



2017

Functioneel Ontwerp



Mazeyar Rezaei

Inhoud

Inleiding	2
Samenvatting	2
Doelstellingen.....	2
Over dit document	2
Risico's en grenzen	3
Scenario's	3
Structuur / Sitemaps	3
a) Database ontwerpen.....	3
b) Database realiseren	4
Functionaliteit per Pagina	4
Navigatie	5
Authenticatieproces	7
.....	8
Pagina realisatie structuur	9
Colleges pagina	9
Teamleiders pagina.....	12
Assessoren Pagina	13
Onderhoud Pagina	16

Inleiding

ROC Ter AA is een onderwijs instelling die mensen opleid met het middelbaar beroeps onderwijs. Deze onderwijsinstelling heeft een mbo examencommissie. Deze commissie zit met het probleem dat de administratie van assessoren moeilijk te monitoren is.

Het idee is om ICT technieken in te zetten om d.m.v. een soort 'tool' dit probleem aan te pakken. Deze tool zal moeten ontwikkelt worden door een ontwikkelaar.

Samenvatting

In een periode van 20 weken is een applicatie gerealiseerd gebaseerd op dit functioneel ontwerp. Deze applicatie beschikt over verschillende functionaliteiten. De applicatie beschikt verschillende talen. Sommige van deze talen zijn waren verplicht te gebruiken bij het ontwikkelen van deze applicatie.

De ontwikkelaar die de applicatie gerealiseerd heeft. Had voor deze ontwikkeling al een grote kennis betreft programmeren. Zo is in deze applicatie Laravel geïmplementeerd als technische basis. Dit is gedaan om efficiënter te vooruit te kunnen. Maar ook om sneller voortgang te tonen doordat met Laravel een hoger tempo vereist is.

Programeer keuzes hebben plaatsgevonden in gevallen om de applicatie vloeiender te laten werken, maar ook om de code overzichtelijk te houden voor de eventuele opvolgende ontwikkelaar die met deze applicatie iets zal moeten aanpassen of toevoegen.

Doelstellingen

Ontwikkel een gereedschap dat de mbo examencommissie van ROC Ter AA helpt hun werk efficiënter en makkelijker te doen. Hierbij horen de volgende doelstellingen.

1. Naar wat voor (inhoudelijk) resultaat wordt gestreefd?
2. Wat is het belang van het resultaat?

Over dit document

Bij het lezen van dit document word van de lezer verwacht dat het technisch niveau betreft ICT termen hoog ligt. Dit document zal in technische wijzen omschrijven tot welke functionaliteiten de ontwikkelde applicatie beschikt.

Er zal eerst worden omschreven welke grenzen er aan de applicatie zijn gesteld. Deze grenzen houden vooral in, voor welke platformen moet deze applicatie ontwikkelt worden, zijn er specifieke grafische wensen die de opdrachtgever in de applicatie toegepast wil hebben. Deze grenzen zullen de ontwikkelaar helpen bij het ontwerpen van een functionele basis.

Hierna zal worden uitgelegd welke functionaliteiten er per pagina toegepast zijn. Hierbij word de denkwijzen van de ontwikkelaar verwoord in een technische wijzen.

Risico's en grenzen

Scenario's

Voortdurend zijn er interviews gehouden met de opdrachtgever. Deze interviews hebben als inhoud een beeld kunnen schetsen over de algemene functionaliteiten waarvan de applicatie moet voorzien. Door dit resultaat te behalen heeft de ontwikkelaar ook een interview gehouden iemand die werkt voor de mbo examenbureau.

Deze persoon monitort voornamelijk ook de administratie van alle assessoren die zich in de instelling bevinden. In deze interview is gezien hoe het mbo examenbureau op het moment de gegevens monitort van alle assessoren binnen de instelling. Al gauw werd opgemerkt dat er niet efficiënt met de gegevens werd omgegaan.

Door deze interviews heeft de ontwikkelaar in overleg met de opdrachtgever besloten dat een webapplicatie de beste oplossing kan zijn om dit probleem aan te pakken.

Structuur / Sitemaps

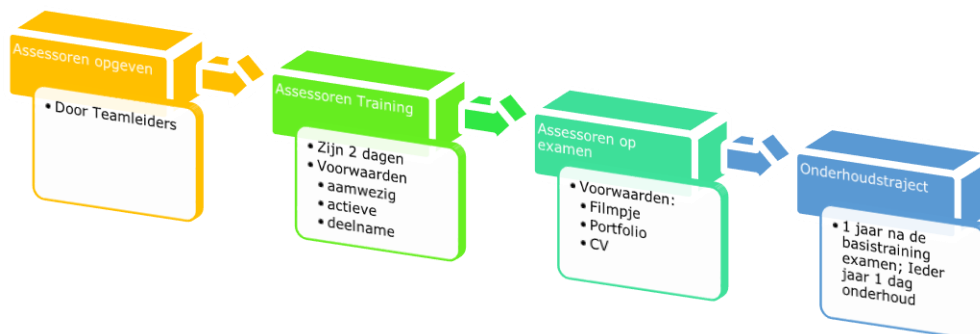
Het probleem is duidelijke, namelijk moeten de gegevens die momenteel in Excel sheets staan betreft de betrokkenen groep assessoren, in de webapplicatie verwerkt moeten worden. Om deze gegevens om te kunnen zetten in een webapplicatie, moet eerst de database structuur moeten krijgen.

a) Database ontwerpen

Het ontwerpen van de database zal tot in detail moeten worden uitgewerkt om toekomstige obstakels te voorkomen. Er word in het ontwerpen van de veel tijd besteed. Zo moeten relaties tussen objecten worden vastgesteld.

Om een duidelijk beeld te krijgen bij de denkwijzen van de ontwikkelaar worden instanties als, assessoren, colleges, teamleiders, etc. Gezien als objecten bij het ontwerpen van een database. Om deze objecten te implementeren in een webapplicatie zal eerst gekeken moeten worden naar de relaties die de objecten onderling hebben.

Bij het leggen van relaties moet eerst begrepen worden hoe een assessor, een assessor kan worden. Hierbij word verwezen naar Figuur 1.



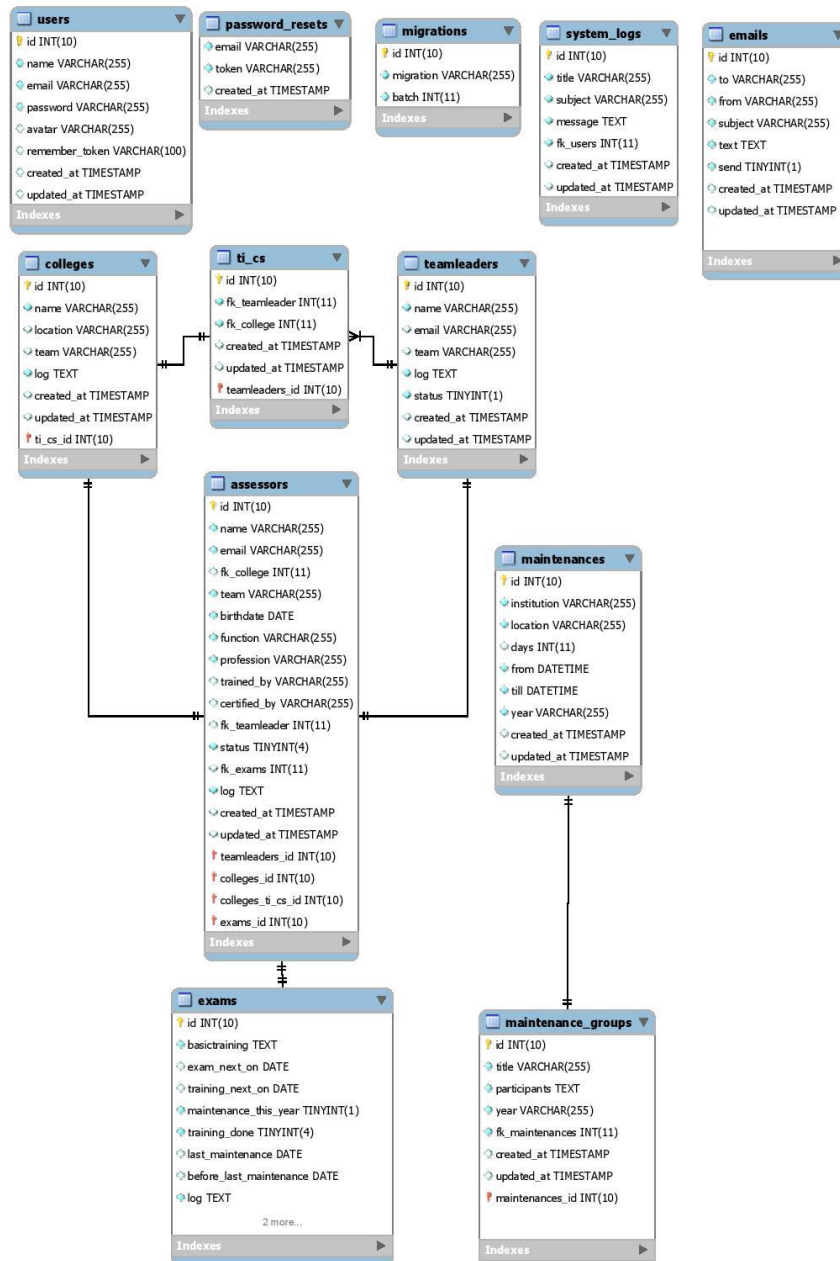
Figuur 1-Proces om assessor te worden

b) Database realiseren

Nu duidelijk is welke stappen doorlopen moeten worden om een assessor te worden kan verder gekeken worden naar de relaties die een assessor heeft met andere objecten. Een assessor is namelijk gelinkt aan een college.

Daarbij heeft een college binnen de instelling ook een teamleider, zo krijgen deze drie objecten een database relatie. Door deze relaties vast te leggen word er een Entity-relationship-diagram (ERD) opgemaakt.

Thu May 18 13:06:58 2017, New Model - EER Diagram (part 1 of 2)



1 of 2

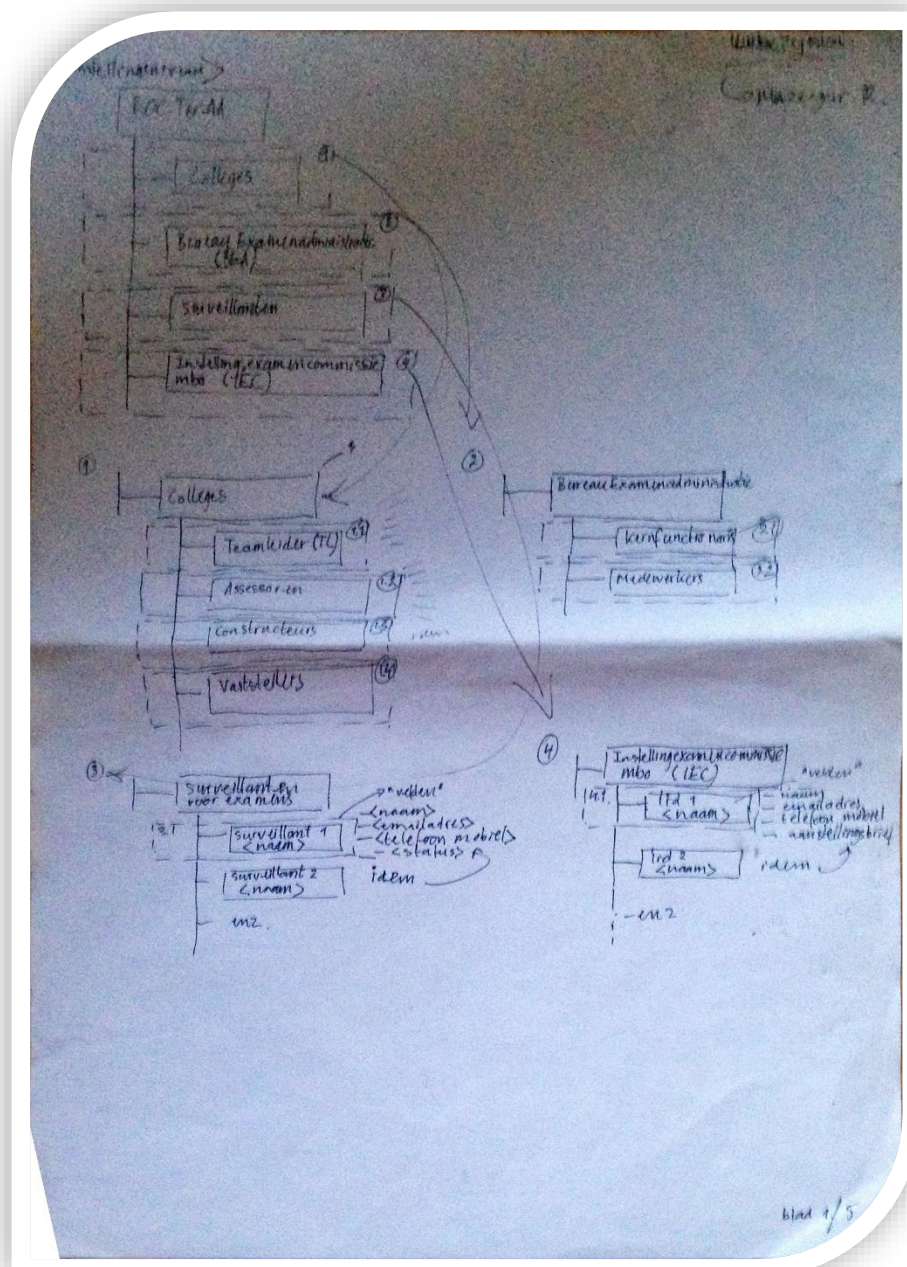
Figuur 2-ERD database schets

Functionaliteit per Pagina

Nu de database is ontworpen kunnen pagina's ontwerpt en gerealiseerd worden. Dit word gedaan met overleg met de opdrachtgever.

Navigatie

Bij het navigeren binnen de webapplicatie is aan de opdrachtgever gevraagd wat de wenselijke navigeer structuur zal moeten zijn. Hieruit is een menu structuur uitgeschetst.

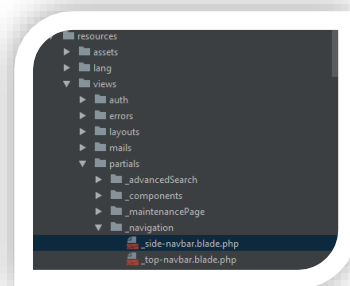


Figuur 3-Schets navigeer structuur

De navigatie structuur word van de lay-out gesplitst. De reden hiervoor is omdat de navigeer structuur meer word gezien als een component. Hierdoor word het wijzigen van de menustructuur in de toekomst gemakkelijker. Ook word code leesbaarder en compact.

Dit hierboven genoemde proces word gedaan door Blade de templating engine van Laravel. De navigatie word omgezet in een component ook wel "partial" genoemd. Deze partial word nu nogmaals gesplitst sinds deze applicatie een top en side navigatie heeft (zie Figuur 6).

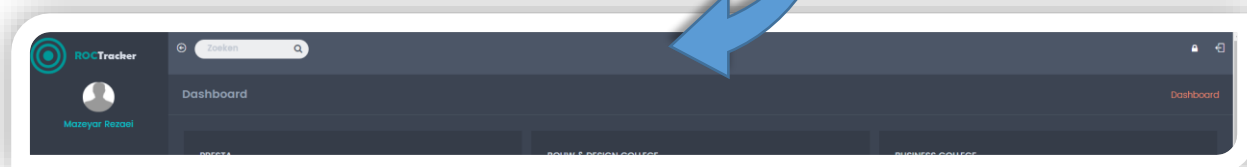
Nu de navigatie structuur is opgesplitst in componenten, kan de schets die is opgemaakt worden toegepast. Zo word de schets omgezet in code.



Figuur 6-Navigatie componenten



Figuur 5-Top navigatie component



Figuur 4-Top navigatie weergaven

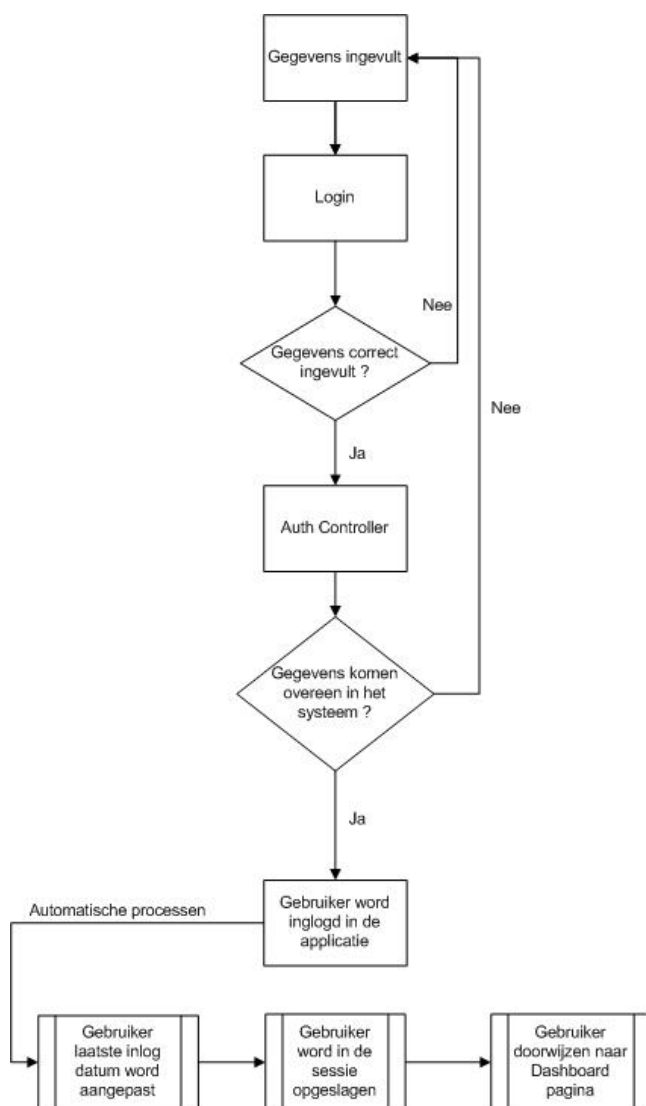
Authenticatieproces

Bij het inloggen van een gebruiker komen er veel processen bij kijken. Er zal moeten worden vastgesteld of de gebruiker die wilt inloggen in het systeem bestaat. Dit word gedaan door twee velden in te vullen, deze velden zijn als volgt:

- Email
- Wachtwoord

Deze velden zullen worden gebruikt om een gebruiker te identificeren die de gelijke waarden heeft. Ten eerste zal hier een proces plaatsvinden dat zich herhaalt op elke pagina (zie Figuur 3).

Wanneer de gebruiker zijn inloggegevens invult en op de knop "Login" klikt, zullen de gegevens door de "Auth Controller" heen gaan. Deze zal de gebruiker valideren tegen de database. Wanneer deze gebruiker is gevalideerd worde deze gebruiker opgeslagen en opgenomen in de model "Auth". De volgende flowchart zal dit proces in een algemene wijzen omschrijven.



Figuur 9-Proces bij het inloggen van een gebruiker



Figuur 7-Event handling



Figuur 8-Authenticatie weergaven

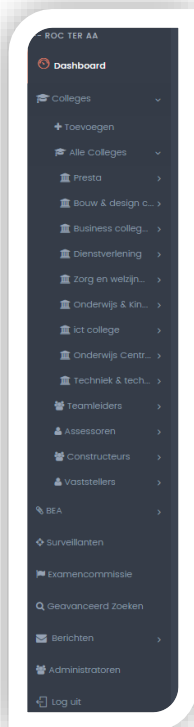
Hier kan gezien worden hoe de applicatie dynamisch een menustructuur maakt door data gegevens van colleges om te zetten in menu items.

```

TODO: SPAN ARROWS ON RESPONSIVE--}}
<ul class="nav" id="side-menu">
  <li class="sidebar-search hidden-sm hidden-md hidden-lg">
    <!-- input-group -->
    <div class="input-group custom-search-form">
      <input class="form-control" placeholder="Zoeken..." type="text"> <span class="input-group-btn"><button class="btn btn-default" type="button"><span class="fa fa-search"></span></button></span></div><!-- /input-group -->
    </li>
    <li class="nav-small-cap m-t-10">--- ROC Ter AA</li>
    <li>
      <a class="waves-effect" href="{ URL::route('dashboard') }"><i class="linea-icon linea-basic fa-fw" data-icon="a"></i> <span class="hide-menu">Dashboard</span></a>
    </li>
    <li>
      <a class="waves-effect" href="{ URL::route('colleges') }"><i class="fa fa-graduation-cap fa-fw" data-icon="a"></i> <span class="hide-menu">Colleges!! \App\Colleges::all()</span></a>
      <ul class="nav nav-second-level">
        <li>
          <a class="waves-effect" href="{ URL::route('add_college') }"><i class="fa fa-plus"></i> <span class="hide-menu">Toevoegen</span></a>
        </li>
        <li>
          <a class="waves-effect btnDoubleClick" href="{ URL::route('colleges') }"><i class="fa fa-graduation-cap"></i> <span class="hide-menu">Alle Colleges</span></a>
          <ul class="nav nav-third-level">
            <li>
              <@if(\App\College::all()->count() != 0)>
                <@foreach(\App\College::all() as $nav_colleges)>
                  <li>
                    <a class="waves-effect" href="{ URL::route('view_colleges', $nav_colleges->id) }"><i class="fa fa-bank"></i> {{ strlen($nav_colleges->name) > 30 ? substr($nav_colleges->name, 0, 30) . '...' : $nav_colleges->name }}</a>
                    <ul class="nav nav-fourth-level">
                      <li>
                        <@if(empty(\App\TiC::where('fk_college', $nav_colleges->id)->first()))>
                          <li> <a style="..." class="waves-effect" href="{ URL::route('add_teamleader', $nav_colleges->id) }"><i class="fa fa-plus"></i> <span class="hide-menu">Teamleiders toevoegen</span></a>
                        </li>
                        <li> <a style="..." class="waves-effect" href="{ URL::route('view_teamleaders', \App\TiC::where('fk_college', $nav_colleges->id)->first()) }"><i class="fa fa-users"></i> <span class="hide-menu">Teamleiders</span></a>
                        </li>
                        <li> <a style="..." class="waves-effect" href="{ URL::route('assessors', $nav_colleges->id) }"><i class="fa fa-user"></i> <span class="hide-menu">Assessoren</span></a>
                        </li>
                        <li> <a style="..." class="waves-effect" href="{ URL::route('constructors', $nav_colleges->id) }"><i class="fa fa-user"></i> <span class="hide-menu">Constructeurs</span></a>
                        </li>
                        <li> <a style="..." class="waves-effect" href="{ URL::route('detectors', $nav_colleges->id) }"><i class="fa fa-user"></i> <span class="hide-menu">Vaststellers</span></a>
                        </li>
                        <li> <a style="..." class="waves-effect btnDelete" href="{ URL::route('delete_college', $nav_colleges->id) }"><i class="fa fa-trash"></i> <span class="hide-menu">Verwijderen</span></a>
                        </li>
                      </ul>
                    </li>
                  </li>
                </li>
              </li>
            </li>
          </ul>
        </li>
      </ul>
    </li>
    <li>
      <a class="waves-effect btnDoubleClick" href="{ URL::route('teamleaders') }"><i class="fa fa-users"></i> <span class="hide-menu">Teamleiders</span></a>
    </li>
  </ul>

```

Figuur 10-Deel van Side navigatie component



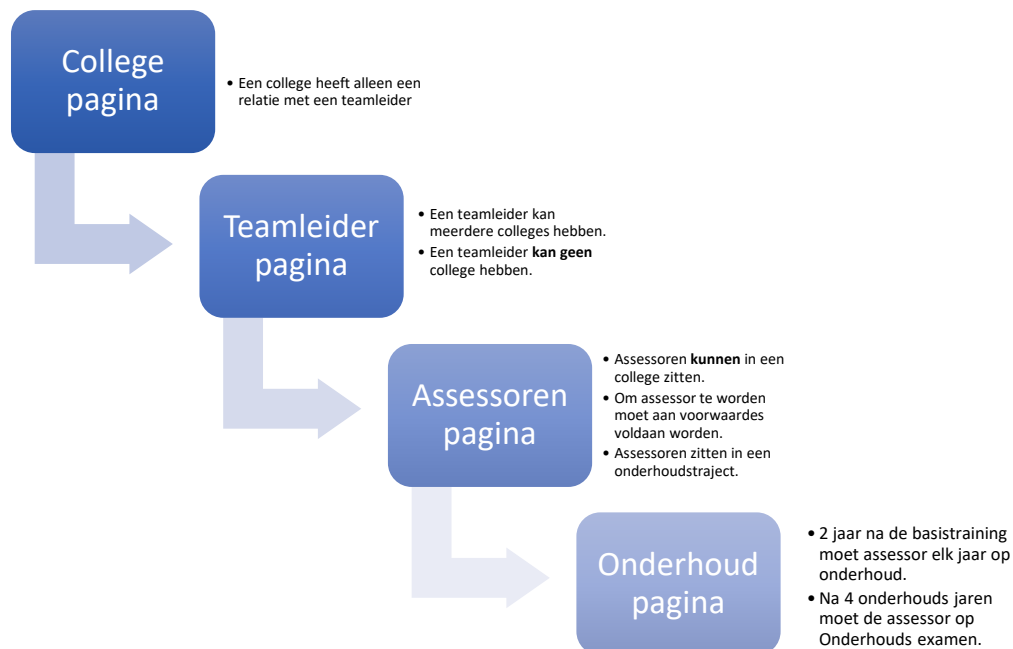
Figuur 11-Side navigatie component

Doordat de webapplicatie dynamisch is zal het aantal colleges nooit een probleem worden voor de lay-out van de pagina.

Dit komt doordat webapplicatie bootstrap gebruikt als opmaak structuur.

Pagina realisatie structuur

Er moet worden vastgesteld welke pagina's voor moeten gaan om andere pagina's te laten werken. Bijvoorbeeld de assessoren pagina kan niet worden ontwikkelt voor de college en teamleider pagina en functionaliteiten in de applicatie zitten, dit komt doordat elke assessor een college en daarbij ook een teamleider heeft. Hierdoor is de volgende structuur van realiseren opgemaakt:



Colleges pagina

Het ontwerpen van de college pagina was belangrijk voor de weergaven van de opvolgende pagina's. De ontwikkelaar heeft hier als doel een pagina te ontwerpen om gegevens te weergeven van het object op een manier, zodat de lay-out kan worden gebruikt voor alle ander pagina's die gegevens in tabel vorm weergeven moeten worden. Een tabel word geschetst met als doel een beeld te creëren, hoe en welke gegevens van het college weergeven moeten worden.

College naam	Actieve Assessoren
<ul style="list-style-type: none"> • (label) Teamleider • (label) Laatste bewerking • (button) College bekijken • (button) Bewerken • (button) Grote bewerking 	<ul style="list-style-type: none"> • (label) Aantal Assessoren

Een door de ontwikkelaar gemaakte functie "getColleges" (zie Figuur 14) zorgt ervoor dat wanneer uitgevoerd word in een compacte manier alle dataset word gemaakt van alle colleges die zijn opgeslagen in de database.

Bij deze functie kan een parameter worden meegegeven. Deze parameter moet overeen komen met het 'id' van een college dat zich in de database bevindt. Wanneer de functie wordt uitgevoerd **met** een parameter word i.p.v. alle colleges, er maar 1 terug gegeven.

```

    this Function gets all Colleges mixes them to with the assigned teamleader(s)
    The $id parameter is used for single searches. It then returns a single college)
    *
    * @param null $id
    * @return array|null
    */
    public static function getColleges($id = null)
    {
        # if there is an ID given as a parameter of this function then we only get 1 college according to the ID
        if (!is_null($id)) {
            $college = self::find($id); # Get the college object
            if (empty($college)) {
                return null; # If the object is empty we cancel this function
            }

            # We initialize our variables here
            $with_teamleaders = array();
            $assigned_teamleaders = TiC::where('fk_college', '=', $college->id)->first();

            # We now merge the teamleaders and colleges together based on the TiC table (Teamleaders in Colleges)
            if (!is_null($assigned_teamleaders)) {
                $with_teamleaders['college'] = $college;
                $with_teamleaders['teamleader'] = null;
            } else {
                $with_teamleaders['college'] = $college;
                $with_teamleaders['teamleader'] = Teamleaders::find($assigned_teamleaders->fk_teamleader);
            }
            return $with_teamleaders;
        } else {
            # else we just grab all the colleges
            $colleges = self::all();
        }

        # if there is no college existent in the database we return a (null) value
        if (empty($colleges)) {
            return null;
        }

        # We initialize our variables
    }
}

```

Hier wordt de view 'colleges' opgehaald. Daarbij wordt de 'with' functionaliteit van Laravel gebruikt om met de view een variabele mee te geven, zo wordt het eerste camel cased woord gezien als een variabele binnen de view.

In dit geval
Colleges → \$colleges

Figuur 14- "getColleges" functie

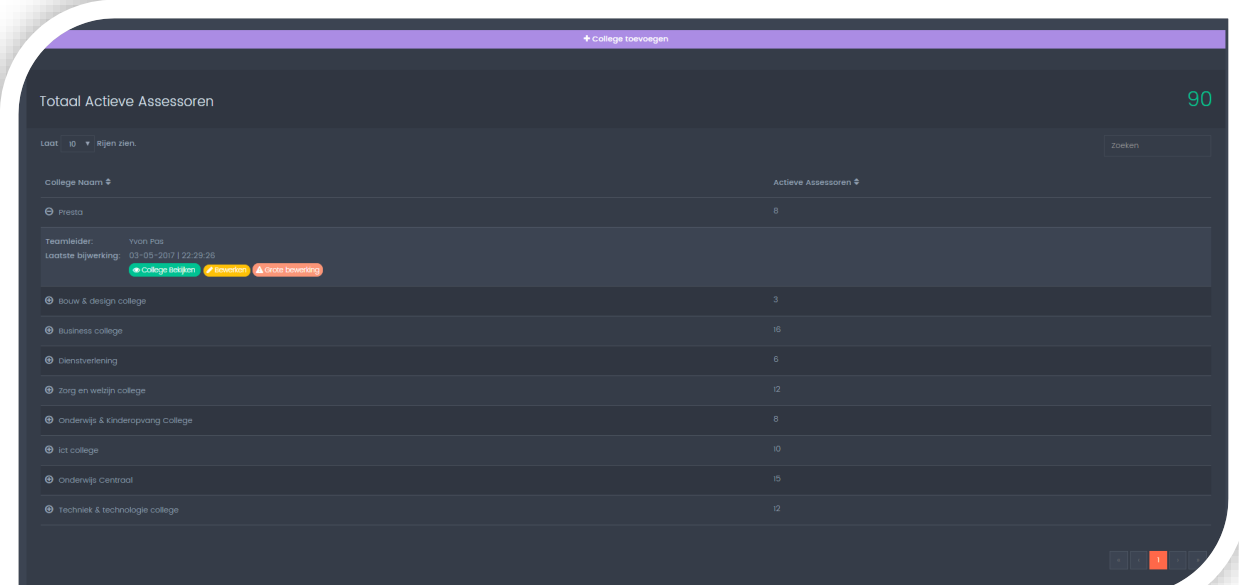
```
return view( view: 'colleges' )->withColleges( College::getColleges() );
```

Figuur 13-Voorbeeld waar "getColleges" functie uitgevoerd wordt

```
</?>
if(!empty($college))
foreach($colleges as $college)
<tr>
<td>{{ $college['college']->name }}</td>
<td>{{ \App\Assessor::where('FK_college','=', $college['college']->id)->where('status','n', 1)->count() }}</td>
<td>{{ !is_null($college['teamleader']) ? "Team" : $college['team_leader']->name }}</td>
<td>{{ date_format($college['college']->updated_at, 'd-m-Y H:i:s') }}</td>
<td>
<a href="{ URL::route('view_colleges', $college['college']->id) }}"
class="college-row-big btn-xs btn-rounded btn-success">
<i class="fa fa-eye" aria-hidden="true"></i>
College Bekijkken
</a>
<span style="..."></span>
<button data-toggle="modal" data-target="#college-little-modal"
id="{{ $college['college']->id }}"
data-name="{{ $college['college']->name }}"
data-location="{{ $college['college']->location }}"
data-team="{{ $college['college']->team }}"
class="college-row-little btn-xs btn-rounded btn-warning">
<i class="fa fa-pencil" aria-hidden="true"></i>
Bewerken
</button>
<a href="{ URL::route('change_colleges', $college['college']->id) }}"
id="{{ $college['college']->id }}"
class="college-row-big btn-xs btn-rounded btn-danger">
<i class="fa fa-exclamation-triangle" aria-hidden="true"></i>
Grote bewerking
</a>
</td>
</tr>
endforeach
endif
</tbody>
</tfoot>
```

Hier wordt een 'foreach' loop gebruikt om de gegevens die door de controller verwerkt zijn uit te lezen.

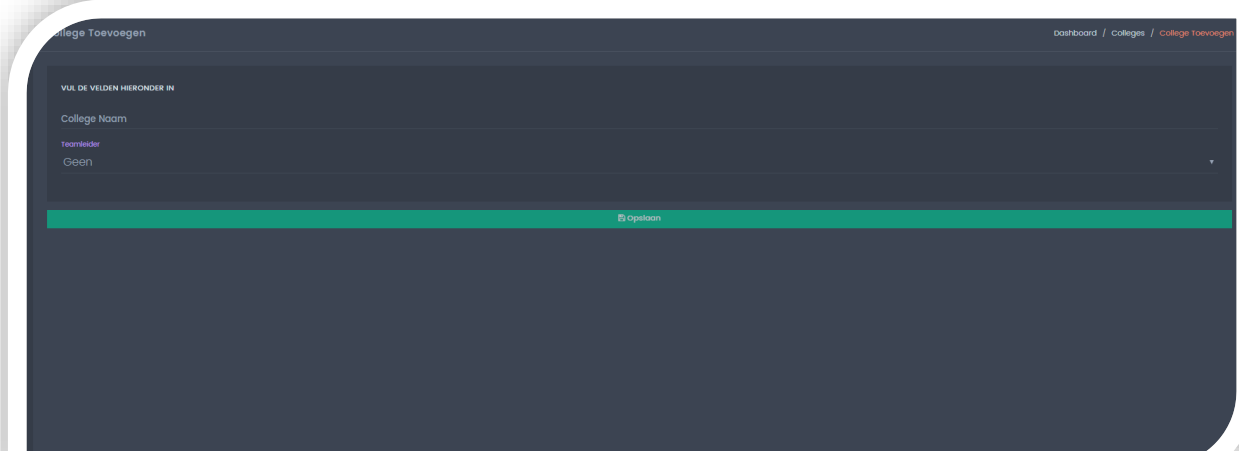
Figuur 12-Gegevens worden grafisch verwerkt om colleges te weergeven



The screenshot shows a web application interface for managing colleges. At the top, there is a purple header bar with a button labeled 'College toevoegen'. Below the header, the main content area has a dark background. On the left, there is a sidebar with a search bar and a list of colleges. The main area displays a table of colleges and their active assessors.

College Naam	Actieve Assessoren
Priest	8
Bouw & design college	3
Business college	16
Dienstverlening	6
Zorg en welzijn college	12
Onderwijs & kinderopvang College	8
ict college	10
Onderwijs Centraal	15
Techniek & technologie college	12

Figuur 15-Weergaven College pagina



The screenshot shows a web application interface for adding a new college. The form is titled 'College Toevoegen' and is located in the top right corner of the page. The form has a dark background and contains several input fields for adding a new college.

College Toevoegen

VUL DE Velden HIERONDER IN

College Naam

Teamleider

Geen

Opslaan

Figuur 16-Weergaven toevoegen van College pagina

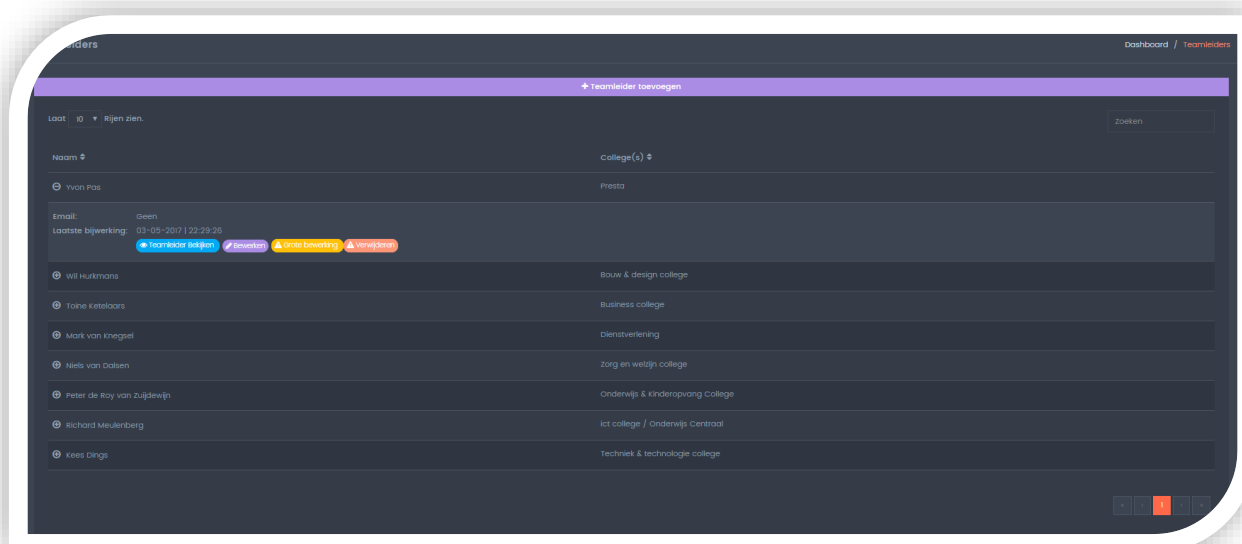
Een college kan door de gebruikers worden toegevoegd door op de "colleges" pagina op de knop "toevoegen" te klikken. De gebruiker wordt hierna verwezen naar een pagina waar een college kan worden toegevoegd. Deze pagina gebruikt exact hetzelfde proces omschreven in Figuur 7 echter word in dit geval de "College Controller" gebruikt om dit event te handelen.

De invulvelden van het formulier worden opgestuurd als een post naar de controller om dit verder te verwerken hier worden de ingevulde gegevens gevalideerd, met als doel ervoor zorgen dat de data correct is ingevuld.

Als de gegevens correct zijn ingevuld word dit nieuw aangemaakte college als object opgeslagen in de database.

Teamleiders pagina

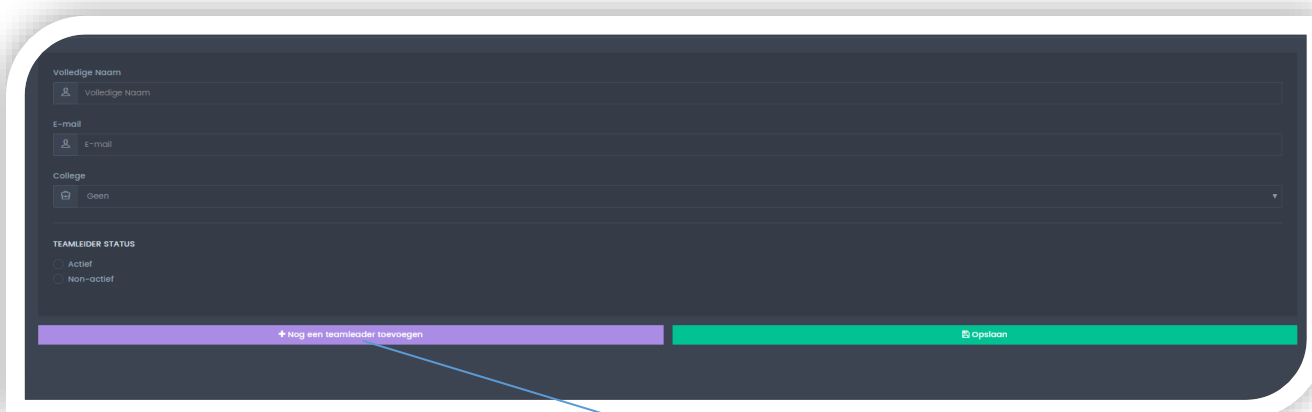
Nu de colleges pagina afgemaakt is, word het steeds makkelijker om andere pagina's te ontwikkelen zo kan de lay-out van de colleges pagina gebruikt worden als ontwerp basis voor de teamleider pagina. Maar op de achtergrond worden de zelfde processen gebruikt te zien in Figuur 14, Figuur 13 en Figuur 12. Alleen worden nu Teamleiders als object gebruikt.



Figuur 17-Weergaven teamleiders pagina

Binnen de applicatie word zo min mogelijk gebruik gemaakt van nieuwe functies. Hiermee word bedoeld dat wanneer de ontwikkelaar een functie moet programmeren, dat zodanig word gedaan, dat deze functie dynamisch zal kunnen werken voor andere pagina's. Zo word de functie voor het toevoegen van colleges ook gebruikt bij het toevoegen van teamleiders.

Dit kan gedaan worden, omdat het allebei objecten zijn. Deze objecten hebben allemaal een toegewezen tabel binnen de database. De functie om deze objecten toe te voegen aan de database is zo ontwikkelt dat het automatisch voor alle object zal werken.



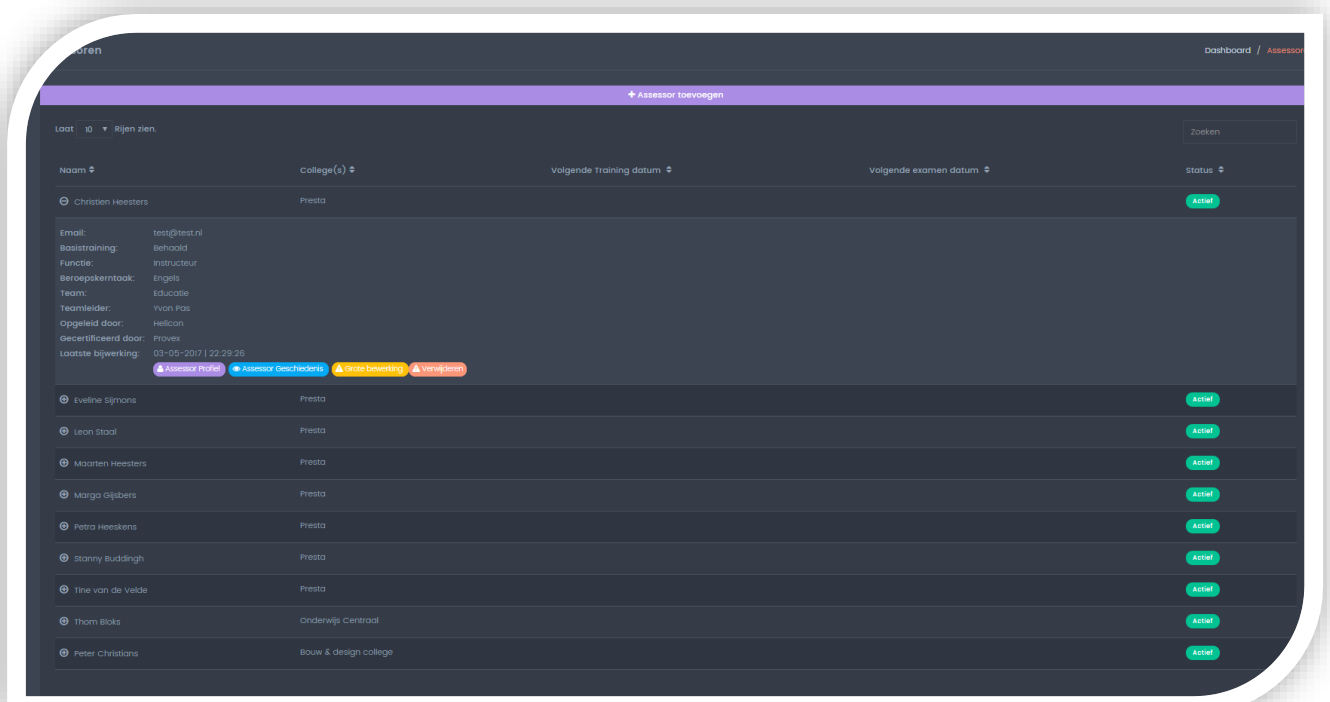
Figuur 18-Weergaven toevoegen teamleider(s)

Bij het toevoegen van een nieuwe teamleider word de optie gegeven om meerdere teamleiders in een keer toe te voegen.

Assessoren Pagina

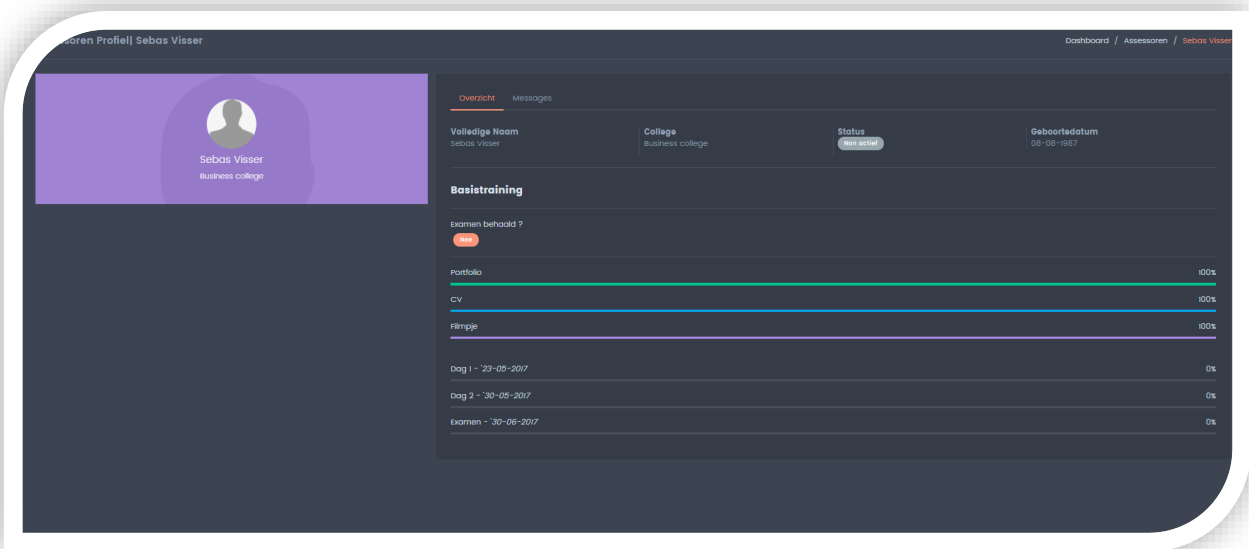
Nu de College en Teamleider pagina's zijn ontwerpt en gerealiseerd kan er een start worden gemaakt aan het hoofdonderwerp van deze applicatie, de assessoren. De assessoren pagina zal dezelfde lay-out krijgen als de College en Teamleider pagina's. Het verschil zal zijn dat het aantal gegevens een stuk groter zal zijn.

Naam	College(s)	Volgende Training datum	Volgende Examen datum	Status
<ul style="list-style-type: none"> Email Basistraining Functie Beroepskerntaak Team Teamleider Opgeleid door Gecertificeerd door Laatste bijwerking 	<ul style="list-style-type: none"> Naam 	<ul style="list-style-type: none"> Datum 	<ul style="list-style-type: none"> Datum 	<ul style="list-style-type: none"> Label



Figuur 19-Weergaven assessoren pagina

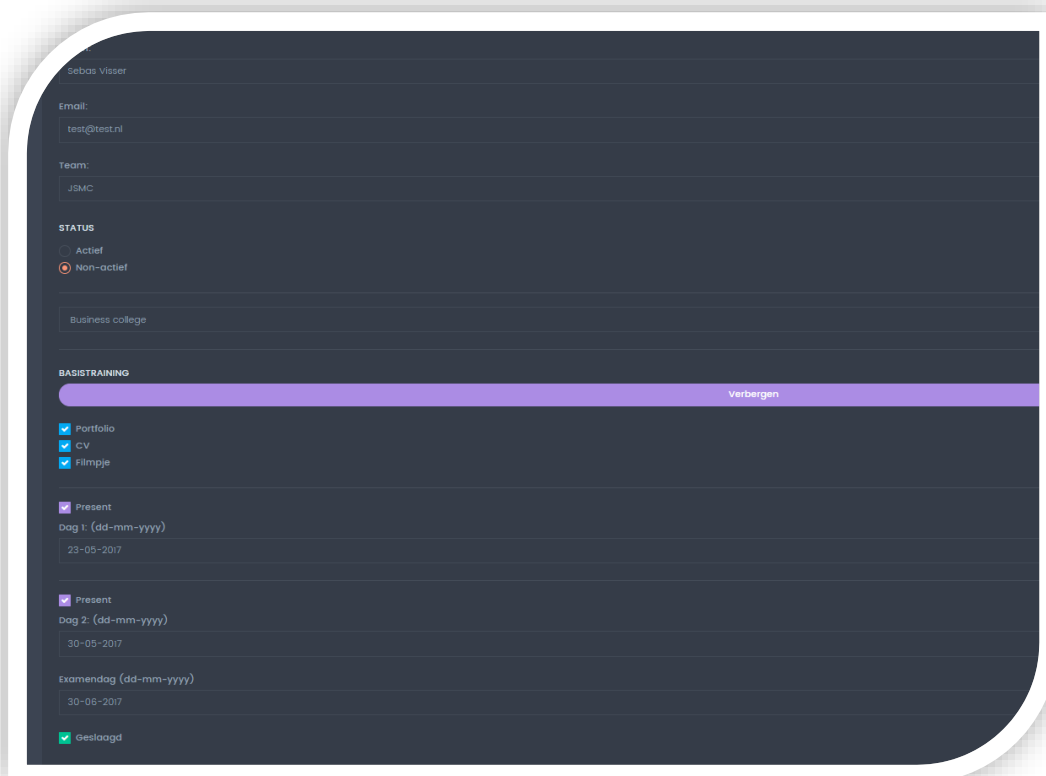
Het doel van deze applicatie is het monitoren van assessoren. Deze applicatie kan weergeven welke assessoren er daadwerkelijk een assessor is, daarbij kan de gebruiker zien welke assessor nog zijn basistraining moet behalen, hierbij kan een profiel worden getoond waar inhoudelijk gezien kan worden aan welke eisen de assessor nog moet voldoen (zie Figuur 20).



Figuur 20-Weergaven profiel assessor pagina

In het voorbeeld hierboven kan gezien worden dat deze assessor zijn basistraining nog moet behalen. Dit kan gezien worden doordat de assessor nog 2 dagen en een examen moet behalen om in het onderhoudstraject te komen (zie Figuur 1).

De gebruikers van de applicatie kunnen dit profiel aanpassen wanneer de assessor deze eisen heeft behaald. Hiervoor zal de gebruiker een "Grote bewerking" moeten uitvoeren. Deze soort bewerking kan teruggevonden worden op andere pagina's zoals College pagina, Teamleider pagina, en de Assessor pagina. Wanneer er een grote wijziging plaatsvindt, zal de toegewezen controller de aangepaste gegevens verwerken en de wijziging opslaan in de log van het object.



Figuur 21-Weergaven Grote bewerking pagina -assessoren


```
(Input::has('graduated')) {
    $basictraining->graduated = true;
    $basictraining->passed = true;
    if ($basictraining->passed != $oldLog['passed'] && $basictraining->graduated != $oldLog['graduated']) {
        $messages[] = "Assessor heeft basistraining behaald !";
    }

    $validator = Validator::make($request->all(), [
        'graduationday' => 'required'
    ]);
    if (!$validator->fails()) {
        $validator = Validator::make($request->all(), [
            'graduationday' => 'required|date_format:"d-m-Y"'
        ], array(
            'graduationday.date_format' => 'Basistraining examen was verkeerd ingevuld,<br> Graag houden aan het dd-mm-yyyy format'
        ));
        if ($validator->fails()) {
            return redirect()->back()->withErrors($validator->getMessageBag()->first());
        }

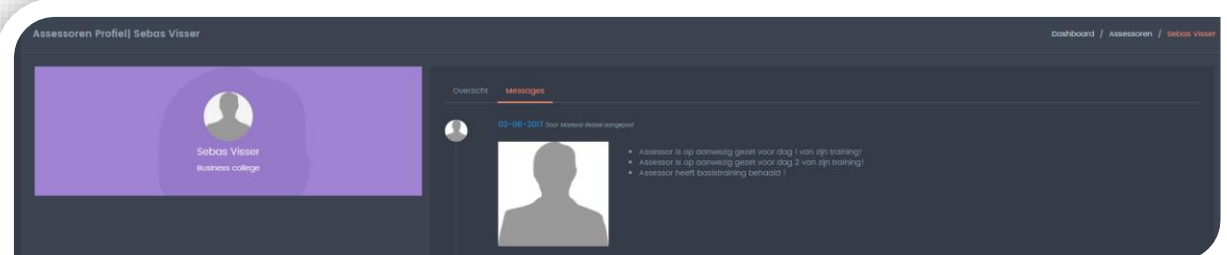
        $basictraining->graduationday = $request->graduationday;
        if ($request->graduationday != $oldLog['graduationday']) {
            $messages[] = "Assessor heeft basistraining <strong>examen</strong> behaald op ". $basictraining->graduationday ."";
        }
    } else {
        $basictraining->graduated = false;
        $basictraining->passed = false;
        if ($basictraining->passed != $oldLog['passed'] && $basictraining->graduated != $oldLog['graduated']) {
            $messages[] = "Basistraining van deze assessor is op niet behaald gezet !";
        }
    }

    Exams::saveBasictraining($assessor->fk_exams, $basictraining);

    $assessor->save();
    if (empty($messages)) {
        return redirect()->back()->withWarning('Geen wijzigingen verkregen...');
    }
    Log::AssessorLog($id, message: $messages, array: true);
    return redirect()->route('route.view_assessor_profile', parameters: $id)->withSuccess('Assessor was succesvol opgeslagen !');
}
```

Figuur 22-Eind proces van "Grote bewerking"

Hier worden de aangepaste waarden opgeslagen in de Log van het object. Daarna zal de gebruiker automatisch worden verwezen naar de profiel pagina van de gewijzigde assessor, hierbij word ook een bericht meegegeven.



Figuur 23-Weergave resultaat van "Grote bewerking"

Onderhoud Pagina

De onderhoud pagina zal ervoor zorgen dat de applicatie meer inhoud zal krijgen. Het mbo examenbureau moet via Excel lijsten 16 plaatsen naar teamleiders sturen die vervolgens deze groep gaan vullen. Hierna zullen deze Excel lijsten van alle teamleiders moeten worden uitgelezen en verwerkt in een onderhoud programma voor de assessoren.

Nu word er van de applicatie verwacht het hierboven genoemde proces te automatiseren tot hoeverre mogelijk zal zijn. Hiervoor moet worden vastgesteld hoe het onderhoudstraject loopt.



Figuur 26-Onderhoudstraject

```

This function will check every assessor for maintenance,
* if there are assessors that need maintenance, it will return an array with:
* * Assessor (Collection)
* * Type maintenance (string: Examag, Onderhoud)
* * Data (Collection: Exam Collection from the assessor)
* @return array|null
*/
public static function MaintenanceUpdate()
{
    # We initialize our variables
    $need_maintenance = null;
    $assessors = Assessors::all();

    # SECTOR 1
    foreach ($assessors as $assessor) {
        # We check every assessor for required maintenance
        $exam = self::find($assessor->fk_exams);

        # SECTOR 1.1
        if (empty($exam)) {
            # this is highly unlikely, but we will Log this error to the system log
            SystemLog::LOG( title: __FUNCTION__, subject: 'SECTOR 1.1', message: "Exam was empty, Assessor: " . $assessor->id, Auth::user()->id);
            continue;
        }

        # SECTOR 1.2
        # We decode the basistraining json data.
        $basistraining = json_decode($exam->basistraining);

        # SECTOR 1.3
        # If this assessor did not graduate his basistraining he does not require basistraining.
        if (!$basistraining->passed) {
            continue;
        }

        # SECTOR 1.4
        # if the assessor passed his basistraining and does not have a second training date,
        # We cannot calculate if this assessor needs a maintenance.
        if ($basistraining->graduationday == null) {
            continue;
        }
    }
}
  
```

Wanneer een gebruiker op de onderhoud pagina komt, zal deze functie worden uitgevoerd om te calculeren welke assessoren er op onderhoud moeten.

Figuur 25-Functie om alle assessoren die op onderhoud moeten te verkrijgen

Naam	Assessor	Type	Afkeuren
Christen Heesters		Onderhoud	<input checked="" type="checkbox"/>
Jeroen Dijk		Onderhoud	<input checked="" type="checkbox"/>
Beckman Heesters		Onderhoud	<input checked="" type="checkbox"/>
Thijs de Boer		Onderhoud	<input checked="" type="checkbox"/>
Peter Oosterhuis		Onderhoud	<input checked="" type="checkbox"/>
Victor van den Broek		Onderhoud	<input checked="" type="checkbox"/>
Marcel Oosterhuis		Onderhoud	<input checked="" type="checkbox"/>
Linda de Boer		Onderhoud	<input checked="" type="checkbox"/>

Ook zal worden berekend welke assessoren op onderhoud moeten en welke op examen moeten.

De gebruiker heeft de optie om de assessor af te vinken na de onderhoud.

Figuur 24-Weergaven van het onderhoud traject van assessoren

Nu er een overzicht van de assessoren die op onderhoud moeten weergeven kan worden. Moet er een functionaliteit komen om d.m.v. deze gegevens deze assessoren in te plannen in groepen. Door op de knop "Onderhoud Groepen maken / inplannen" word de gebruiker verwezen naar een pagina waar groepen kunnen worden ingedeeld.

The screenshot shows a web application interface for creating maintenance groups. The page has a dark theme with a blue header bar. The main content area is divided into sections for group configuration and a list of assessors.

Callout 1: Aangemaakte groepen kunnen worden aangepast naar de wensen van de gebruiker. (Points to the 'Groep I' title and the 'Instelling' section.)

Callout 2: Alleen assessoren die onderhoud vereisen kunnen worden ingedeeld in groepen. (Points to the 'Reserve' dropdown menu in the assessor list.)

Callout 3: Validatie word live op de pagina gedaan hierdoor kunnen groepen snel achter elkaar aangemaakt worden. (Points to the 'Groep I' title and the 'Instelling' section.)

Form Fields:

- GROEP ANSTD** (Header)
- Groep Titel** (Label)
- Groep I** (Text input)
- Instelling** (Section header)
- Locatie** (Section header)
- Van: (dd-mm-yyyy)** (Text input)
- 02-08-2017** (Text input)
- Tot: (dd-mm-yyyy)** (Text input)
- 02-08-2017** (Text input)
- #** (Text input)
- Naam** (Text input)
- College** (Text input)
- Teamleider** (Text input)
- Reserve** (Dropdown menu)

#	Naam	College	Teamleider
1	Reserve		
2	Reserve		
3	Reserve		
4	Reserve		
5	Reserve		
6	Reserve		
7	Reserve		
8	Reserve		
9	Reserve		

Figuur 27-Weergaven Onderhoud Groepen maken / inplannen pagina