username en password) van de NAS zijn dus anders dan de database credentials van het test systeem. Ik moet de credentials dus aanpassen, deze worden in de functions.php geïnitialiseerd. Deze hoeven dus maar één keer aangepast worden omdat elk keer er een connectie met de database wordt gemaakt, dit bestand wordt aangeroepen.



Hier kan ik dus de juists gegevens invoeren zodat dit overeenstemt met de database van de NAS.

Als dit is gebeurd kan ik de SQL database exporteren van het test systeem, en vervolgens importeren in de database van de NAS. Het exporteren en importeren van een database is geen ingewikkelde klus met phpmyadmin, het is met een druk op de knop gedaan.

Vervolgens kan ik de project map overzetten van het test systeem naar de NAS. De project map bestaat uit alle onderdelen waaronder; de index, alle Javascript/PHP/CSS bestanden en een aantal plaatjes. De project map zet ik in de root folder van de localhost. Op die manier wordt het project gehost op de localhost server. Zo kan je de web applicatie bezoeken in de browser op het lokale netwerk.



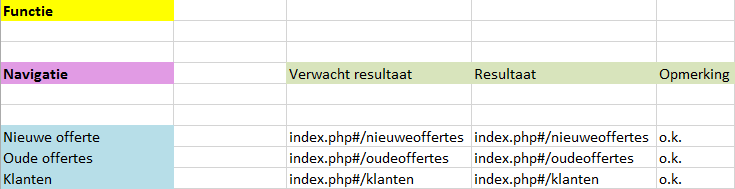
De NAS User interface(gebruikers omgeving)

Het implementeren is goed gegaan, de web applicatie staat werkend op het lokale netwerk met een werkende database verbinding.

### Test

Het bedrijf was te klein om een acceptatietest te maken. Wel is de web applicatie getest door de geen die het systeem gaat gebruiken, namelijk Job. Hij heeft het systeem uit geprobeerd en zo doende feedback gegeven. Zo vind Job het systeem vloeiend werken. Hij vindt het een mooi en duidelijk design hebben.

Een aantal kleine aanpassingen zijn wel nog nodig. Deze punten zal ik verwerken in de web applicatie, zo zal ik de web applicatie beheren. En wanneer nodig veranderingen doorvoeren.

Ik heb een kleine acceptatie test gemaakt deze beschrijft alleen de navigatie, er worden namelijk geen waarschuwingen of validatie’s in de web applicatie uitgevoerd

# Conclusie

Na 3 weken gewerkt te hebben aan dit project is het resultaat er naar. Ik heb er veel van geleerd ook het verslag schrijven vond ik leerzaam. Het systeem voldoet aan de eisen en wensen en voelt compleet aan, echter mist het wel nog een aantal kleine dingetjes. Ook omdat ik uiteindelijk minder tijd had dan ik had verwacht. Namelijk moest dit verslag 2 weken eerder worden opgeleverd.

Ook heb ik veel geleerd van de verschillende fases met het ontwikkelen van de web applicaties, ook dat er meer bij komt kijken dan alleen programmeren. Over het resultaat ben ik zeer tevreden, vooral omdat ik het gehele project van de grond af aan heb opgebouwd en nu staat er een bruikbare en volledige web applicatie.

### Het resultaat

Zoals in de eisen en wensen is beschreven bestaat de web applicatie uit eindelijk uit 3 pagina’s: Nieuwe offerte, Oude offertes en Klanten. Deze pagina bestaan alle 3 uit de zelfde lay-out en hetzelfde menu. Alleen de content verandert.

Ook hoort de database bij het resultaat, de database is volledig ingericht en de connectie met de web applicatie is uitstekend, ze zijn goed op elkaar afgestemd. De NAS hoort ook bij het resultaat, het project is volledig geïnstalleerd op de NAS. Het resultaat ziet er zo uit: 