

Scriptie ingediend tot het behalen van de graad van
PROFESSIONELE BACHELOR IN DE ELEKTRONICA-ICT

The Pitchpoint agency tool

Nick Lauwerijs

academiejaar 2015-2016

AP Hogeschool Antwerpen
Wetenschap & Techniek
Elektronica-ICT

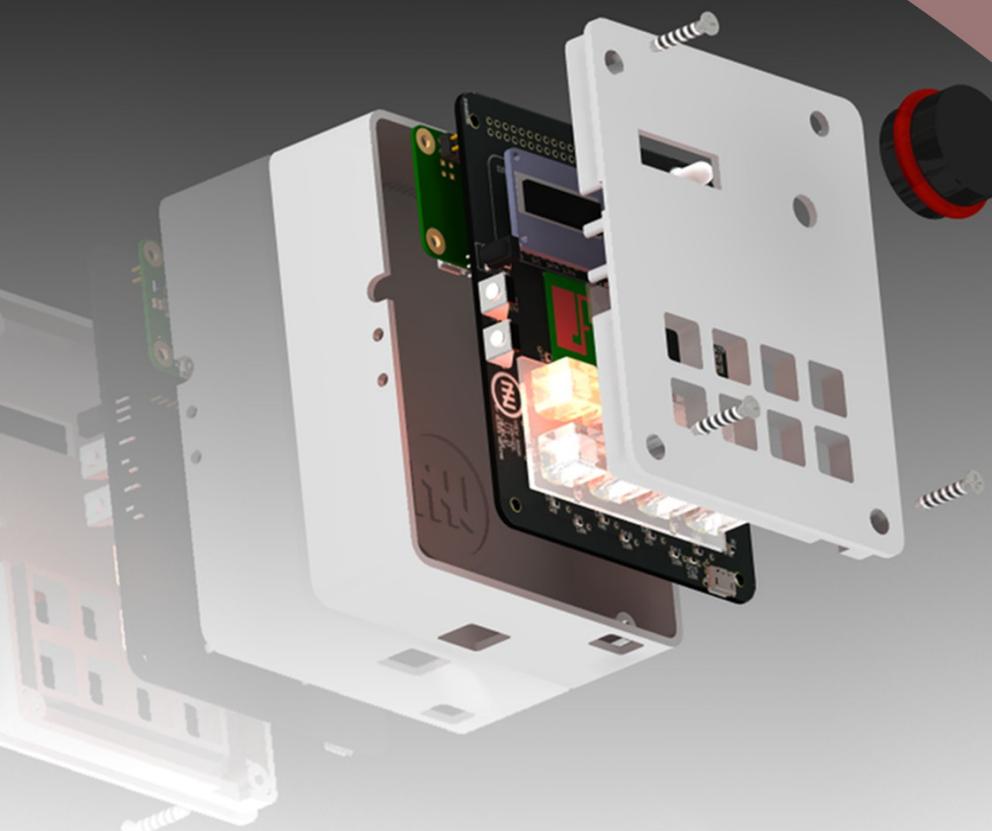


Table of Contents

Abstract	1.1
Dankwoord	1.2
Introductie	1.3
Bedrijf	1.3.1
Probleem	1.3.2
Doelstelling	1.3.3
Technisch	1.4
Gebruikte technologieën	1.4.1
Laravel	1.4.1.1
Bootstrap	1.4.1.2
Sass	1.4.1.3
Gulp	1.4.1.4
Bespreking ontwikkeling	1.4.2
Bespreking Applicatie	1.4.3
Gebruikte tools	1.4.4
Sublime Text	1.4.4.1
Mamp pro	1.4.4.2
HeidiSQL	1.4.4.3
Besluit	1.5
Kleinere opdrachten	1.6
Appendices	1.7
Bibliografie	1.8

Abstract

Prophets is een digitaal marketing bureau gesitueerd in Antwerpen. Tijdens mijn stage bij Prophets moest ik werken aan een web applicatie voor het bedrijf Pitchpoint. De ontwikkeling deed ik samen met Veerle De Vos (andere stagiaire van Prophets).

Pitchpoint helpt adverteerders om een passend communicatiebureau te vinden, maar dit is niet altijd even makkelijk. Daarom vroegen ze om een web platform te ontwikkelen waar communicatiebureau's zich kunnen registreren zodat ze een overzicht hebben van mogelijke kandidaten.

Deze scriptie beschrijft hoe dit project structureel in elkaar zit, de technologieën en software die we hebben gebruikt en hoe het project verliep.



Dankwoord

Na een stage periode van 12 weken waarin ik heel veel heb bijgeleerd omtrent Web Development zou ik graag een dankwoord willen richten aan diegene die hebben geholpen om deze bachelorproef tot een goed einde te brengen.

Graag zou ik ten eerste Kris Van Hauwermeiren bedanken die deze stage plaats heeft aangeboden en mij heeft begeleid gedurende de stage.

Ook zou ik graag Samuel Joos en Peter Vanwyck bedanken omdat deze altijd klaar stonden voor eventuele vragen en advies en ook omdat deze mijn code hebben nagekeken en hierop feedback hebben gegeven.

Alsook zou ik graag Veerle De Vos bedanken voor de fijne samenwerking aan het project en omdat ik heel veel heb bijgeleerd van haar.

Verder wil ik mijn interne promotor Patrick Van Houtven bedanken omdat hij zijn tijd en moeite heeft gestoken in het verbeteren van al de documenten die ik moest inleveren.

*Antwerpen, 10 juni 2016
Nick Lauwerijs*

introduction

Bedrijf

Prophets is een digitaal marketingbureau gesitueerd in Antwerpen. Het bedrijf staat in voor de groei en populariteit van andere bedrijven en probeert dit te verbeteren door marketing strategieën te ontwikkelen. Zo ontwikkeld het bedrijf zelf websites, applicaties, reclame-filmpjes,... om deze strategieën te ondersteunen. Prophets heeft heel wat grote klanten zoals: Carrefour, Orange, De Persgroep, Crelan, ...

Mijn opdracht was om te werken aan projecten voor klanten van Prophets. Eerst ben ik enkele weken bezig geweest om een website te maken voor het [Nucleaire Forum](#). De rest van de stage heb ik gewerkt aan een web applicatie voor Pitchpoint. Aangezien ik het meeste van de tijd aan het Pitchpoint project heb gewerkt heb ik de scriptie hierover geschreven.

Probleem

Pitchpoint's taak is om adverteerders te helpen om een communicatiebureau te vinden dat perfect past bij de behoeftes van de adverteerde. Ze gebruiken hun ervaring met reclamebureau's om zo een match te vinden, dit is niet altijd zo evident omdat er heel wat reclamebureau's bestaan. Het zou dus handig zijn als ze een overzicht zouden hebben van zo veel mogelijk communicatiebureau's zodat ze hun keuze hier ook wat door kunnen laten beïnvloeden.

Doelstelling

Een web applicatie ontwikkelen waarop communicatiebureau's zich kunnen registreren. Elk communicatiebureau moet een login en wachtwoord ingeven zodat ze al hun eigen data kunnen zien en aanpassen.

Dit alles moet worden weergegeven in een admin sectie zodat Pitchpoint een overzicht heeft van elk communicatiebureau. In deze admin sectie moet er gefilterd/gesearched kunnen worden op bepaalde fields. De bureau's in het admin menu moeten in detail kunnen bekijken worden en zijn volledig aanpasbaar door Pitchpoint zelf. Ook moet pitchpoint een overzicht van een bureau kunnen afdrukken waarbij ze zelf kunnen kiezen wat er mee afgedrukt wordt en wat niet.

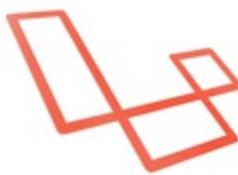
Om deze applicatie te ontwikkelen hebben we gebruik gemaakt van een PHP framework genaamd Laravel in combinatie met Bootstrap, Sass en Jquery voor de Front-end.

Technisch

Gebruikte technologieën

laravel

Laravel is een open-source full-stack PHP web-framework. Laravel kent een grote populariteit omdat het framework de reputatie heeft gemakkelijk in gebruik te zijn en omdat het voorzien is van een heel leesbare syntax. Ook is het framework uitstekend [gedocumenteerd](#) en heeft het een e-learning service genaamd [laracasts](#) die de developer helpt bij het gebruiken van het framework.



Figuur 1: Laravel logo

Het wordt een full-stack framework genoemd omdat het elk onderdeel van een web-applicatie kan hanteren. Dit gaat van data opslaan in de databank tot het renderen van views door de built-in templating engine. Veel componenten die regelmatig terugkomen bij web applicaties zijn al in het framework ingebakken. Zo verliest de developer geen tijd meer met functionaliteit zoals authentication, sessions, caching, ... te implementeren. Ook bieden ze de developer de mogelijkheid tot het gebruik van Laravel Homestead. Dit is een virtuele development environment met alle mogelijke tools die nodig zijn om aan een Laravel project te werken. In deze ontwikkel omgeving zijn volgende programma's inbegrepen:

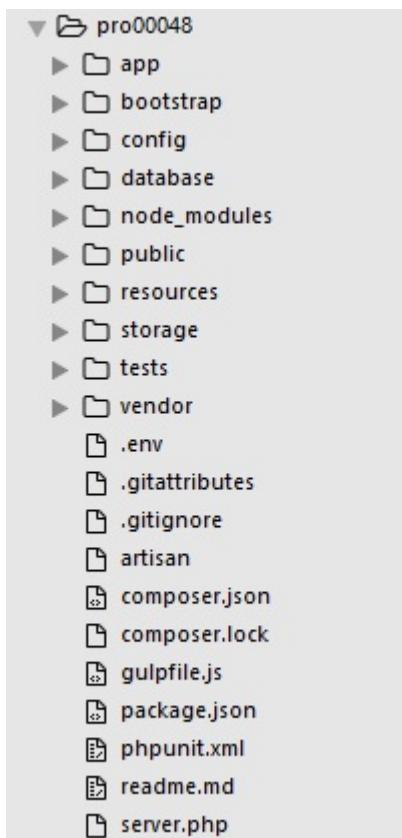
- Ubuntu 14.04
- Git
- PHP 7.0
- HHVM
- Nginx
- MySQL
- MariaDB
- Sqlite3
- Postgres
- Composer
- Node (With PM2, Bower, Grunt, and Gulp)
- Redis
- Memcached
- Beanstalkd

Conventie over configuratie

Het framework volgt het principe van conventie over configuratie , dit wil zeggen dat laravel bijna geen configuratie nodig heeft om te starten. Het framework beperkt het aantal keuzes dat de developer moet maken door default configuratie waardes te gebruiken. Uiteraard kunnen deze default waardes verandert worden.

Applicatie structuur

Laravel heeft wat beperkingen als het gaat over applicatie structuur. Het forceert de ontwikkelaar een bepaalde mappenstructuur te volgen. In het begin is het vrij moeilijk om mee te werken omdat het project wel wat folders telt maar dit heeft een groot voordeel. Omdat elke ontwikkelaar geforceerd word om dezelfde structuur aan te houden resulteert dit op een mappenstructuur die in elk laravel project hetzelfde is. Als er nu later iets moet geïmplementeerd worden dan moet men geen tijd verspillen met uit te zoeken waar alles staat.



Figuur 2: Applicatie structuur van laravel project.

App: de core code van de applicatie , dit houd in : models, controllers, middleware,..

Bootstrap: enkele files voor het configureren van autoloading, ... (framework files)

config: folder waarin alle configuratie files te vinden zijn. Bijvoorbeeld: database, mail, authenticatie, ...

database: folder waarin alle migration en seeder files te vinden zijn, deze worden gebruikt om aan database management te doen en de database te vullen met data.

node_modules: folder die alle packages bevat die binnengehaald zijn door de npm package manager.

public: public folder, root folder van server dient hiernaar gemapt te worden

storage: deze folder bevat door framework gegenereerde files en applicatie log files.

tests: deze folder bevat al de tests van het project.

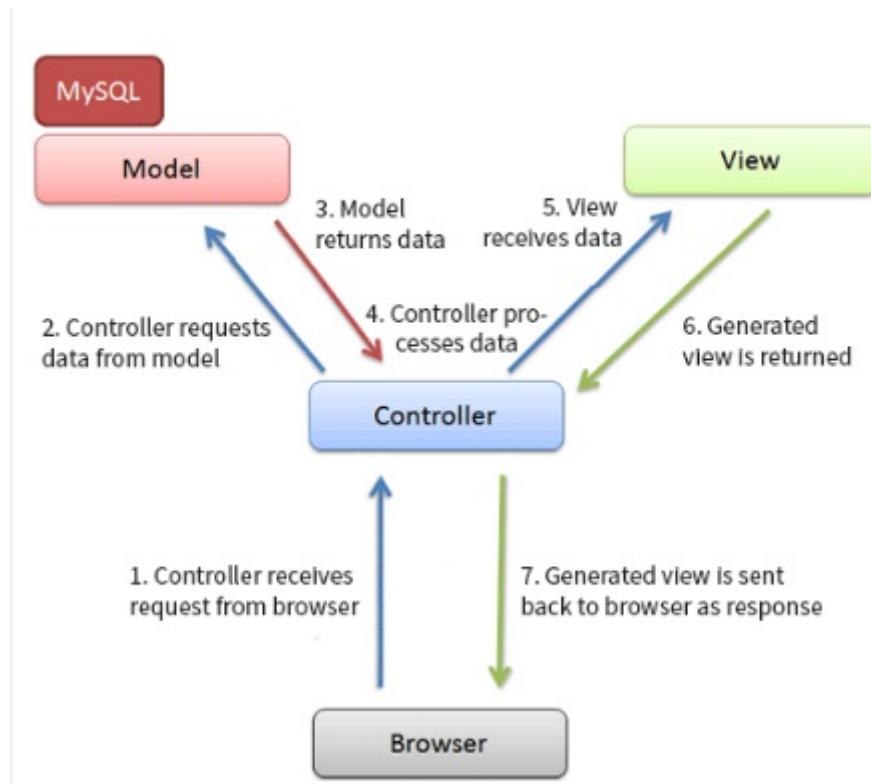
vendor deze folder bevat alle dependencies die zijn ingevoerd met composer.

Model-View-Controller

Laravel volgt het MVC (model-view-controller) design pattern , wat wil zeggen dat onze applicatie in 3 delen word gesplitst.

- **Model** Het model representeert de data. Het is volledig onafhankelijk van de view of controller.
- **View** De view geeft de model data weer, users kunnen interacteren met de view en kunnen hiermee requests sturen naar de server waardoor er een controller actie getriggerd kan worden zoals bijvoorbeeld data opvragen en terugsturen naar de view.
- **Controller** De controller doet het ontvangen , interpreteren en valideren van user input, data opvragen en updaten van views. Het is een scheiding tussen de view en het model (applicatie logica).

Door dit design pattern te volgen kan de applicatie logica worden gescheiden van de presentatie logica. Deze aanpak zorgt voor een vermindering aan complexiteit in de applicatie. Ook zal het er voor zorgen dat de applicatie heel veel code kan hergebruiken omdat alles gescheiden is van elkaar.



Figuur 3: Laravel MVC pattern

Artisan

Artisan is een command-line tool die al sinds laravel 3 bij het framework zit , Artisan bied vele commands aan die de ontwikkelaar kunnen helpen om verschillende taken uit te voeren. Hieronder een overzicht van de taken die wij hebben gebruikt in ons project:

- **db:seed**
 - Vult de database met dummy data om zo de applicatie te kunnen testen met werkelijke data in de database
- **make:Auth**
 - genereert de authenticatie logica en views.
- **make:controller**
 - genereert een controller met boilerplate code.
- **make:middleware**
 - genereert middleware met boilerplate code.
- **make:model**
 - genereert een model met boilerplate code.
- **make:seeder**
 - genereert een seeder met boilerplate code.
- **make:migration**
 - genereert een migration met boilerplate code.
- **migrate**
 - opzetten van de databank aan de hand van migrations.
- **migrate:refresh**
 - na het toevoegen van een nieuwe migration moest deze command worden uitgevoerd om de databank volledig opnieuw op te zetten.
- **session:table**
 - Creeërt migration voor de sessie tabel in de DB.

Door het gebruik van Artisan hebben we heel wat tijd uitgespaard omdat er heel veel skeleton code gewoon kan gegenereerd worden.

Composer

Composer is een dependency manager voor PHP, Laravel maakt gebruik van deze dependency manager om zijn dependencies te behandelen.



Figuur 4: Composer logo

Door een composer.json file in de root van het project aan te passen kunnen we dependencies toevoegen aan ons project door het `composer install` command uit te voeren. Dit biedt als voordeel dat sommige functionaliteit zo kan geïmplementeerd worden zonder dat dit helemaal van 0 moet geprogrammeerd worden. Zo hebben wij enkele dependencies gebruikt in ons project:

- **Laravel collective Forms** is sinds laravel 5 niet meer deel van het framework , als men hier gebruik van wilt maken moet deze dependency toegevoegd worden
- **Intervention image** Een open source php bibliotheek waarmee men afbeeldingen kan editeren.
- **laravel-gravatar** Bibliotheek om makkelijk random gravatars te genereren.
- **purifier** Bibliotheek om HTML te purifiern , dit wil zeggen uitfilteren van bepaalde tags (bv:script tags)

Composer wordt ook gebruikt om Laravel te installeren. Eerst installeren we de Laravel installer globaal zodat we eender waar een Laravel project kunnen aanmaken dit doen we door deze command uit te voeren. `composer global require "laravel/installer"` Door dit te doen kunnen we eender waar een Laravel project aanmaken door de `Laravel new "projectnaam"` command uit te voeren.

Bootstrap

Voornamelijk voor het grid-systeem en enkele componenten hebben we het Front-end framework bootstrap gebruikt. Het is een combinatie van HTML , CSS en JavaScript code bedoelt om de ontwikkelaar te helpen met het bouwen van user interfaces . Het grote voordeel van Bootstrap is dat het ontwikkel process veel sneller verloopt. Dit framework bied ons design templates die we kunnen gebruiken om onze UI te maken. Er zijn onder andere :forms, buttons, tables, carrousel, ... layouts die meteen kunnen geïmplementeerd worden in het project.



Figuur 5: Bootstrap logo

Bootstrap beschikt ook over een geweldig responsive grid systeem dat eenvoudig is in gebruik. Het grid telt 12 kolommen die je kan gebruiken om elementen een bepaalde breedte te geven aan de hand van css klasses. Deze css klasses bestaan voor verschillende beeldformatten zodat men de layout kan definiëren op verschillende devices. Zo bestaan er css klasses voor :

extra small devices	small devices	medium devices	large devices
.col-xs-	col-sm-	col-md	col-lg
auto	750px	970px	1170px

Het framework is ook cross-platform dus de output zal altijd hetzelfde zijn. Het maakt dus niet uit in welke browser je het project openst.

Om de applicatie performanter te maken kan er op de bootstrap website ook een gecustomizede versie gemaakt worden zodat de developer kan kiezen wat er allemaal gebruikt moet kunnen worden. Dit resulteert in een kleinere css/js file. De applicatie gebruikt moet dus geen onnodige bootstrap componenten laden.

Om Bootstrap toe te voegen aan ons project gebruiken we de npm package manager. Eens deze package is geïnstalleerd voegen we deze toe aan onze main Sass file zodat dit mee gecompileerd word in de publieke css file. Hetzelfde voor de Javascript file van Bootstrap, de Bootstrap js file word gerequired in de main.js file. Gulp zal dan via de Browserify task dit compileren naar de publieke js file.

Sass

Sass is een extensie van CSS3 , omdat web applicaties en websites uitgebreider worden met de jaren kunnen we gebruiken maken van deze CSS preprocessor om onze stylesheets modulair en overzichtelijker te maken.



Figuur 6: Sass logo

Waarom ? Omdat Sass enkele nieuwe features aanbied die men momenteel niet kan gebruiken bij normale CSS , zoals :

- Variabelen: Door deze feature kunnen we een aantal variabelen aanmaken die we zullen gebruiken doorheen het maken van de applicatie zoals : kleuren ,fonts , margins , line-heights , breakpoints. Het handige aan deze feature is dat als de klant wilt dat bv. een kleur of bv de margins wilt veranderen dan moet dit maar op 1 plaats aangepast worden.
- Partials Deze feature staat ons toe om onze stylesheets op te delen in verschillende delen zodat we onze css op een heel modulaire manier konden schrijven. Dit is mogelijk door partials aan te maken die allemaal worden toegevoegd aan een main.scss bestand op deze manier : `@import('filepath')` . De partials worden aangeduid met een `_` voor hun naam zodat sass weet dat dit partials zijn en niet mee moeten gecompileerd worden. Wat er nu wel gecompileerd word is de main.scss file waarin alle partials worden toegevoegd. Zo maakten we voor alles dat logisch gescheiden kon worden van elkaar een partial , dit zorgt voor een overzichtelijker project. Het voordeel van alles te compileren in 1 file is dat het aantal HTTP requests enorm zal dalen tegenover al deze verschillende partials apart.
- Mixins Omdat sommige css declaraties veel terugkomen kan men gebruik maken van mixins , dit zijn groepjes van css declaraties die men kan hergebruiken doorheen de website.
- Extends Met extend kan je de css declaraties van een klasse extenderen naar een andere klasse. bv als je een button maakt dan kan je deze extenderen en zo een groene en rode knop klasse maken die de button klasse extenderen. Op deze manier moet je veel minder code schrijven.

Gulp

Gulp is een Javascript taskrunner die de ontwikkeling van websites kan versnellen door het automatiseren van algemene taken zoals : compileren van preprocessed CSS (sass) , het minifyen en bundelen van JavaScript , reloaden van de browser,...



Figuur 7: Gulp logo

De installatie gaat als volgt:

- Installeren van Node.js.
- Gulp installeren aan de hand van de package manager van Node.js genaamd npm.

Gulp maakt gebruik van plug-ins die men kan installeren om specifieke taken uit te voeren. Deze plugins kan men installeren via de npm package manager. Hierna kan men taken declareren in de file 'gulpfile.js' die in de root folder staat. De plug-ins die men wil gebruiken moeten toegevoegd worden bovenaan deze file , daarna worden de taken gedeclareerd. Voor deze applicatie zijn er enkele taken die moeten worden uitgevoerd namelijk :

- compileren van sass files.
- compileren van js files.

Voor de taken te declareren gebruiken we Laravel Elixir , dit is een soort wrapper rond Gulp. Elixir laat ons toe om op een simpele manier taken te declareren in Gulp, het ondersteunt ook enkele taken dus moeten we deze niet meer toevoegen via npm.

Taken die wij gebruiken van Elixir:

- **Sass compiler** Compileert Sass naar Css in public folder
- **Browserify** Laat ons toe om dependencies te definieëren zodat browserify dit kan bundelen in één enkele js file => zorgt voor betere performance omdat er maar 1 enkele .js file moet geladen worden.
- **Copy** Kopiëren van files naar een andere map.

Bij gulp kan men gebruik maken van streams , dit wil zeggen dat men files door een stream kan sturen van verschillende taken. Deze taken zullen dus allemaal uitgevoerd worden op de files die in de stream worden gestoken. Gulp kan taken uitvoeren door een bepaald commando in te voeren of zal kijken naar veranderingen in de files die je declareert in de taken. Als er nu iets verandert in deze files zal Gulp de nodige taken automatisch uitvoeren.

Bespreking ontwikkeling

In dit hoofdstuk word in grote lijnen uitgelegd hoe de applicatie in elkaar steekt en hoe alles samenwerkt met elkaar.

Database

We hebben eerst een database schema (bijlage 1) gemaakt omdat we zo een overzicht kregen van de data en relaties in de applicatie. Voor er connectie mogelijk is moesten we de configuratie file aanpassen in de config folder van het laravel project:

```
'mysql' => [
    'driver' => 'mysql',
    'host' => env('DB_HOST', 'localhost'),
    'port' => env('DB_PORT', '3306'),
    'database' => env('DB_DATABASE', 'pitchpoint'),
    'username' => env('DB_USERNAME', 'root'),
    'password' => env('DB_PASSWORD', 'root'),
    'charset' => 'utf8',
    'collation' => 'utf8_unicode_ci',
    'prefix' => '',
    'strict' => false,
    'engine' => null,
],
```

MySQL

Momenteel ondersteunt Laravel 4 database systemen: MySQL, Postgres, SQLite en SQL server. Wij hebben gebruik gemaakt van een MySQL database.

migrations

Migrations laten ons toe om gemakkelijk in een team met een databank te werken. Telkens een nieuwe table moet aangemaakt worden moet er een migration file worden toegevoegd. Als nu een andere developer wilt werken aan het project zal hij zijn database kunnen bijwerken of aanmaken door het `php artisan migrate` command uit te voeren. Ook een handig voordeel met het werken van migrations is dat als er iets misloopt met de databank dan kan deze helemaal opnieuw aangemaakt worden met Artisan. Migrations zijn een soort van version control van je databank , het houd een geschiedenis bij in de databank van wat er allemaal al gemigrated is. Als er nu nieuwe migrations bestaan die nog niet in deze tabel staan dan zullen deze migrations uitgevoerd worden bij het migraten met Artisan.

pitchpoint.migrations: 30 rijen totaal (ongeveer)

migration	batch
2014_10_12_000000_create_users_table	1
2014_10_12_100000_create_password_resets_table	1
2016_03_07_170634_create_levels_table	1
2016_03_07_170837_create_services_table	1
2016_03_07_170900_create_languages_table	1
2016_03_07_170919_create_markets_table	1
2016_03_07_170943_create_comments_table	1
2016_03_07_171007_create_strongpoints_table	1
2016_03_07_171037_create_domains_table	1
2016_03_07_171057_create_expertises_table	1
2016_03_07_171124_create_employees_table	1
2016_03_07_171150_create_references_table	1
2016_03_07_171210_create_awards_table	1
2016_03_07_171237_create_clients_table	1
2016_03_08_100843_create_agencies_table	1
2016_03_08_143704_add_contactPerson_to_agencies_t...	1
2016_03_14_160446_add_other_to_expertises_table	1
2016_03_22_141622_add_other_to_domains_table	1
2016_03_23_163656_add_company_to_references_table	1
2016_03_25_130649_change_datatype_startDate_end...	1
2016_04_06_064943_change_nullable_agency_id_on_ta...	1
2016_04_08_133241_add_differentiators_to_agencies	1
2016_04_15_132042_add_register_step_to_users	1
2016_04_19_071243_add_original-image-name_to_agen...	1
2016_04_19_073317_add_original-picture-name_to_emp...	1

Figuur 8: migration table in de database

Migrations kunnen gegenereerd worden met artisan en bestaan uit telkens 2 functies :

- **up** Hierin worden er nieuwe tables , kolommen of indexen toegevoegd aan de databank.
- **down** Hierin worden de nieuwe tables, kolommen of indexen die in de up methode werden aangemaakt ongedaan gemaakt.

Seeders

Om de applicatie te testen is het handig dat er wat test data gebruikt kan worden. Dit kan verwezenlijkt worden door het gebruik van seeders. Een seeder klasse kan gegenereerd worden met Artisan, deze klasse bevat maar één functie genaamd run en zal aangesproken worden als het artisan seed command word uitgevoerd. In deze functie word er data in de database opgeslagen door gebruik van de laravel query builder, men kan ook grote hoeveelheden van data random genereren door gebruik van de model factories. In het begin van het ontwikkelen hebben we gebruik gemaakt van seeders omdat we dan al data uit de database konden halen.

Hierna hebben we de models aangemaakt zodat we de ORM konden gebruiken om met de data uit de database te werken.

Model

Eloquent ORM

De Eloquent ORM is de built-in Object Relational Mapper van Laravel. De ORM voorziet een Active Record pattern implementatie, dit wil zeggen dat er voor elk model er aangemaakt word een bijhorende tabel is in de database. Dit geeft ons enkele voordelen die hieronder besproken worden.

leesbaarheid

Doordat men met objecten kan werken om data uit de databank te halen blijven alle queries overzichtelijk omdat het veel gemakkelijker leest als raw SQL queries.

relaties

Als er relaties bestaan tussen tabellen en je zou een SQL query moeten schrijven zou je elke keer volledig de relatie opnieuw moeten schrijven in SQL , bij het gebruik van Eloquent word dit gedefinieerd in het model en is daarom veel korter om te schrijven..

Om de naam van het agency van user met id=1 uit de databank te halen gebruiken we deze ORM methode:

```
User::find(2)->agency()->name;
```

Als we dit nu met een SQL query zouden doen dan zou het er zo uit zien:

```
select name from users join agencies  
where users.id = '2'
```

Zoals je kunt zien is de syntax in deze eenvoudige query veel overzichtelijker met de ORM. Als men heel veel relaties heeft in de applicatie dan zal men met de ORM veel minder code moeten schrijven en zal men dus een overzichtelijker project hebben dan met SQL queries.

SQL injectie bescherming

Eloquent gebruikt PDO parameter binding om SQL injectie te voorkomen. Dit wil zeggen dat gebruikers de query niet kunnen manipuleren door een bepaalde input te geven.

Bijvoorbeeld: een input veld waar gesearched kan worden naar een telefoon nummer met als input = '036653050'; drop table users;.

```
select * from users where phone = '036653050'; drop table users;
```

Deze query zal de users table en alle data erin verwijderen, maar als er nu PDO parameter binding zou toegepast worden dan zou de query er zo uitzien:

```
select * from users where phone = '036653050; drop table users;'
```

omdat er geen telefoon nummer bestaat dat "036653050; drop table users;" bevat zal de query niets returnen. De PDO parameter binding gebeurt allemaal behind the scene doordat Eloquent de ORM methodes vertaald in SQL queries.

Models

Models worden aangemaakt door dit command : `php artisan make:model modelName` . Het aangemaakte model zal geplaatst worden in de map `App` . In het eloquent model worden er eerst enkele dingen gedefinieerd zoals :

- **Welke attributes er mass assignable moeten zijn.**

- Als een field als fillable word gedefinieerd dan zal deze mass assignable zijn. Dit wil zeggen dat als er een input array word verstuurd wanneer de gebruiker zijn form submit, dan zullen de attributes die in deze array zitten en mass-fillable zijn in 1 keer kunnen worden opgeslagen en niet allemaal apart. Als deze

fields allemaal mass-assignable zouden zijn dan kan men iets meesturen met de input dat normaal niet verandert zou mogen worden bv: is_admin=true

- **Welke attributes er gecast moeten worden.**

- Wanneer nummers worden opgeslagen worden deze automatisch als strings gereturned, wanneer men gebruik maakt van attribute casting kan men deze casten naar een ander data type bv: int, double,...

- **Welke attributes hidden zijn.**

- als er bv: users uit de databank moeten gehaald worden dan zal het password meegegeven worden maar als er een hidden attribute password word gedefinieerd in het model dan zal dit niet meer gebeuren. Anders zou de back-end misschien gevoelige data kunnen sturen naar de front-end en dat is niet de bedoeling.

Elke Table heeft een bijhorend model dat gebruikt word om data uit de database te halen en toe te voegen. Als er nergens explicet word gedefinieerd welke table uit de database er gebruikt moet worden voor het model zal eloquent het meervoud van de modelnaam + enkel kleine letters uit de db gebruiken als table. bv: model User zal standaard de users table uit de db gebruiken.

De relaties tussen de tables moesten ook gedefinieerd worden in de Eloquente models. een Agency heeft één user maar een Agency heeft ook meerdere Werknemers . Zo zijn er wel meerdere relaties die gedefinieerd werden. Door deze relaties als functies te definieeren zal men de mogelijkheid hebben om de relaties gemakkelijk te queryen door het gebruik van method chaining. Hieronder een voorbeeld van method chaining :

```
Agency::find($id)->myDomains()
```

Om deze relaties te definieren zal men eerst moeten bepalen welk type relatie er gelegd moet worden. Zo bestaan er een aantal types (die wij gebruikt hebben):

1. **One To One relationship** Bv: one Agency has one User account => om deze relatie te definieren zetten we in het Agency model een methode user:

```
public function user()
{
    return $this->hasOne('App\User');
}
```

2. **One To Many relationship** Bv: one Agency has many Employees om deze relatie te definieren zetten we in het Agency model een methode employee:

```
public function employees()
{
    return $this->hasMany('App\Employee', 'agency_id')->get();
}
```

De relatie moet ook andersom gedefinieerd worden zodat we bv de agency kunnen achterhalen aan de hand van een employee. Dit word gedaan aan de hand van deze methode toe te voegen.

```
public function agency()
{
    return $this->belongsTo('App\Agency');
}
```

Authentication

Vanaf laravel 5.2 kan men via artisan gebruik maken van de Auth scaffold . Door het command `php artisan make:auth` uit te voeren, zal men alle views genereren die nodig zijn voor authenticatie. Deze command zal volgende views genereren:

- welcome.blade.php - the public welcome page

- home.blade.php - the dashboard for logged-in users
- auth/login.blade.php - the login page
- auth/register.blade.php - the register/signup page
- auth/passwords/email.blade.php - the password reset confirmation page
- auth/passwords/reset.blade.php - the password reset prompt page
- auth/emails/password.blade.php - the password reset email

zoals je kan zien zijn er een aantal views gegenereerd die we kunnen gebruiken in ons project.

Als we nu de Route::auth() methode zetten in de routes.php file dan zullen we de nodige routes in onze applicatie hebben om het authenticatie systeem te laten werken. Deze methode staat voor meerdere routes:

```
//authentication routes
$this->get('login', 'Auth\AuthController@showLoginForm');
$this->post('login', 'Auth\AuthController@login');
$this->get('logout', 'Auth\AuthController@logout');

// Registration Routes...
$this->get('register', 'Auth\AuthController@showRegistrationForm');
$this->post('register', 'Auth\AuthController@register');

// Password Reset Routes...
$this->get('password/reset/{token?}', 'Auth\PasswordController@showResetForm');
$this->post('password/email', 'Auth\PasswordController@sendResetLinkEmail');
$this->post('password/reset', 'Auth\PasswordController@reset');
```

Om het authenticatie systeem naar onze wensen aan te passen moesten we de registratie methode in de Authcontroller overschrijven en de \$redirectTo variabele in de Authcontroller aanpassen. Deze variabele geeft aan naar waar geredirect word na het inloggen , de uri die we in deze variabele steken wijst naar een functie in de agency controller die users zal redirecten naar de laatst ingevulde stap. Ook hebben we de login en reset password views aangepast zodat dit past in het design het project.

Routes

De gebruiker kan naar verschillende pagina's navigeren met behulp van routes. Deze routes dienen gedefinieerd te worden in de file `app/Http/routes.php`. Als er nu een request word gedaan door de client en deze is gelijk aan de gespecifieerde route uri, dan zal de server weten welke controller er moet worden aangesproken om een bepaalde functie uit te voeren en/of de juiste pagina terugsturen naar de client.

Bv: `Route::get('/{id}/step1', 'AgencyController@showStep1');`

Deze route verteld ons dat als je naar `/2/step1` surft dan zal de `id='2'` meegegeven worden als parameter in de `showStep1` functie in de `AgencyController` en dan zal de eerste stap terug gestuurd worden met alle bijhorende data. Om een overzicht te zien van al onze routes word verwezen naar bijlage 2.

resourceful routes

We hebben gebruik gemaakt van resourcefull routes , dit zijn routes die geregistreerd worden voor een resource controller. Als men een controller aanmaakt dat bijvoorbeeld requests verwerkt aangaande werknemers kan men gebruik maken van resource controllers. Deze kunnen aangemaakt worden via Artisan door het command : `php artisan make:controller EmployeeController --resource` uit te voeren. Hierna zal er een resource route worden gedefinieerd die geassocieerd word met deze controller. Deze resource route zal meerdere routes bevatten die we nodig hadden om data op te slaan , data te updaten , data te verwijderen en data weer te geven.

Verb	Path	Action	Route Name
GET	/resource	index	resource.index
GET	/resource/create	create	resource.create
POST	/resource	store	resource.store
GET	/resource/{resource}	show	resource.show
GET	/resource/{resource}/edit	edit	resource.edit
PUT/PATCH	/resource/{resource}	update	resource.update
DELETE	/resource/{resource}	destroy	resource.destroy

Figuur 9: routes die bestaan in een resourceful route

Deze routes zullen naar methodes linken in de controller. Al deze methodes worden ook al gegenereerd bij het aanmaken van de controller.

Zo hadden we enkele resourceful routes gecreeerd samen met hun bijkomende controller:

- voor werknemers van een agency
- voor awards van een agency
- voor referenties van een agency
- voor klanten van een agency
- voor commentaar over een agency
- voor sterke punten van een agency

HTTP Middleware

HTTP Middleware gebruiken we als filter om op bepaald requests een gepaste response te geven. Bv. middleware kan controleren of de persoon die de request stuurt geauthenticeerd is of niet.

Welke middleware hebben wij gebruikt in onze applicatie ?

- **web group middleware** Word gebruikt voor cookies , sessies en Cross site request forgery protection. Deze middleware word toegepast op elke route in onze applicatie omdat overall sessies en cookies en CSRF protection nodig zijn deze middleware zit ingebakken in het framework.
- **Auth middleware** Word gebruikt bij routes die enkel toegankelijk zijn voor geauthenticeerde users. Als user niet geauthenticeerd is dan zal deze een pagina krijgen met een 'unauthorized' message. Deze middleware komt ook out of the box bij het laravel framework.
- **Admin middleware** Word gebruikt bij routes die enkel toegankelijk zijn voor geauthenticeerde users met admin rechten. Deze middleware hebben we zelf geschreven.

Middleware kan aangemaakt worden via artisan door het command `php artisan make:middleware MiddlewareName` uit te voeren. Om deze middleware te gebruiken dienen we te declareren op welke routes deze middleware moet uitgevoerd worden. Dit gebeurt in de `route.php` file waar alle routes staan gedefinieerd. Men kan ook groepen maken van routes waar deze middleware moet op worden uitgevoerd. Elke keer als er nu een request verstuurd word naar een bepaalde route met middleware zal eerst de code van de middleware worden uitgevoerd voordat de controller aangesproken zal worden.

Application logic

Onze applicatie logica word uitgevoerd door de controllers. Zoals uitgelegd bij Routes zullen HTTP requests een specifieke controller functie kunnen uitvoeren. Wij hebben 2 "hoofd" controllers :

Admin controller

In de admin controller worden (bijna) alle requests verwerkt die uit het admin gedeelte van de webapplicatie komen. Als er word ingelogd met een admin account zal de index() functie uitgevoerd worden.

- **index()** Deze functie zal data van bureau's uit de databank halen, ook zal deze functie agencies rangschikken op voltooide stappen (voltooide registraties van voor). In deze functie word ook gedefinieerd dat als er meer dan 10 agencies zijn dan moeten deze gepagineerd worden. Als laatste zal deze functie de view admin.dashboard terugsturen samen met de data die hierin weergeven word.
- **show(\$id)** als er nu op een bureau word geklikt in het dashboard dan zal deze functie worden aangesproken in de controller en dan zal de controller een nieuwe view terug sturen namelijk het overzicht van het geklikte agency met alle data die nodig is om dit weer te geven. Het \$id verteld de controller op welke agency er is geklikt.
- **store(\$id , Request \$request)** In het agency overzicht zal de admin commentaar, sterke punten en een laatste contactdatum kunnen toevoegen. Dit opslaan naar de databank zal gebeuren door deze functie. een request instantie word geïnjecteerd zodat men uit de huidige HTTP request de input data kan halen en opslaan. Het \$id verteld de controller naar welk agency er moet worden opgeslagen.

Agency controller Deze controller zal zich bezig houden met het verwerken van requests aangaande het registratie proces.

- **showStep1(\$id)** deze functie zal uitgevoerd worden wanneer een user op de registreer knop drukt. De controller zal hierna de view van de eerste registratiestap terug sturen naar de client.
- **show(\$id , \$stepId)** Deze functie zal alle verschillende stappen van de registratie weergeven aan de hand van een switch statement. Als parameters gebruiken we \$id en \$stepId , \$id gebruiken we voor de correcte data van het juiste agency terug te sturen en \$stepId gebruiken we zodat de controller weet welke stap er moet weergeven worden. telkens dat deze functie word uitgevoerd zal deze ook updaten tot welke stap van het registratieprocess de gebruiker alles al heeft ingevuld , dit word allemaal opgeslagen in de database. Deze waarde word gebruikt om een afbakening te maken tot waar de gebruiker allemaal kan navigeren : de gebruiker kan navigeren tot de stap na de laatst ingevulde stap , zodat er geen stappen kunnen overgeslagen worden. Deze functie zal ook een mail sturen naar de gebruiker als deze de registratie volledig heeft doorlopen. Deze mail bevat een bedankje voor de registratie en een link waar het account kan geëditeerd worden. Ook zal er een mail gestuurd worden naar pitchpoint zelf als een nieuw agency zich heeft geregistreerd , deze mail bevat de naam van het geregistreerde agency en een link naar de admin pagina van pitchpoint. Omdat we in de view aan form model binding doen om het form te hervullen moeten we telkens bij elke stap het juiste object meesturen met de view zodat dit mogelijk word.
- **store(\$id , \$stepId , Request \$request)** Deze functie zal alle ingevoerde data van het registratie proces verwerken en opslaan in de databank. Telkens er een submit word gedaan in het registratieproces zal deze functie checken op welke stap er juist gesubmit is , hierna zal hij de juiste validatie regels in de validator steken. De data zal gevalideerd worden , als de data niet voldoet aan de regels dan zal de gebruiker geredirect worden naar dezelfde stap maar deze keer zullen er error messages meegestuurd worden zodat de gebruiker bovenaan de pagina een message te zien krijgt over oncorrect ingevulde data. Als de data wel correct gevalideerd is dan zal deze opgeslagen worden en zal de user geredirect worden naar de volgende stap.
- **downloadWorkProcess(\$id)** bevat code om een file te downloaden die geupload is door een user.
- **Redirect** redirect gebruikers die inloggen naar de laatst ingevulde stap van de registratie.
- **registerStep(\$stepId, \$user)** slaagt op in de database welke stap de laatst ingevulde stap is van de gebruiker.
- **sendSupportMail(Request \$request)** als een user gebruik maakt van het support formulier (om een vraag te stellen) zal deze functie een mail sturen naar pitchpoint met de desbetreffende vraag.

We hebben ook nog aparte controllers voor de resource routes (employees , awards, references, clients, comments, strongpoints). In employees en clients zijn er functies voor het aanmaken, updaten, verwijderen, en editeren. Bij awards , references , comments , strongpoints kan er enkel aangemaakt en verwijderd worden.

Validation

Zoals hierboven vermeld zal de controller de binnenkomende data valideren aan de hand van een aantal regels. Hoe werkt dit nu juist ? Wel, de eerste stap is het definieëren van de validatie regels. Laravel heeft enkele out of the box validatie regels die men kan gebruiken. Hieronder een kort lijstje van de validatie regels die wij gebruiken in het project.

- **required** Deze field moet worden ingevuld.
- **min:#** Deze field moet min x aantal characters hebben.
- **max:#** Deze field mag max x aantal characters hebben.
- **email** Deze field moet een geldig email adres bevatten.
- **numeric** Deze field mag enkel numerieke characters bevatten.
- **unique** Deze field moet uniek zijn in een bepaalde database table en kolom. dus bv: email adres in de users table.
- **between:#,#** Deze field moet tussen de 2 ingevulde waardes liggen.
- **integer** Deze field moet een integer bevatten.
- **confirmed** Als deze field validatie regel op bv: field password word toegepast dan zal er eenzelfde waarde in password_confirmation field moeten staan.

Deze regels worden allemaal opgeslagen in een array die mee word gegeven aan de validate methode. De validate methode accepteert 2 parameters , de http request met alle ingevoerde data en de array met de validatie regels. Als de validatie gelukt is zal de controller verder de functie uit voeren, maar als de validatie mislukt zal de controller de user terugsturen naar de huidige pagina maar deze keer worden de error messages meegegeven. De errors worden meegegeven in de sessie data:

```
if ($validator->fails()) {  
    $messages = $validator->messages();  
    return redirect()  
        ->withErrors($validator)  
        ->withInput();  
}
```

De errors kunnen in de view dan benaderd worden door de \$errors variabele te gebruiken. Wij hebben problemen hiermee gehad toen we 2 dezelfde forms op 1 pagina hadden staan. De error messages werden op beide forms weergegeven terwijl er maar 1 form ingevuld was. Dit gebeurde in stap 9 waar er vorige en huidige clienten kunnen toegevoegd worden. Het enige verschil tussen deze forms is dat de huidige cliënten geen eindjaar konden invullen. Hiervan hebben we gebruik gemaakt , in de backend word gecheckt of in de request een field zit met eindjaar. Als dit aanwezig is dan worden de validator regels voor vorige clienten in de validator geladen , als dit niet zo is dan de regels voor huidige clienten. Deze Errorbag word terug gestuurd naar de frontend met als naam previous voor vorige clienten en als naam current voor huidige clienten. Op deze manier komen de error messages op de juiste plaats en zal er correct gevalideerd worden.

In de view word aan de hand van de Blade templating engine een if geschreven dat als er een error message van een bepaalde field in de errorbag zit, dat deze field een css klasse met rode border moet krijgen. Hierdoor is makkelijk te zien waar de validatie fout zich bevindt in de form.

```
{!! Form::text('box', null, $errors->has('box') ? ['class'=>'form__text form-control has-errors'] :  
['class'=>'form__text form-control'])  
!!}
```

Views

Wij hebben in onze views gebruik gemaakt van de built-in templating engine genaamd blade. Als men blade in een view wilt gebruiken dient de view file de blade.php extentie te hebben. 2 grote voordelen die we hiermee hebben zijn het gebruik van template overerving en het gebruik van secties. Wij hebben voor onze applicatie een aantal standaard layouts gemaakt waarvan andere pagina's zullen overerven.

- **Master layout** Elke pagina erft hiervan over => deze pagina bevat meta properties, link naar css file en link naar js file.
- **Admin layout** Bevat de standaard layout van het admin gedeelte van de applicatie.
- **Simple layout** Bevat de standaard layout die gebruikt word bij het editeer scherm van Employees en Clients en bij de support en legal agreement pagina.
- **step layout** bevat de standaard layout voor de verschillende stappen van het registratieprocess.

In deze layout files hebben we sections gecreeerd aan de hand van @yield directives. Deze kunnen we in de child pages (pages die overerven) vullen aan de hand van @section directives. Deze aanpak zorgt ervoor dat al de code die in de layout files staat maar 1 keer geschreven moet worden terwijl deze voor meerdere pagina's word gebruikt. Hieronder een voorbeeld van overerving in onze applicatie.

De **master.blade.php layout** is de layout waar elke pagina van overerft , het bevat de css/js links, metadata en een yield directive die gebruikt word om de inhoud van sections uit child pages te injecteren.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>PitchPoint - @yield('title')</title>
    {{-- meta data from pitchpoint.be--}}
    <meta name="fbadmins_meta_tag" content="PitchPoint.be"/>
    <meta name="keywords" content="DNA, Merk, Pitchconsultant, audit, bureau, bureaurelaties, communicatie, communication, construction, d'agence, de, marque, reclamebureaus, s&acute;s;lection"/>
    <meta name="description" content="Selectie reclamebureaus - Managen bureaurelaties - Communicatie audit - Merk DNA S &acute;s;lection d'agence de publicit&acute;s; - relation d'agence - Audit de communication"/>
    <link rel="shortcut icon" href="http://static.wixstatic.com/ficons/7e9bf7_e30e3533bf41412a82eab350ddeab3ca_fi.ico"
      type="image/x-icon"/>
    <link rel="apple-touch-icon" href="http://static.wixstatic.com/ficons/7e9bf7_e30e3533bf41412a82eab350ddeab3ca_fi.ico"
      type="image/x-icon"/>
    <meta http-equiv="etag" content="72c88a88450dd52f345f0e4b7c7728e8"/>
    <meta property="og:title" content="PitchPoint"/>
    <meta property="og:type" content="website"/>
    <meta property="og:url" content="http://www.pitchpoint.be"/>
    <meta property="og:image" content="https://static.wixstatic.com/media/7e9bf7_6adebff6c1d142a0b3935a1dcc727a07.jpg"/>
    <meta property="og:site_name" content="PitchPoint"/>
    <meta property="fb:admins" content="PitchPoint.be"/>
    <meta property="og:description" content="Selectie reclamebureaus - Managen bureaurelaties - Communicatie audit - Merk DNA S &acute;s;lection d'agence de publicit&acute;s; - relation d'agence - Audit de communication"/>
    {{-- end meta data from pitchpoint.be--}}

    {{-- CSS --}}
    <link rel="stylesheet" type="text/css" media="all" href="/css/app.css">
    {{-- JavaScript --}}
    <script type="text/javascript" src="/js/main.js"></script>
  </head>

  <body>
    @yield('body')
  </body>
</html>
```

Figuur 10: master layout

step.blade.php layout is de layout die gebruikt word in alle stappen van het registratieproces, zoals voorheen vermeld zal deze pagina (net zoals elke pagina) overerven van de master layout. Om de yield directives van de master layout te vullen moesten er @section directives worden gedeclareerd met daarin de code die geinjecteerd moet worden.

```
@extends('layouts.master')
@section('body')
    <div class="page-wrap">
        <div class="container">
            <header>
                @include('includes.header')
            </header>
        </div>
        <hr>
        <div class="container">
            @yield('content')
        </div>
    </div>
    <div class="container">
        <footer>
            @include('includes.footer')
        </footer>
    </div>
@endsection
```

Figuur 11: Layout voor de stappen van het registratie proces

step2.blade.php Erft over van de step layout en zal de content en title sections vullen.

```
@extends('layouts.step')
<?php $agency = $viewbag['agency']?>

@section('title', Lang::get('step2.title'))
@section('content')


{!! Form::model($agency) !!}
    {{-- error handling --}}
    @include('includes.info')
    <div class="col-md-12 col-margin-top">
        @include('includes.errors')
    </div>

    {{-- company information --}}
    <div class="row">...
    </div>

    {{-- network --}}
    <div class="row">...
    </div>

    {{-- collaboration --}}
    <div class="row">...
    </div>
    <div class="row row--margin-top">...
    </div>
    <div class="row">...
    </div>
    {!! Form::close() !!}
</div>
@endsection


```

Figuur 12: Stap 2 view

Als er nu een view gestuurd moet worden naar de client zal de templating engine één view renderen aan de hand van deze 3 blade files.

data weergeven

Viewbag

Telkens er een pagina word geladen door de controller zal de controller data meesturen die weergegeven moet worden in de view. Deze data sturen we mee in de vorm van een multidimensionale array genaamd de viewbag. Hieronder een voorbeeld van de viewbag voor het weergeven van register stap 2.

```

array:3 [▼
  "agency" => Agency {#279 ▶}
  "languages" => array:5 [▶]
  "employees" => Collection {#284 ▼
    #items: array:1 [▼
      0 => Employee {#282 ▼
        #fillable: array:9 [▶]
        #connection: null
        #table: null
        #primaryKey: "id"
        #perPage: 15
        +incrementing: true
        +timestamps: true
        #attributes: array:16 [▼
          "id" => 1
          "firstName" => "Nick"
          "lastName" => "Lauwerijs"
          "picture" => ""
          "previousEmployers" => "AP"
          "brands" => "Mobistar"
          "phone" => "0494 79 15 93"
          "email" => "nlauwerijs@prophets.be"
          "startYear" => 2016
          "function" => "developer"
          "agency_id" => 2
          "language_id" => 3
          "created_at" => "2016-05-17 16:14:53"
          "updated_at" => "2016-05-17 16:14:53"
          "deleted_at" => null
          "original_filename_picture" => ""
        ]
      ]
    #original: array:16 [▶]
    #relations: []
    #hidden: []
    #visible: []
    #appends: []
    #guarded: array:1 [▶]
    #dates: []
    #dateFormat: null
    #casts: []
    #touches: []
    #observables: []
    #with: []
    #morphClass: null
    +exists: true
    +wasRecentlyCreated: false
  }
]
]

```

Hoe word dit nu weergegeven in de view ? Als we alle werknemers willen weergeven zullen we met een foreach loop door de array met index employees lopen.

```

@foreach($viewbag['employees'] as $employee)
  @include('includes.preview.employee')
@endforeach

```

Dit wil zeggen dat voor elke werknemer in de viewbag men de employee view moet includen en in deze employee view renderen we de werknemer data. De naam geven we weer op deze manier: `$employee->name`

Dubbele curly braces

Zonder het gebruik van een templating engine zouden we telkens de data moeten echoën op deze manier:

```
<p><?php echo $employee->firstname; ?></p>
```

maar doordat blade alles dat tussen dubbele curly braces vertaald in echo's hoeft dit niet meer. Dit geeft de template een nettere syntax en zorgt ervoor dat er minder code moet worden geschreven.

Als men dubbele curly braces gebruikt dan zal de templating engine automatisch de content escapen. Dit wil zeggen dat er bescherming is tegen cross site scripting , als een gebruiker nu html tags invoert en deze worden terug weergegeven dan zullen alle HTML entiteiten geescaped worden.

We hadden ook een rich text editing functie toegevoegd waarbij we de content die weergegeven werd niet mocht geescaped worden omdat de opmaak zichtbaar moest zijn. Hiervoor konden we dus niet de templating engine gebruiken omdat deze alle tags escaped, dus om dit op te lossen maakten we gebruik van een package genaamd purifier. Deze package zal alle html tags die gebruikt worden voor de opmaak toelaten en alle andere tags zullen geescaped worden. Omdat dit in de backend niet goed werkte (cleanen voordat opgeslagen word) word de content nog eens gecleaned in de view. Dit doen we op deze manier: `{!! clean($agency->philosophy) !!}`

Lang files

De statische content van de pagina renderen we door het gebruik van language files. Op deze manier is het heel makkelijk om meerdere talen toe te voegen aan de applicatie , momenteel was dit nog niet noodzakelijk maar als het toch ooit eens geïmplementeerd moet worden is dit niet zoveel extra werk. Een language file bestaat uit een return met een array in met de statische content. voorbeeld van lang file:

```
return [  
  
    'edit' => [  
        'title' => 'Edit Client',  
  
        'b1' => 'Save',  
        'b2' => 'Cancel',  
  
        'b3' => 'Upload logo',  
        'b3a' => 'Select logo',  
  
        'e1' => 'Only .jpg, .jpeg, .gif and .png formats are supported.',  
    ],  
];
```

De titel kan men dan weergeven in de view op deze manier : `@lang('client.edit.title')`

Data opslaan

Data ingeven doen we aan de hand van forms. Om forms te gebruiken in het laravel framework moeten we een dependency toevoegen via composer omdat dit niet meer in de core van het framework zit sinds versie 5.0.

Form model binding

Omdat alle gegevens die worden ingegeven geëditeerd moeten kunnen worden, hebben we ervoor gezorgd dat het form automatisch ingevuld word op basis van het object dat meegestuurd word naar de view. Dit hebben we gedaan aan de hand van model forms, een model form word geopend op deze manier: `{!! Form::model($agency, array('files' => true, 'id'=>'new_agency', 'class'=>'form--top-page')) !!}`

Als hierna een form element gedefinieerd word met als naam hetzelfde als een attribuutnaam van het bijhorende model(\$agency) dan zal dit form element gevuld worden met data uit de database. Op deze manier kan de user heel het account editeren.

Er is 1 moment wanneer het form niet mag gevuld worden met data uit de databank namelijk na een validatie fout. Om de gebruiker een aangename gebruikservaring te geven worden na validatiefouten de forms gevuld met de data die gevalideerd werd zodat makkelijk aangepast en terug gesubmit kan worden. Als er in de sessie data een item is met dezelfde naam als een input veld dan zal dit sessie-item gebruikt worden om het inputveld te vullen.

Data sturen naar controller en opslaan naar de databank

Wanneer een form word ingevuld moet er op de save knop gedrukt worden om de data op te slaan en naar de volgende stap te gaan. Vanaf op deze knop word gedrukt zal er een POST HTTP request worden verstuurd naar de server. Aan de hand van de gedeclareerde route URI's zal een functie in de controller worden uitgevoerd. Om met de request te werken moeten we een instantie van de Request klasse injecteren in de controller. Dit doen we door het te type-hinten in de constructor.

```
public function store($id, $stepId, Request $request)
```

Deze instantie bied ons enkele methodes die we kunnen gebruiken om de input data te behandelen zoals:

```
$input = $request->all();
$request->has('name')
```

Al de data die in \$input zit en fillable is kan gesaved worden op deze manier:

```
$agency->fill$input);
$agency->save();
```

De data die niet fillable is moet men apart opslaan bv:

```
$newPassword = $input['password'];
$user->password = bcrypt($newPassword);
$user->save();
```

Making the application dynamic

Om onze web applicatie interactief en dynamisch te maken hebben we gebruik gemaakt van javascript/Jquery. Hierdoor kunnen we bv. inputs limiteren, animeren van elementen, reageren op mouseclicks zonder page refresh, berekeningen doen,... Zo hadden we heel wat javascript bestanden die we samen bundelen met behulp van browserify. Dit helpt ons om beter onderhoudbare code te schrijven door het opsplitsen van verschillende javascript files die uiteindelijk gecompileerd worden naar 1 main javascript bestand. Dit laat ons toe om op een modulaire manier code te schrijven. Hoe gebeurt dit nu juist ? We hebben een main.js bestand in de root van de javascript folder , hierin requiren we andere javascript files. Als we nu gulp runnen dan zal browserify al deze javascript files compileren in 1 groot javascript bestand. Welke verschillende javascript files hebben we allemaal gebundeld en wat is hun functie in ons project ?

- **awards.js** animatie voor het weergeven van het awards form voor een award toe te voegen.
- **bootstrap-dropdown.js** code voor de multiselect die gebruikt word voor het selecteren van services voor klanten van een agency.
- **checkUrl.js** word gebruikt voor het formateren van een url input.
- **clients.js** animatie voor het weergeven van het clients form om een client toe te voegen + het toevoegen van een has-errors klasse om een red border weer te geven bij validatie fout , hier moest dit gedaan worden met javascript omdat dit niet werkte bij de multiselect op de gewone manier (via de templating engine).
- **dashboard.js** enkele animaties voor het tonen/hiden van buttons + een check-all functie voor de search filters in het dashboard.
- **disableButton.js** word gebruikt om buttons op een disabled state te zetten.
- **employee.js** animatie voor het weergeven van het employee form voor een employee toe te voegen.
- **enableEditor.js** Het enablen van een rich text editing functie op sommige textarea fields.
- **expertises.js** bevat de logica om expertises te selecteren met een bepaalde level , eenmaal deze is geselecteerd kan er een percentage aan deze expertise worden meegegeven. In deze js file staat ook een teller die controleerd of het totaal van de geselecteerde percentages gelijk is aan 100%. Als dit het geval is zal de save button actief worden.

- **fallbackDatepicker.js** Fallback voor elke browser die geen html datepicker ondersteunt. Deze js file zal detecteren of de browser het datepicker element ondersteunt , zoniet dan zal deze een jquery ui datepicker laden.
- **imagePreview.js** Zorgt ervoor dat als de user een image selecteerd dat deze instant gepreviewd word op de pagina.
- **jquery-ui.min.js** jquery-ui file voor datepicker functie.
- **limitInput.js** word gebruikt om inputs te limiteren zodat er op een bepaalde char count geen chars meer kunnen worden ingegeven + de gebruiker zal bij het typen van de laatste 10 chars een counter te zien krijgen met de resterende chars die nog kunnen getyped worden.
- **markets.js** bevat de logica om percentages te tellen van verschillende markets. als het totaal gelijk is aan 100% dan zal de save button actief worden.
- **mobile-nav.js** animatie en weergave van de navbar als deze mobiel word weergeven.
- **modernizr-custom.js** Tool die gebruikt word om te checken of een bepaald html element ondersteunt word door de browser , een custom versie zodat het js bestand zo klein mogelijk blijft.
- **numericInput.js** zorgt ervoor dat op sommige velden enkel numerieke characters kunnen worden ingegeven. Zorgt ook voor het formateren van telefoon nummers en geld bedragen.
- **print.js** zorgt ervoor dat de admin kan aanklikken wat er mee moet afgeprint worden en wat niet.
- **references.js** animatie voor het weergeven van het references form om referenties toe te voegen.
- **RichtextEditor.js** bevat de Rich text editor.
- **saveWarning.js** zorgt ervoor dat als men form aanpassingen heeft gedaan en als men ergens naar wilt navigeren dat de user een melding krijgt die zegt dat niet alles is opgeslagen met dan als keuze om door te gaan of op deze pagina te blijven.
- **smoothScroll.js** zorgt voor een smooth scroll effect.
- **vat.js** formateren van het vat veld.

Design

Het design van de applicatie hebben we gekregen in de vorm van een psd. Dit is een photoshop bestand waar heel het design van de applicatie opstaat. Het is een guideline om onze CSS en HTML te schrijven.

Bespreking van de applicatie

In dit hoofdstuk word de werking van de applicatie besproken aan de hand van afbeeldingen.

hoofdpagina

Dit is de hoofdpagina van de applicatie, hier heeft gebruiker de keuze om in te loggen of om de registratie van een communicatie bureau te starten. Op deze pagina kan men ook naar de passwoord reset pagina gaan zodat de gebruikers hun wachtwoord kunnen resetten aan de hand van een mail. Ook hebben de gebruikers de mogelijkheid om naar de support pagina te gaan om daar via een form een mail te sturen naar de mensen van pitchpoint als ze enige vragen hebben.

registratie stap 1

Op de eerste stap moet de gebruiker zowel algemene info ingeven over het communicatiebureau als de login gegevens om het account terug te kunnen editeren. fields zoals phone/website/VAT Nr worden live geformateerd naar het juiste formaat, ook kunnen geen letters getypt worden in fields waar enkel nummers verwacht worden,dit komt de gebruikservaring ten goede. Als er nu een geregistreerd communicatie bureau zijn wachtwoord wilt aanpassen dan zal er een extra field staan namelijk Old password zodat eerst het vorige passwoord moet worden ingegeven voordat het pas verandert kan worden.

The name field is required.

Agency name*

Figuur 13: validation error in view

Als validatie niet gelukt is zal er een error message worden weergeven bovenaan de pagina , de velden waar de validatie is mislukt zullen een rode border krijgen zodat makkelijk gezien kan worden waar de foute input zit.

registratie stap 2

Op stap 2 moet de gebruiker info ingeven over de grootte van het communicatiebureau, of ze deel zijn van een netwerk en of ze samenwerkingen hebben met externe partners. In de netwerk en samenwerking fields kan men de tekst opmaken aan de hand van een rich text editing functie die op de textarea's staat.

registratie stap 3

In stap 3 moet men de belangrijkste werknemers ingeven. Op deze pagina kan men meerdere werknemers toevoegen. Door op de add new knop te drukken zal er een form verschijnen om de werknemer gegevens in te geven samen met 2 knoppen: een cancel en add knop. Alle fields zijn verplicht in te vullen buiten de foto , als er geen foto word geselecteerd dan zal er een random generated gravatar worden toegevoegd als foto.

als er een werknemer word toegevoegd zal de pagina refreshen en zal de werknemer te zien zijn op de view. Als er nu geklikt word op de werknemer zal deze openschuiven om meer info te laten zien en de gebruiker de optie te geven om de werknemer te verwijderen of aan te passen. Als men op aanpassen klikt dan zal men geredirect worden naar een editeerpagina waar alle gegevens van de werknemer aangepast en opgeslagen kunnen worden.

registratie stap 4

In deze stap krijgt het bureau de mogelijkheid om wat meer te vertellen over zijn bedrijfsfilosofie , en hoe het werk proces van het bureau in elkaar zit. Dit kan ingegeven worden aan de hand van 2 text area's met rich text editing functie. Ook is er de mogelijkheid tot het uploaden van een bestand zodat er op deze manier ook info kan gegeven worden over het werk proces.

registratie stap 5

In deze stap kan het bureau enkele expertises aanduiden samen met een bepaald expertise level. Als deze expertises worden aangeduid zullen deze verschijnen bij de sectie omzet percentage, ook kunnen er custom expertises worden toegevoegd door de 'other' velden te gebruiken. Bij omzet percentage zal het bureau moeten aanduiden hoeveel percentage van de omzet elke expertise opbrengt aan de hand van een slider. Links naast de sliders staat de resterende percentage die gebruikt kan worden in de sliders als die niet 0 is staat dit in het vet rood. De save knop is niet actief vooraleer het percentage 0 is.

registratie stap 6

Bij deze stap moet het bureau aanduiden in welke markten het bureau ervaring heeft alsook hoeveel procent van de omzet ze verdienen in elk van deze markten. Net zoals bij stap 5 zal men het resterende percentage links van de markten kunnen zien en zal de save knop niet actief zijn vooraleer het resterende percentage 0 is.

registratie stap 7

In deze stap moet het bureau specifieke talenten beschrijven , ook moet het bureau zichzelf onderscheiden en beschrijven waarom ze uniek zijn in het main differentiators veld. Beide velden heeft ook een rich text editing functie zodat de tekst opgemaakt kan worden.

registratie stap 8

Op deze pagina kunnen bureau's hun awards en referenties ingeven. Men kan awards of referenties toevoegen door op het bijhorende + icoontje te drukken, hierdoor zal er een kader openschuiven waarbij de gegevens ingegeven kunnen worden. Onderaan deze kader staat een knop om de award/werknemer op te slaan of om het toevoegen te annuleren. Als er een award of referentie toegevoegd word zal de pagina refreshen en zal de toegevoegde award/referentie hier te zien zijn. Deze zijn niet meer aan te passen maar kunnen verwijderd worden door op delete award of delete reference te drukken.

registratie stap 9

Op deze pagina kan men de huidige en vorige clients ingeven door op de bijhorende add new knop te drukken. Als je drukt op deze knop zal er een kader openschuiven waar alle gegevens kunnen worden ingegeven. Hier weer hetzelfde als bij stap 3, als er geen Logo is geselecteerd dan zal er een random generated gravatar als Logo worden gebruikt. Als er nu een client word toegevoegd dan zal pagina refreshen en zal deze te zien zijn in de view. Als men nu op de client klikt dan zal deze openschuiven om meer info te laten zien en dan heeft men de optie om de client aan te passen of te verwijderen. Als men op aanpassen drukt zal men naar een editeerpagina worden geredirect waar de gegevens aangepast en opgeslagen kunnen worden.

registratie stap 10

Als het bureau op deze pagina komt wilt dit zeggen dat de registratie is afgelopen. Het bureau krijgt nu een overzicht van alle ingevoerde data , en kan deze aanpassen als dit nodig is door terug naar de desbetreffende stap te gaan. Het bureau krijgt ook de mogelijkheid om dit overzicht via een knop af te drukken. Bij deze laatste stap krijgt het bureau ook een mail met een bedankje voor de registratie. Pitchpoint krijgt ook een mail maar dan om Pitchpoint op de hoogte te houden dat er een nieuw bureau zich heeft geregistreerd.

Navigatie

De gebruiker kan navigeren naar de stap na de laatst gesavede stap. Als de gebruiker de registratie tot en met stap 6 heeft ingevuld en je logt opnieuw in dan zal je automatisch naar stap 7 worden geredirect dit zorgt ervoor dat als de registratie niet volledig was afgerond en er opnieuw ingelogd word er meteen verder kan gegaan worden met de registratie.

Navigeren kan via de Previous en Save buttons op elke stap of via de navigatie balk bovenaan elke pagina.

Enkel de Save button zal de input op de pagina opslaan er word dus NIETS opgeslaan als men naveert via de navigatie balk. Om te voorkomen dat er dan toch info zou verloren gaan omwille dat mensen navigeren (en denken dat alles opgeslagen word) via de navigatiebalk, word er een alert weergegeven wanneer er geklikt word op de navigatiebaar als er iets van form input verandert is.



Figuur 14: Alert wanneer iemand naveert zonder te saven

Als er op OK geklikt word dan zal de user naar de geklikte pagina genaveerd worden , als er op Annuleren word gedrukt dan zal de user op de huidige pagina blijven.

Admin pagina: Dashboard

Op deze pagina krijgt Pitchpoint een overzicht van alle geregistreerde communicatiebureau's, er is een search functie beschikbaar waarmee je naar bureau's kunt searchen of men kan bureau's filteren aan de hand van specifieke expertises/markets. Vanaf dat er meer als 10 bureau's verschijnen op deze pagina , zullen deze gepaginate worden. Dit wil zeggen dat er een paginering plaats gaat vinden met 10 bureau's per pagina. Als een bureau niet volledig geregistreerd is dan zullen deze zich achteraan de lijst bevinden en zal hun kader grijs zijn ipv groen zodat pitchpoint kan zien dat de registratie niet volledig is.

Admin pagina: Bureau overzicht

Wanneer nu op een bureau word geklikt zal er een overzicht geopend worden dat ongeveer gelijk is aan het overzicht die het bureau te zien krijgt op het einde van de registratie.

Enkel krijgt de admin enkele extra opties

- aanduiden of employees mee moeten afgedrukt worden of niet.
- toevoegen van commentaar
- toevoegen van sterke punten
- toevoegen van contact historiek
- toevoegen laatste update
- navbar rechts van de pagina waarmee men kan navigeren naar het dashboard, het bureau kan editeren, overzicht kan afdrukken.

The screenshot shows the homepage of the PitchPoint website. At the top left is the PitchPoint logo. The page is divided into two main sections: 'Register' on the left and 'Login' on the right.

Register Section:

- The title 'Register' is at the top.
- A paragraph of text explains the agency's specialization in communication agencies and optimization of client-agency relations.
- A note states that the more information provided, the better the recommendation to clients.
- A paragraph about the website's purpose: providing detailed information on the agency for clients to create longlists in a pitch process.
- A note about requesting registration if not yet registered.
- A green button at the bottom says 'Register your company'.

Login Section:

- The title 'Login' is at the top.
- A note informs users that they can log in with their username and password if registered.
- Input fields for 'Email' (containing nlauwerijjs@prophets.be) and 'Password' (containing five asterisks).
- A green 'Login' button.
- A link 'Forgot your password?'.

Figuur 15: Hoofdpagina

Contact information

Provide the agency coordinates and the person PitchPoint can contact in order to request extra information or updates.

Agency name*



Agency logo

Upload logo

Street* No* Box

Zip* City*

Phone*

Website*

Contact person*

Function*

Legal company name

VAT Nr

 BE

E-mail*

Repeat e-mail*

Password*

Repeat password*

* required

Save >

Figuur 16: register stap 1

Agency size

When was the agency launched, what was the turnover of the last fiscal year (media not included), the total budget you managed last year for all your clients and how many employees and freelancers do work for the agency?

Start year*	2000
Turnover last year*	€ 1
Number of employees*	1
Number of freelancers*	1
Total budget managed for clients last fiscal year*	€ 1

Network*

Are you part of an international network? If yes, please provide the name of the network and -if applicable- the holding. Shortly describe the role of your agency in the network (e.g. Belgian representative, international lead on ..., etc...).

B I U

test test

Collaboration*

Do you have a fixed collaboration with other external partners, agencies (including media agencies), providers? If yes, please identify these stakeholder and the nature of the expertise and collaboration.

B I U

test test test

* required

< Previous

Save >

Figuur 17: register stap 2

Key employees

Who are the key employees in your agency? Please provide as a minimum: CEO/MD, Management Team, Strategic Director/Planner, Creative director, Account director, ... For each person provide the requested data, including previous employer(s) and the brands worked for. A small picture of the team member is handy, but not required.

Veerle De Vos is added to Prophets

+ Add new



Nick Lauwerijs
developer



Veerle De Vos
Developer

< Previous

Save >

Figuur 18: register stap 3

Last name*	First name*
De Vos	Veerle
Mother tongue*	Picture
Nederlands	<input type="button" value="Select file"/>
Previous employer(s)*	Brands worked for*
Thomas More	Crelan
characters left	characters left
Phone Number*	Email*
0495 25 48 51	vdevos@prophets.be
At agency since*	Job title*
2016	Developer
<small>* required</small>	
<input type="button" value="Cancel"/>	<input type="button" value="Add"/>

Figuur 19: Form om werknemers toe te voegen (stap 3)

Agency philosophy*

Describe the core philosophy of your agency. What is the agency 'credo'? What are the fundamental 'beliefs'? What is the agency 'culture'?

B I U

orem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra non, feugiat a,

Work process*

How do you prefer to cooperate with your clients? Do you have specific organization structures, workflows, processes or tools?

B I U

orem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra non, feugiat a,

Select file

* required

< Previous

Save >

Figuur 20: register stap 4

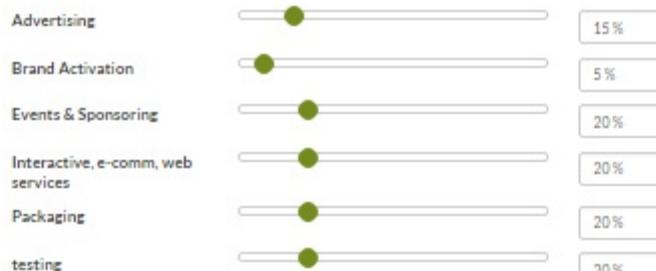
Agency expertise*

Indicate the key expertises of your agency.

Advertising	Experienced
Brand Activation	Experienced
Corporate Communication	Select expertise level
Design	Select expertise level
DM/eDM/CRM	Select expertise level
Events & Sponsoring	Basic
HR Advertising	Select expertise level
Integrated campaigns	Select expertise level
Interactive, e-comm, web services	Basic
Internal communication	Select expertise level
Market research	Select expertise level
Media agency	Select expertise level
Packaging	Experienced
PR	Select expertise level
Retail	Select expertise level
Social Media	Select expertise level
testing	Through partner
Other	Select expertise level
Other	Select expertise level

Turnover percentage*

There is 0 % available to divide over the selected expertises.



* required

Figuur 21: register stap 5

Markets*

Indicate the markets where the agency is experienced in. Please do make choices, do not tick all boxes!

There is 0% available to divide over the selected markets.

Click next to continue.

	% of turnover
<input type="checkbox"/> Beauty-Hygiene	0 %
<input checked="" type="checkbox"/> BtB	20 %
<input checked="" type="checkbox"/> Clothes & accessoires	20 %
<input type="checkbox"/> Culture/ Leisure/ Tourism/ Sports	0 %
<input checked="" type="checkbox"/> Distribution	20 %
<input type="checkbox"/> Energy	0 %
<input checked="" type="checkbox"/> Food	20 %
<input type="checkbox"/> House & Office equipment	0 %
<input type="checkbox"/> NGO	0 %
<input checked="" type="checkbox"/> Pets	20 %
<input type="checkbox"/> Services	0 %
<input type="checkbox"/> Telecom	0 %
<input type="checkbox"/> Transport	0 %
<input type="checkbox"/> Others (specify)	0 %

* required

< Previous

Save >

Figuur 22: register stap 6

Specific skills

Indicate any specific extra skills of the agency.

B I U

Lorem ipsum dolor sit amet consectetur adipiscing elit. Donec convallis ligula eget dolor. Ut eu massa. Cum sociis nesciunt perpendicula et magnis dir. Per conubia nostra, vixic et ridentibus mus. Donec quam felis ultricies nec ullamcorper etiam pretium quis enim. Nulla tempor quam massa quis enim.

Main differentiators*

Indicate any specific differentiators of the agency.

B I U

Lorem ipsum dolor sit amet consectetur adipiscing elit. Donec convallis ligula eget dolor. Ut eu massa. Cum sociis nesciunt perpendicula et magnis dir. Per conubia nostra, vixic et ridentibus mus. Donec quam felis ultricies nec ullamcorper etiam pretium quis enim. Nulla tempor quam massa quis enim.

* required

< Previous

Save >

Figuur 23: register stap 7

Awards

What are the main awards (max. 10) your agency won during the last year(s).

1	Award Award	Year 2016	Brand Mobistar
Delete award			

2

Award*	Year*	Brand*
<input type="text" value="I"/>	<input type="text"/>	<input type="text"/>
* required		
<input type="button" value="Cancel"/>	<input type="button" value="Add Award"/>	

References

Please provide us the coordinates of some key clients (current or previous) we (or our pitch-client) may contact as a reference for the agency services.

1	Company Company Name	Contact person Contact Name
	Phone 04 999 99 99	Email E@mail.com
	Function Function Name	Delete reference



< Previous

Save >

Figuur 24: register stap 8

Current clients

Who are the actual active clients the agency works for?

+ Add new

Company
Name
Brand Name
Since: 2015



Previous clients

Who are the prior clients the agency worked for in the last 3 years, but who are no longer agency client?

+ Add new

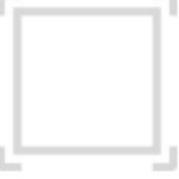
Company
Name
Brand Name
From: 2015 -
2015



< Previous

Save >

Figuur 25: register stap 9

Company*	Brand*
<input type="text"/>	<input type="text"/>
Market*	Service*
<input type="text" value="B2B"/>	<input type="text"/>
Logo	<input type="button" value="Upload logo"/>
	
Since*	<small>* required</small>
<input type="text" value="year"/>	<input type="button" value="Add"/>
<input type="button" value="Cancel"/>	

Figuur 26: Form om klanten toe te voegen (stap 9)

Thank you for registering!

Enclosed an overview of the current data of your agency in our database. Please review and update if necessary.



Prophets

Mechelsesteenweg 64b301
2018 Antwerpen
Tel: 0321616 60
www.prophets.be

Last updated: 07-06-2016 Founded: 2000 Employees: 1
Turnover last year: € 1 Freelancers: 1

[Employees](#) [Work Process](#) [Expertise](#) [Markets](#) [Skills](#) [Awards](#) [Clients](#) [References](#)

Fixed collaboration with other agencies:

test t<script> alert("collab")</script>est test

Part of a network:

test test <script> alert("network")</script>

Employees

Nick Lauwerijs
developer | 2016
Nederlands



Veerle De Vos
Developer | 2016
Nederlands



Figuur 27: register step 10 (overzicht registratie)



Thanks for the registration of your agency. We will use these data to inform our clients who are looking for new communication partners. Please do not forget to update your data on a regular basis. The quality of your data is the basis for our recommendation to our clients.

If you want to update your data, click [here](#).

For more information you can mail martine@pitchpoint.be.

Best regards, Martine Ballegeer.

© 2016 PitchPoint

[About us](#)

Figuur 28: Mail die gestuur word na user die zich registreert



Prophets has registered on the Pitchpoint Agency Tool.

Click [here](#) to check the new data.

© 2016 PitchPoint

[About us](#)

Figuur 29: Mail die pitchpoint krijgt zodat ze op de hoogte zijn van een nieuwe registratie.

All agencies

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet. Proin gravida dolor sit amet lacus accumsan et viverra justo commodo. Proin sodales pulvinar tempor. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam fermentum, nulla luctus pharetra vulputate, felis tellus mollis orci, sed rhoncus sapien nunc eget odio.

🔍

Expertises: ▾

- Check all
- Other
- Brand Activation
- DM/eDM/CRM
- Interactive, e-comm, web services
- Media agency
- Retail
- Advertising
- Corporate Communication
- Events & Sponsoring
- Internal communication
- Packaging
- Social Media
- Integrated campaigns
- Design
- HR Advertising
- Market research
- PR

Markets: ▾

Apply Filter

 **Prophets**
Last updated on: 07-06-2016

Employees: 1 Tel: 032161660Founded: 2000 www.prophets.beTurnover: € 1 2018 Antwerpen

Expertises: Other / Advertising / Brand Activation / Events & Sponsoring / Interactive, e-comm, web services / Packaging

Markets: BtB / Clothes & accessoires / Distribution / Food / Pets

 **November five**
Last updated on: 07-06-2016

Employees: Tel: +323500922Founded: novemberfive.coTurnover: € 2000 Antwerpen

Expertises:

Markets:

Figuur 30: Admin dashboard

The screenshot shows a web-based administration interface for a system named 'Prophets'. At the top left is the logo 'Prophets 2018 Antwerpen'. To its right are contact details: 'Tel: 0321616 60' and 'www.prophets.be'. Further right are the names 'Nick Lauwerijns (developer)' and 'nlauwerijns@prophets.be', followed by a 'Back to top ↑' link. A vertical sidebar on the right contains icons for navigation: a house, a pencil, a list, a chart, and a person.

Comments
Be the first to comment on Prophets.
Enter your comment below.

Strong Points
Be the first to add a strongpoint for Prophets.
Fill in the form below.

Contact history*
12/05/2016

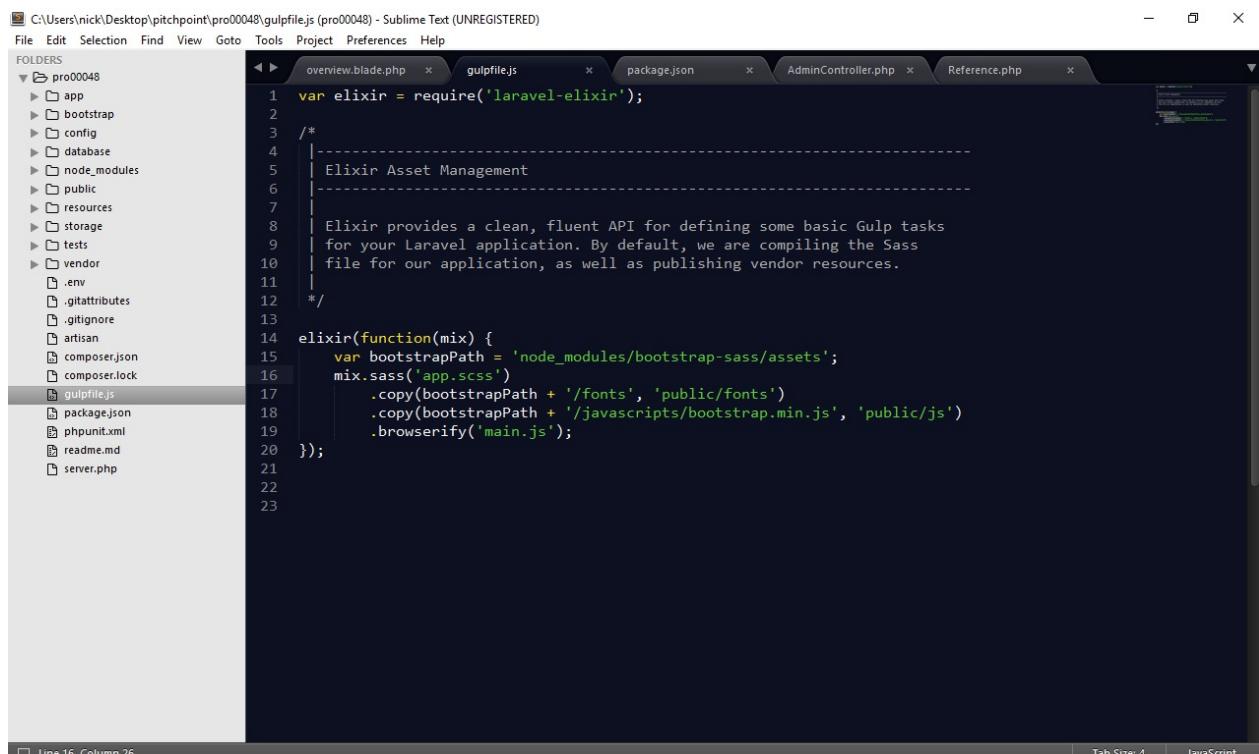
Last update
testing

Figuur 31: bureau overzicht in admin gedeelte.

Gebruikte software

Sublime Text 3

Voor dit project hadden we een text-editor nodig. Eerst maakte ik gebruik van Visual Studio Code maar omdat iedereen bij Prophets gebruik maakte van Sublime Text heb ik toch de stap gemaakt om dit ook te gebruiken omdat het makkelijker is om hulp te krijgen als ik enkele vragen had. Sublime text is één van de populairste text-editors voor web development. Het bied ondersteuning voor plugins, dit kan heel handige extra features bieden zoals css properties sorteren, code formatteren, Browser support checken bij CSS properties,... Één van de beste features van sublime is de hoeveelheid keyboard-shortcuts dat er kunnen gebruikt worden, dit is ongelooflijk handig voor de ontwikkelaar omdat het heel vlot werkt. Ontwikkelen met dit programma was een heel aangenaam proces, ik zal dit programma zeker nog in de toekomst gebruiken.



```
var elixir = require('laravel-elixir');

/*
 |--------------------------------------------------------------------------
 | Elixir Asset Management
 |--------------------------------------------------------------------------
 |
 | Elixir provides a clean, fluent API for defining some basic Gulp tasks
 | for your Laravel application. By default, we are compiling the Sass
 | file for our application, as well as publishing vendor resources.
 |

elixir(function(mix) {
    var bootstrapPath = 'node_modules/bootstrap-sass/assets';
    mix.sass('app.scss')
        .copy(bootstrapPath + '/fonts', 'public/fonts')
        .copy(bootstrapPath + '/javascripts/bootstrap.min.js', 'public/js')
        .browserify('main.js');
});
```

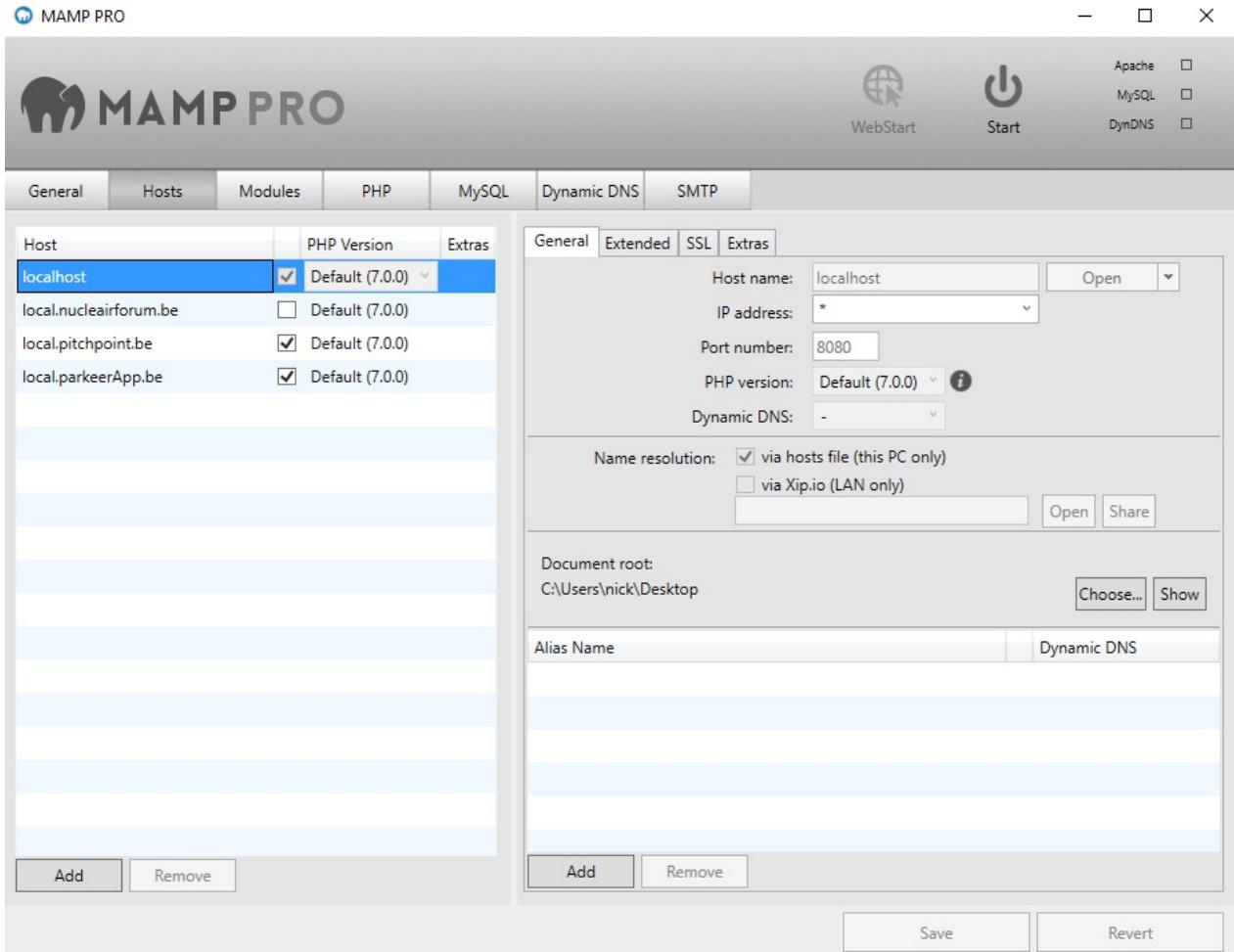
Figuur 32: Sublime text 3

Mamp pro

Om onze web applicatie te maken hadden we een ontwikkelomgeving nodig om de applicatie op te laten draaien. Dit hebben we verwezenlijkt door het gebruik van Mamp pro, dit programma bevat een aantal software componenten die we nodig hebben om onze applicatie werkende te krijgen. Deze software bundel samen met het gebruik van het windows bestuur systeem noemen we de WAMP stack.

- **Windows OS**
- **Apache server** web server
- **MySQL** Relationale database management systeem.
- **PHP** Programmaertaal van de Applicatie.

Via Mamp kunnen we meerdere virtuele servers laten draaien door verschillende hosts aan te maken. Zo kunnen we elk project openen in de browser door te surfen naar een bepaalde hostnaam en port nummer. Ik heb aparte servers gemaakt voor het pitchpoint project, de website voor nuclear forum en de parkeer app van prophets.



Figuur 33: Mamp Pro

HeidiSQL

Als grafische User Interface van de database hebben we gebruikt gemaakt van HeidiSQL. Om databases te beheren met heidiSQL moeten we inloggen op de Mamp server door de juiste credentials in te geven. Als we zijn ingelogd op de server kunnen de database zien, aanpassen, exporteren, queries op uitvoeren, ... Dit programma heb ik heel veel gebruik omdat het handig is dat je de data in de database kunt zien en aanpassen.

HeidiSQL Portable 9.3.0.4984

Bestand Veranderen Zoek Tools Help

Database filte Tabel filter ★

Host: localhost Database: pitchpoint Query

pitchpoint

Naam	Rijen	Grootte	Aangemaakt	Bijgewerkt	Engine	Commentaar	Type
agencies	1	16,0 KiB	2016-05-17 16:08:24		InnoDB		Table
awards	0	16,0 KiB	2016-05-17 16:08:24		InnoDB		Table
clients	2	16,0 KiB	2016-05-17 16:08:24		InnoDB		Table
comments	0	16,0 KiB	2016-05-17 16:08:23		InnoDB		Table
domains	5	16,0 KiB	2016-05-17 16:08:24		InnoDB		Table
employees	1	16,0 KiB	2016-05-17 16:08:24		InnoDB		Table
expertises	6	16,0 KiB	2016-05-17 16:08:24		InnoDB		Table
languages	4	16,0 KiB	2016-05-17 16:08:23		InnoDB		Table
levels	4	16,0 KiB	2016-05-17 16:08:23		InnoDB		Table
markets	14	16,0 KiB	2016-05-17 16:08:23		InnoDB		Table
migrations	30	16,0 KiB	2016-05-17 16:08:21		InnoDB		Table
password_resets	0	48,0 KiB	2016-05-17 16:08:22		InnoDB		Table
references	4	16,0 KiB	2016-05-17 16:08:24		InnoDB		Table
services	17	16,0 KiB	2016-05-17 16:08:23		InnoDB		Table
strongpoints	0	16,0 KiB	2016-05-17 16:08:23		InnoDB		Table
temporary_files	0	16,0 KiB	2016-05-17 16:08:28		InnoDB		Table
users	2	32,0 KiB	2016-05-17 16:08:22		InnoDB		Table

```

19 SHOW PROCEDURE STATUS WHERE `Db`='pitchpoint';
20 SHOW TRIGGERS FROM `pitchpoint`;
21 SELECT * FROM `COLLATION_NAME` FROM `information_schema`.`SCHEMATA` WHERE `SCHEMA_NAME` = 'information_schema';
22 SHOW TABLE STATUS FROM `information_schema`;
23 SHOW FUNCTION STATUS WHERE `Db` = 'information_schema';
24 SHOW PROCEDURE STATUS WHERE `Db` = 'information_schema';
25 SHOW TRIGGERS FROM `information_schema`;
26 SHOW EVENTS FROM `information_schema`;
27 SELECT *, `EVENT_SCHEMA` AS `Db` , `EVENT_NAME` AS `Name` FROM `information_schema`.`EVENTS` WHERE `EVENT_SCHEMA`='pitchpoint';

```

pitchpoint: 17 tables | Verbonden: 00:00 h | MySQL 5.5.41 | Uptime: 00:05 h | Inactief

Figuur 34: HeidiSQL

Besluit

Zoals je bij de resultaten kunt zien zijn alle doelstellingen die in het begin van deze scriptie zijn beschreven afgerond. De webapplicatie kan gebruikt worden door communicatiebureau's om hun data in te geven en hebben ook de mogelijkheid om opnieuw in te loggen om hun data aan te passen.

Pitchpoint heeft een admin gedeelte die kan gebruikt worden om een overzicht te krijgen van alle bureau's en heeft ook de mogelijkheid tot het aanpassen van alle data van elk bureau. Ook kan er gesearched/gefilterd worden zodat makkelijk specifieke bureau's gevonden kunnen worden.

Het was een grote verandering om een applicatie te ontwikkelen in een professionele omgeving dan op school. Ik werkte aan dit project samen met iemand die ik niet kende, waardoor in het begin de communicatie misschien een beetje stroef liep. Maar na enkele weken was dit ongeloofelijk hard verbeterd en verliep het project ook veel vlotter. Ik heb hierbij ook heel veel geleerd omdat ik nog niet zo uitgebreid PHP had gebruikt op school. Ik vond het zeker niet erg om dan ineens een PHP framework te gebruiken omdat ik hier dan veel uit kon blijven.

Ik vond het heel interessant om te werken in dit bedrijf omdat ik zo een idee heb gekregen van hoe websites/webapplicaties ontwikkeld worden in een professionele omgeving terwijl ik dit hiervoor enkel via schoolprojecten kende.

Kleinere opdrachten

In dit hoofdstuk worden 2 kleinere opdrachten, die ik heb moeten doen voor ik aan het Pitchpoint project begon, besproken.

Mail

De eerste week bij prophets moest ik met html/css een mail maken voor Carrefour, die deze dan doorstuurt naar zijn klanten (reclame dus).

Ik kreeg een HTML template waarop ik verder moest werken, deze template bestond uit heel veel tables. Waarom word een email niet hetzelfde gemaakt als een website gemaakt word ?

Het verschil tussen het opstellen van een mail en een website zit in het aantal mailclients en internetbrowsers. Er zijn heel veel mailclients tegenover internetbrowsers en al deze mailclients ondersteunen een verschillende subset van html en css . Bovendien gebruiken desktop, webmail en mobiele email clients allemaal verschillende rendering engines om de content van de mail te displayen. Om dit te illustreren kunt u een voorbeeld hieronder zien van enkele mail clients en welke css properties deze ondersteunt.

	Outlook '07	Windows Mail	Mac Mail	Entourage 2008	Thunder- bird 2	AOL 9	AOL 10	AOL Mac	Notes 6
font-family	✓	✓	✓	✓	✓	✓	✓	✓	✓
font-size	✓	✓	✓	✓	✓	✓	✓	✓	✓
font-style	✓	✓	✓	✓	✓	✓	✓	✓	✓
font-variant	✓	✓	✓	✓	✓	✓	✓	✓	✗
font-weight	✓	✓	✓	✓	✓	✓	✓	✓	✓
height	✗	✓	✓	✓	✓	✓	✓	✓	✗
left	✗	✓	✓	✓	✓	✓	✓	✓	✗
letter-spacing	✓	✓	✓	✓	✓	✓	✓	✓	✗
line-height	✓	✓	✓	✓	✓	✓	✓	✓	✗
list-style-image	✗	✓	✓	✓	✓	✓	✓	✓	✗
list-style-position	✗	✓	✓	✓	✓	✓	✓	✓	✗
list-style-type	✗	✓	✓	✓	✓	✓	✓	✓	✓
margin	✓	✓	✓	✓	✓	✗	✗	✓	✗
opacity	✗	✗	✓	✓	✓	✓	✓	✓	✗
overflow	✗	✓	✓	✓	✓	✓	✓	✓	✗
padding	✓	✓	✓	✓	✓	✓	✓	✓	✗
position	✗	✓	✓	✓	✓	✓	✓	✓	✗
right	✗	✓	✓	✓	✓	✓	✓	✓	✗
table-layout	✓	✓	✓	✓	✓	✓	✓	✓	✗
text-align	✓	✓	✓	✓	✓	✓	✓	✓	✓
text-decoration	✓	✓	✓	✓	✓	✓	✓	✓	✓
text-indent	✓	✓	✓	✓	✓	✓	✓	✓	✗
text-transform	✓	✓	✓	✓	✓	✓	✓	✓	✗
top	✗	✓	✓	✓	✓	✓	✓	✓	✗
vertical-align	✗	✓	✓	✓	✓	✓	✓	✓	✗
visibility	✗	✓	✓	✓	✓	✓	✓	✓	✗
white-space	✓	✗	✓	✓	✓	✗	✗	✓	✗
width	✗	✓	✓	✓	✓	✓	✓	✓	✗
word-spacing	✗	✓	✓	✓	✓	✓	✓	✓	✗
z-index	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figuur 35: Overzicht ondersteunde CSS properties bij verschillende mailclients

Mijn taak bestond voornamelijk uit images slicen in photoshop. En deze toevoegen aan de template, de images moesten ook nog correct gepositioneerd worden. Achteraf heb ik de mail ook nog getest in een testing tool genaamd litmus , deze tool laat het toe om mails zoals dit te testen in meer dan 40 verschillende email clients.

Uit deze opdracht heb ik geleerd dat zoveel tables moeilijk zijn om mee te werken , ik had een tag ergens vergeten te sluiten met als gevolg dat heel de pagina versprong vanaf een bepaald punt. Het is niet zo evident om een niet gesloten tag te zoeken in een pagina met 1000-2000 van deze tags. Gelukkig zijn er online tools beschikbaar die dit kunnen detecteren. Ik vond het verbazingwekkend dat een 'simpele' mail van carrefour zoveel lijnen code kon bevatten.

Website Fukushima

Als tweede opdracht moest ik een website maken voor het Nucleaire Forum van België over [Fukushima](#), omdat de feiten in Fukushima 5 jaar geleden gebeurd zijn.



Figuur 36: Website Fukushima

De website is ontwikkeld met Susy als grid systeem , Sass, Tweenmax.js en Gulp als Task runner. Er werd een design (.psd file) voorzien die nagemaakt moest worden.

Hieronder leg ik kort de technologieën uit die gebruikt zijn bij de ontwikkeling:

Susy

Susy is een set van Sass Mixins die dienen om een responsive grid op te stellen. Het leuke hieraan is dat je volledig zelf je grid definieert en dat susy al de nodige berekeningen doet. De sass compiler zal kijken naar de mixin definities in de susy bestanden en zal deze omzetten naar css. De mixin die het meeste wordt gebruikt is de span mixin :

```
.picblock {  
    @include span(4 of 12);
```

deze mixin word gebruikt om de breedte van een kolom te bepalen en word berekend aan de hand van de container mixin (wrapper). Susy staat het toe om een aantal settings in te stellen zodat je het grid helemaal kunt afstellen naar jou wensen , hieronder enkele settings die aan te passen zijn:

```
$susy: (  
    flow: ltr,  
    math: fluid,  
    output: float,  
    gutter-position: after,  
    container: auto,  
    container-position: center,  
    columns: 4,  
    gutters: .25,  
    column-width: false,  
    global-box-sizing: content-box,  
    last-flow: to,  
    debug: (  
        image: hide,  
        color: rgba(#66f, .25),  
        output: background,  
        toggle: top right,  
    ),  
,  

```

Sass

Deze technologie werd uitgelegd in hoofdstuk gebruikte technologieën.

Tweenmax.js

Er waren enkele DOM elementen die geanimeerd moesten worden. Hierbij hebben we gebruik gemaakt van Tweenmax.js Tweenmax.js is een javascript animatie library die onderdeel is van GSAP (Greensock Animation Platform) , Tweenmax is een uitbreiding van tweenlite.js. Om deze library te gebruiken steken we deze gewoon mee in de package.json file zodat deze mee word geïnstalleerd met het npm-install command , hierna hoeven we deze enkel nog te requiren in de main javascript file en alles is gereed om gebruikt te worden. Waarom GSAP ? Omdat deze library goed gedocumenteerd en dus makkelijk in gebruik is en bovendien zijn de animaties ook heel soepel (hoge framerate) dit kan je makkelijk zien met deze tool [SpeedTest](#)

Hoe werkt dit nu juist ? Een Tweenmax instantie zal 1 of meerdere properties van eender welk DOM object aanpassen over een bepaalde tijdsperiode. Bv: 2 markers die op de map moeten 'vliegen'

```
TweenMax.fromTo([".map__pin",".map__pin"] ,0.7 , {top:"40%", opacity:"0", visibility:"hidden"} , {top:"54%", opacity:"1" , visibility:"visible"} );
```

Dit stukje code zal het object aanspreken met het ID map__pin , en de properties :top:40% , opacity:0 en visibility:hidden veranderen naar top:54% , opacity:1 en visibility:visible over de tijd van 0.7 seconden. Zoals je ziet is hier helemaal niet veel code voor nodig , waardoor ik tweenmax.js heel aangenaam vind om mee te werken.

Al de animaties die gebeuren op de website zijn getriggerd op een bepaalde scroll hoogte die word gemeten vanaf de bovenzijde van de pagina maar doordat deze website is toegevoegd aan hun huidige website worden de animaties op de verkeerde moment getriggerd. Omdat ze een navigatie balk hebben toegevoegd , klopt de juiste scrollhoogte niet meer om de animaties te triggeren met als gevolg dat deze allemaal te vroeg getriggerd worden.

Gulp

Werd ook al besproken in gebruikte technologieën maar in dit project hebben we gulp iets anders gebruikt omdat we in het pitchpoint project gebruik hebben gemaakt van laravel elixir en hier niet.

Om de taken uit te kunnen voeren op verschillende files hadden we een src (=source) folder en een build folder nodig. De source folder gebruiken we om alles in te coderen en in de build folder word alle gecompileerde code gezet door middel van gulp zijn streams.

Gulp maakt gebruik van plug-ins die men kan installeren om specifieke taken uit te voeren. Deze plugins kan men installeren via de npm package manager. Hierna kan men taken declareren in de file 'gulpfile.js' die in de root van de website folder staat. De plug-ins die men wil gebruiken moeten toegevoegd worden bovenaan deze file , daarna worden de taken gedeclareerd. Een taak is opgebouwd in 3 delen :

1. Source files selecteren waarop taken moeten worden uitgevoerd.
2. Files die geselecteerd zijn worden door een pipe stream gestuurd die alle taken zal uitvoeren die gedeclareerd werden.
3. De aangepaste files zullen geplaatst worden in de build map.

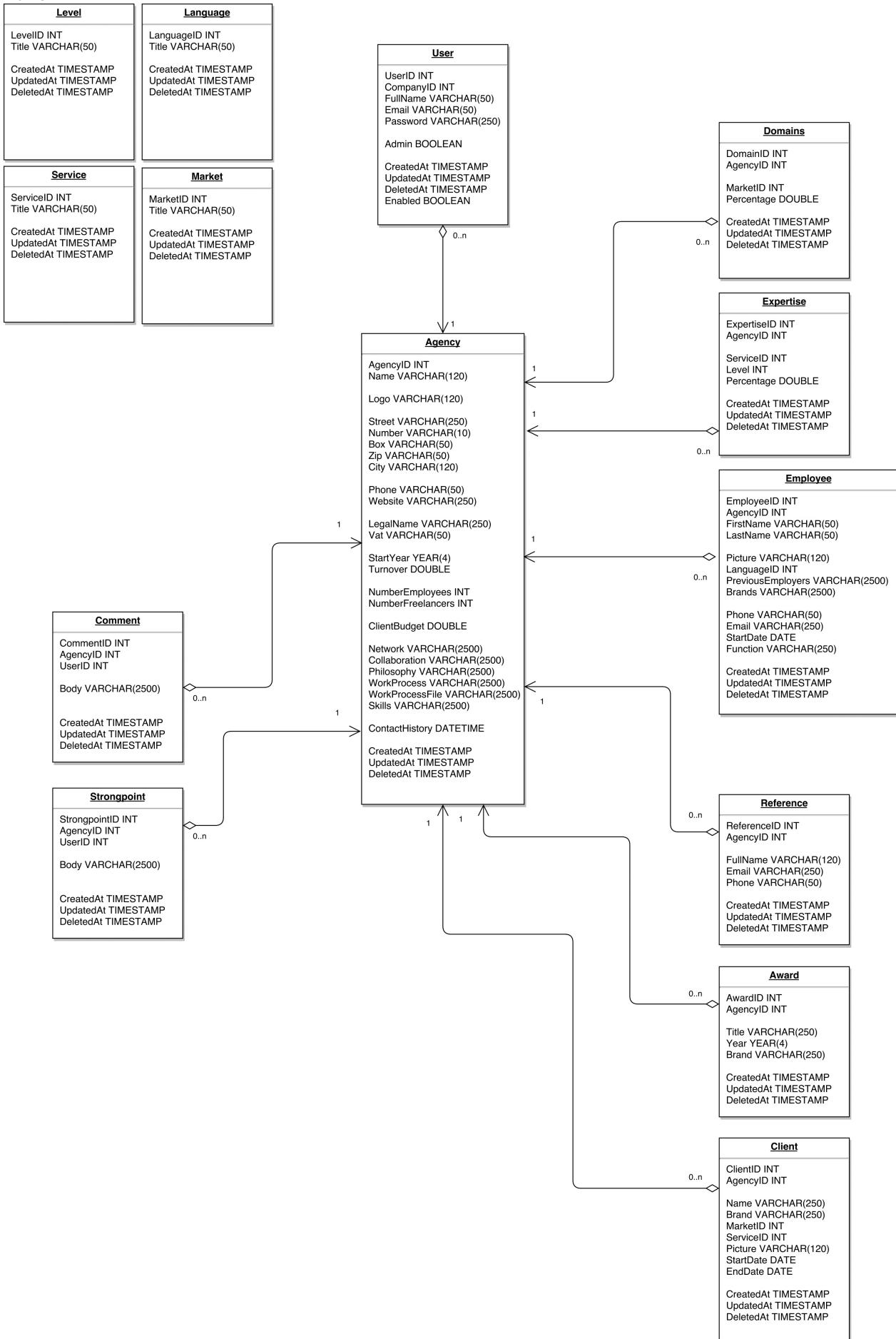
In onze gulpfile hadden we verschillende taken :

- templates
 - Content nemen uit json file en in html file toevoegen zodat we 2 json files kunnen gebruiken één voor de franse en nederlandse versie van de content. Er worden nu 2 index.html files gegenereerd in de build map 1 fr en 1 nl.
 - Alle js files toevoegen aan 1 main.js.
- styles
 - Alle partials bundelen in 1 css file.
 - Sass omzetten naar css.
 - Autoprefix, produceert code zodat alles ook ondersteunt is door oudere browsers
- watch
 - voert styles en template taak uit als er iets verandert in de projectmap.

Appendices

Bijlage 1

Bijlage 1 bevat het database schema van onze applicatie.



Bijlage 2

Bijlage 2 bevat al de routes die bestaan in onze applicatie.

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/	Closure		
	POST	@/step1		App\Http\Controllers\Auth\AuthController@register	web,guest
	POST	award	award.store	App\Http\Controllers\AwardController@store	web,auth
	GET HEAD	award	award.index	App\Http\Controllers\AwardController@index	web,auth
	GET HEAD	award/create	award.create	App\Http\Controllers\AwardController@create	web,auth
	DELETE	award/{award}	award.destroy	App\Http\Controllers\AwardController@destroy	web,auth
	PUT PATCH	award/{award}	award.update	App\Http\Controllers\AwardController@update	web,auth
	GET HEAD	award/{award}	award.show	App\Http\Controllers\AwardController@show	web,auth
	GET HEAD	award/{award}/edit	award.edit	App\Http\Controllers\AwardController@edit	web,auth
	POST	client	client.store	App\Http\Controllers\ClientController@store	web,auth
	GET HEAD	client	client.index	App\Http\Controllers\ClientController@index	web,auth
	GET HEAD	client/create	client.create	App\Http\Controllers\ClientController@create	web,auth
	DELETE	client/{client}	client.destroy	App\Http\Controllers\ClientController@destroy	web,auth
	GET HEAD	client/{client}	client.show	App\Http\Controllers\ClientController@show	web,auth
	PUT PATCH	client/{client}	client.update	App\Http\Controllers\ClientController@update	web,auth
	GET HEAD	client/{client}/edit	client.edit	App\Http\Controllers\ClientController@edit	web,auth
	POST	comment	comment.store	App\Http\Controllers\CommentController@store	web,auth,App
\Http\Middleware\Admin\Middleware	GET HEAD	comment	comment.index	App\Http\Controllers\CommentController@index	web,auth,App
	GET HEAD	client/{client}/edit	client.edit	App\Http\Controllers\ClientController@edit	web,auth
	POST	comment	comment.store	App\Http\Controllers\CommentController@store	web,auth,App
\Http\Middleware\Admin\Middleware	GET HEAD	comment	comment.index	App\Http\Controllers\CommentController@index	web,auth,App
\Http\Middleware\Admin\Middleware	GET HEAD	comment/create	comment.create	App\Http\Controllers\CommentController@create	web,auth,App
\Http\Middleware\Admin\Middleware	GET HEAD	comment/{comment}	comment.show	App\Http\Controllers\CommentController@show	web,auth,App
\Http\Middleware\Admin\Middleware	DELETE	comment/{comment}	comment.destroy	App\Http\Controllers\CommentController@destroy	web,auth,App
\Http\Middleware\Admin\Middleware	PUT PATCH	comment/{comment}	comment.update	App\Http\Controllers\CommentController@update	web,auth,App
\Http\Middleware\Admin\Middleware	GET HEAD	comment/{comment}/edit	comment.edit	App\Http\Controllers\CommentController@edit	web,auth,App
\Http\Middleware\Admin\Middleware	GET HEAD	dashboard		App\Http\Controllers\AdminController@index	web,auth,App
\Http\Middleware\Admin\Middleware	GET HEAD	download/process/{id}		App\Http\Controllers\AgencyController@downloadWorkProcess	web,auth
	POST	employee	employee.store	App\Http\Controllers\EmployeeController@store	web,auth
	GET HEAD	employee	employee.index	App\Http\Controllers\EmployeeController@index	web,auth
	GET HEAD	employee/create	employee.create	App\Http\Controllers\EmployeeController@create	web,auth
	PUT PATCH	employee/{employee}	employee.update	App\Http\Controllers\EmployeeController@update	web,auth
	DELETE	employee/{employee}	employee.destroy	App\Http\Controllers\EmployeeController@destroy	web,auth
	GET HEAD	employee/{employee}	employee.show	App\Http\Controllers\EmployeeController@show	web,auth
	GET HEAD	employee/{employee}/edit	employee.edit	App\Http\Controllers\EmployeeController@edit	web,auth
	POST	login		App\Http\Controllers\Auth\AuthController@login	web,guest
	GET HEAD	login		App\Http\Controllers\Auth\AuthController@showLoginForm	web,guest
	GET HEAD	login/redirect		App\Http\Controllers\AgencyController@Redirect	web,auth
	GET HEAD	logout		App\Http\Controllers\Auth\AuthController@logout	web
	POST	password/email		App\Http\Controllers\Auth\PasswordController@sendResetLinkEmail	web,guest

POST	password/email		App\Http\Controllers\Auth\PasswordController@sendResetLinkEmail	web,guest
POST	password/reset		App\Http\Controllers\Auth\PasswordController@reset	web,guest
GET HEAD	password/reset/{token?}		App\Http\Controllers\Auth\PasswordController@showResetForm	web,guest
GET HEAD	privacy&terms		Closure	web
GET HEAD	reference		App\Http\Controllers\ReferenceController@index	web,auth
POST	reference		App\Http\Controllers\ReferenceController@store	web,auth
GET HEAD	reference/create		App\Http\Controllers\ReferenceController@create	web,auth
PUT PATCH	reference/{reference}		App\Http\Controllers\ReferenceController@update	web,auth
GET HEAD	reference/{reference}		App\Http\Controllers\ReferenceController@show	web,auth
DELETE	reference/{reference}		App\Http\Controllers\ReferenceController@destroy	web,auth
GET HEAD	reference/{reference}/edit		App\Http\Controllers\ReferenceController@edit	web,auth
GET HEAD	register		Closure	web
POST	register		App\Http\Controllers\Auth\AuthController@register	web,guest
GET HEAD	search		App\Http\Controllers\QueryController@search	web,auth,App
\Http\Middleware\AdminMiddleware	POST	strongpoint	App\Http\Controllers\StrongpointController@store	web,auth,App
\Http\Middleware\AdminMiddleware	GET HEAD	strongpoint	App\Http\Controllers\StrongpointController@index	web,auth,App
\Http\Middleware\AdminMiddleware	GET HEAD	strongpoint/create	App\Http\Controllers\StrongpointController@create	web,auth,App
\Http\Middleware\AdminMiddleware	GET HEAD	strongpoint/{strongpoint}	App\Http\Controllers\StrongpointController@show	web,auth,App
\Http\Middleware\AdminMiddleware	PUT PATCH	strongpoint/{strongpoint}	App\Http\Controllers\StrongpointController@update	web,auth,App
\Http\Middleware\AdminMiddleware	DELETE	strongpoint/{strongpoint}	App\Http\Controllers\StrongpointController@destroy	web,auth,App
\Http\Middleware\AdminMiddleware	GET HEAD	strongpoint/{strongpoint}/edit	App\Http\Controllers\StrongpointController@edit	web,auth,App
\Http\Middleware\AdminMiddleware	GET HEAD	support	Closure	web
	GET HEAD	support	Closure	web
	POST	support/mail	App\Http\Controllers\AgencyController@sendSupportMail	web
	POST	{id}	App\Http\Controllers\AdminController@store	web,auth,App
\Http\Middleware\AdminMiddleware	GET HEAD	{id}	App\Http\Controllers\AdminController@show	web,auth,App
\Http\Middleware\AdminMiddleware	POST	{id}/award	App\Http\Controllers\AwardController@store	web,auth
	POST	{id}/client	App\Http\Controllers\ClientController@store	web,auth
	POST	{id}/client/edit/{clientId}	App\Http\Controllers\ClientController@update	web,auth
	GET HEAD	{id}/client/edit/{clientId}	App\Http\Controllers\ClientController@edit	web,auth
	POST	{id}/employee	App\Http\Controllers\EmployeeController@store	web,auth
	POST	{id}/employee/edit/{employeeId}	App\Http\Controllers\EmployeeController@update	web,auth
	GET HEAD	{id}/employee/edit/{employeeId}	App\Http\Controllers\EmployeeController@edit	web,auth
	POST	{id}/reference	App\Http\Controllers\ReferenceController@store	web,auth
	GET HEAD	{id}/step1	App\Http\Controllers\AgencyController@showStep1	web
	POST	{id}/step{stepId}	App\Http\Controllers\AgencyController@store	web,auth
	GET HEAD	{id}/step{stepId}	App\Http\Controllers\AgencyController@show	web,auth

Bibliografie

- Laravel documentatie -- <https://laravel.com/docs/5.2>
- Form model binding -- <https://scotch.io/tutorials/laravel-form-model-binding>
- Laravel collective (Forms) -- <https://laravelcollective.com/docs/5.2/html>
- The auth scaffold in Laravel -- <https://mattstauffer.co/blog/the-auth-scaffold-in-laravel-5-2>
- Laracasts (Laravel e-learning platform) -- <https://laracasts.com/lessons>
- Sass -- http://sass-lang.com/documentation/file.SASS_REFERENCE.html
- Bootstrap -- <http://getbootstrap.com/components/>
- Susy grid system -- <http://susydocs.oddbird.net/en/latest/install/>
- Composer -- <https://getcomposer.org/doc/>
- Tweenmaxjs -- <https://greensock.com/tweenmax>

Figuur bronnen

- Figuur 1: Laravel logo
Bron: <https://www.rosehosting.com/blog/install-laravel-on-centos-7/>
- Figuur 3: Laravel MVC pattern
Bron: <http://www.easylara.com/lesson-4-laravel-application-structure/>
- Figuur 4: Composer logo
Bron: <https://getcomposer.org/>
- Figuur 5: Bootstrap logo
Bron: <http://geekhmer.github.io/blog/2015/05/11/ruby-on-rails-with-bootstrap-sass/>
- Figuur 6: Sass logo
Bron: <http://sass-lang.com/>
- Figuur 7: Gulp logo
Bron: <https://github.com/gulpjs/gulp>
- Figuur 9: Resource route
Bron: <http://stackoverflow.com/questions/21978906/create-a-single-route-for-every-controller-and-method-in-laravel>
- Figuur 35: Email-clients ondersteunde CSS properties
Bron: http://www.xequate.com/support/maillistking/css_in_emails.html