

**Planningstool**

**Zwart op Wit**

Jo Brunau

Kathleen Jansen

HBO5 Informatica- Optie Programmeren

Academiejaar 2016-2017

# Dankwoord

Na een intensieve periode van zwoegen en zweten is het zo ver. Met het schrijven van dit dankwoord leggen we de laatste hand aan ons eindwerk project, een planningstool voor Zwart op Wit.

Het is een periode waar we beiden veel hebben bijgeleerd. Niet enkel op gebied van programmeren, maar ook persoonlijke ontwikkeling en het samenwerken als team om zo tot een mooi product te komen.

Op de eerste plaats willen we elkaar graag bedanken. Door het uitwerken van dit project werden we een hecht team. We leerden veel bij en ondersteunden elkaar waar nodig. Communicatie is hierin zeer belangrijk.

Ook graag een dankjewel aan onze projectbegeleidster Maaike Van den Bulck. We konden steeds op haar rekenen voor een mooie opvolging en begeleiding van ons project.

Als laatste, maar zeker niet onbelangrijk willen we graag onze partners bedanken. Door het project waren we afgelopen maanden niet altijd even hard aanwezig maar dat werd door jullie met veel begrip opgevangen.

Voor iedereen een welgemeende dankjewel,

Jo Brunau & Kathleen Jansen

# Samenvatting

Drukkerij Bulckens is een drukkerij waar via de website zwartopwit.com online drukorders kunnen worden besteld. De gegevens van de job worden geïmporteerd en verwerkt in Filemaker Pro voor Mac. Om de jobs in te plannen op de afwerking machines van de drukkerij en een volwaardig planningstool mee uit te bouwen is deze software jammer genoeg niet geschikt. Er is nood aan een programma die deze jobs gemakkelijk importeert en verdeelt over de planning van de verschillende machines.

Om dit te bereiken zijn we gestart van een csv-file, gegenereerd in Filemaker die geïmporteerd wordt in ons programma. De gegevens van de jobs worden in de database geplaatst en de werktijden berekend. De jobs krijgen de status ‘todo’ en kunnen volgens leverdatum opgeroepen worden. De werknemer dient in te loggen en kan zo het programma gebruiken. In de todo-lijst is het mogelijk de jobs in te plannen op de machines. De jobs kunnen gestart worden waardoor een tijdsregistratie in de database komt, per job en per werknemer. Als de jobs klaar zijn verdwijnen ze uit de lijst. Werknemers, machines, afdelingen etc... kunnen worden toegevoegd en gewijzigd. Ook de parameters die gebruikt worden om de planningstijden te berekenen.

De structuur van het programma is MVC gebouwd (Model-View-Controller). Dit wil zeggen object-georiënteerd waardoor alle parameters van de verschillende objecten gemakkelijk gebruikt kunnen worden, aanpasbaar zijn en waardoor het programma gemakkelijk uit te breiden is. De gebruik van deze structuur is niet voldoende mogelijk in het oorspronkelijke programma Filemaker. Ons programma is dus de perfecte aanvulling en kan stand-alone draaien op een eigen database.

Inhoud

[4 Planningstool – Zwart Op Wit 1](#_Toc483169303)

[4.1 Wie zijn Zwart op Wit 1](#_Toc483169304)

[4.2 Een planningstool? 1](#_Toc483169305)

[4.3 Requirements 1](#_Toc483169306)

[4.3.1 Business & Functional Requirements 1](#_Toc483169307)

[4.3.2 Design Requirements 2](#_Toc483169308)

[4.4 Technologie 2](#_Toc483169309)

[5 Analyse 3](#_Toc483169310)

[5.1 Mindmap 3](#_Toc483169311)

[5.2 User Stories & BPMN 4](#_Toc483169312)

[5.2.1 Inloggen 4](#_Toc483169313)

[5.2.2 Gebruikers beheren 5](#_Toc483169314)

[5.2.3 Afdelingen beheren 6](#_Toc483169315)

[5.2.4 Machines beheren 7](#_Toc483169316)

[5.2.5 File Importeren 8](#_Toc483169317)

[5.2.6 Job inplannen 9](#_Toc483169318)

[5.2.7 Job Starten/Stoppen 10](#_Toc483169319)

[5.3 UML klassendiagramma 11](#_Toc483169320)

[5.4 Gegevensmodel ERD 12](#_Toc483169321)

[6 Proof of concept 13](#_Toc483169322)

[7 Technologische concepten 14](#_Toc483169323)

[7.1 MVC 14](#_Toc483169324)

[7.1.1 Introductie 14](#_Toc483169325)

[7.1.2 Model 14](#_Toc483169326)

[7.1.3 View 14](#_Toc483169327)

[7.1.4 Controller 14](#_Toc483169328)

[7.1.5 Voordelen 14](#_Toc483169329)

[7.1.6 Nadelen 15](#_Toc483169330)

[7.1.7 MVC in .net Core 15](#_Toc483169331)

[7.2 Git 18](#_Toc483169332)

[7.2.1 Introductie 18](#_Toc483169333)

[7.2.2 Basishandelingen 18](#_Toc483169334)

[7.2.3 Branching 19](#_Toc483169335)

[7.3 Bootstrap 19](#_Toc483169336)

[7.4 Rendering engine (razor) 19](#_Toc483169337)

[7.5 Identity 19](#_Toc483169338)

[7.5.1 Introductie 19](#_Toc483169339)

[7.5.2 Identity activeren 20](#_Toc483169340)

[7.5.3 Registreren + Inloggen 21](#_Toc483169341)

[7.5.4 Uitloggen 22](#_Toc483169342)

[7.5.5 Uitbreiding userklasse 22](#_Toc483169343)

[7.5.6 Rollen 23](#_Toc483169344)

[7.5.7 Gebruikers beheren 26](#_Toc483169345)

[7.5.8 Forgot password 27](#_Toc483169346)

[7.6 Globalisation and localization 28](#_Toc483169347)

[7.6.1 Introduction 28](#_Toc483169348)

[7.6.2 Gebruik in ons project 28](#_Toc483169349)

[7.7 Components 32](#_Toc483169350)

[7.7.1 Introduction 32](#_Toc483169351)

[7.7.2 Components vs. Partial View 32](#_Toc483169352)

[7.8 Entity Framework 34](#_Toc483169353)

[7.8.1 Inleiding 34](#_Toc483169354)

[7.8.2 Wat doet Entity Framework? 34](#_Toc483169355)

[7.8.3 Voordelen 34](#_Toc483169356)

[7.9 CRUD 35](#_Toc483169357)

[7.9.1 Index 35](#_Toc483169358)

[7.9.2 Create 38](#_Toc483169359)

[7.9.3 Read 39](#_Toc483169360)

[7.9.4 Update 40](#_Toc483169361)

[7.9.5 Delete 41](#_Toc483169362)

[7.10 Patterns 41](#_Toc483169363)

[7.10.1 Inleiding 41](#_Toc483169364)

[7.10.2 Options pattern 41](#_Toc483169365)

[7.10.3 Simple factory pattern (Calculation method) 42](#_Toc483169366)

[7.11 CSV Import 43](#_Toc483169367)

[8 Werking programma 45](#_Toc483169368)

[8.1 Users beheren 45](#_Toc483169369)

[8.1.1 Introductie 45](#_Toc483169370)

[8.1.2 Index pagina 45](#_Toc483169371)

[8.1.3 User CRUD 47](#_Toc483169372)

[8.1.4 Zoeken 50](#_Toc483169373)

[8.1.5 Navigeren 50](#_Toc483169374)

[8.2 Departementen beheren 51](#_Toc483169375)

[8.2.1 Introductie 51](#_Toc483169376)

[8.2.2 Index pagina 51](#_Toc483169377)

[8.2.3 Department CRUD 52](#_Toc483169378)

[8.2.4 Zoeken 55](#_Toc483169379)

[8.2.5 Navigeren 55](#_Toc483169380)

[8.3 Machines beheren 55](#_Toc483169381)

[8.3.1 Introductie 55](#_Toc483169382)

[8.3.2 Index pagina 56](#_Toc483169383)

[8.3.3 Machine CRUD 57](#_Toc483169384)

[8.3.4 Zoeken 63](#_Toc483169385)

[8.3.5 Navigeren 63](#_Toc483169386)

[8.4 Time registers beheren 64](#_Toc483169387)

[8.4.1 Introductie 64](#_Toc483169388)

[8.4.2 Index pagina 64](#_Toc483169389)

[8.4.3 Time register CRUD 65](#_Toc483169390)

[8.4.4 Zoeken 68](#_Toc483169391)

[8.4.5 Navigeren 69](#_Toc483169392)

[8.5 Job planning 69](#_Toc483169393)

[8.5.1 Jobs menu items 69](#_Toc483169394)

[8.5.2 Jobs starten en stoppen 73](#_Toc483169395)

[8.5.3 Job flow 73](#_Toc483169396)

[9 Algemeen Besluit 75](#_Toc483169397)

[10 Bijlages 76](#_Toc483169398)

[Plan van Aanpak 77](file:///C:\Users\Koen\Downloads\EindwerkPaperFINALFinal.docx#_Toc483169399)

[11 Bronnen 81](#_Toc483169400)

# Inleiding

Deze paper omschrijft ons project waarin we op zoek gegaan zijn naar een juiste oplossing voor een werkende planningstool voor drukkerij Bulckens / zwartopwit.com.

We zijn gestart met een mindmap waarin de ruggengraat van alle functionaliteiten aan bod komen. Van daaruit hebben we Userstories aangemaakt. Deze hebben we gebruikt om een idee te vormen van de klasses die we nodig hebben. BPMN schema’s zijn ook uitgewerkt. Door middel van een ‘proof of concept’ zijn we erin geslaagd een framework te bouwen die ons de basis gegeven heeft voor de verdere uitwerking van het programma. De tijd die we gestoken hebben in het framework is van die aard geweest dat we de basis van het programma hebben kunnen bouwen, met name het gedeelte voor de nietmachine, van waaruit we in de toekomst makkelijk de verdere volledige uitbouw van het definitieve planningstool kan uitgewerkt worden.

Na het opstarten van een Git-Repository op GitHub zijn we gestart met de opbouw van het programma. In de opbouw hebben we rekening gehouden met de MVC-structuur om zo het programma overzichtelijk te houden en gemakkelijk uit te breiden.

Ten slot hebben we autorisatie en authenticatie voorzien. Deze eigenschap is cruciaal voor de verdere verwerking van de rest van het programma: de effectieve planningstool zelf. We zijn gestart met de opbouw van het tool voor de afdeling Stitch. Verdere uitbreiding naar de ander afdelingen is voor de nabije toekomst.

# Planningstool – Zwart Op Wit

## Wie zijn Zwart op Wit

Zwart op Wit is een bedrijf voor het online bestellen van drukwerk. Sinds 2000 is ‘www.zwartopwit.be’ de online afdeling van Drukkerij Bulckens. Een bedrijf met meer dan 80 jaar ervaring.

PrePress, drukken, printen, afwerken en inpakken. De volledige productie is in eigen beheer. Met hun uitgebreide collectie afwerkingsmogelijkheden vindt elke klant wat hij of zij nodig heeft.

## Een planningstool?

Zwart op wit krijgt bestellingen online binnen. Nadat deze bestellingen ontvangen en verwerkt zijn, is de vertaling van het bestelde pakket drukwerk naar de werkvloer toe complex georganiseerd.

Gegevens die binnenkomen via de website worden geïmporteerd in Filemaker voor Mac. De administratieve opvolging gebeurt via Filemaker. De technische opvolging hiervan gebeurt via MultiPress. Deze software is speciaal ontwikkeld voor de verwerking van grafische gegevens.

De ingave in Multipress vereist voornamelijk een manuele ingave en werkt traag. Een ander nadeel is dat enkel bedienden met een grafische achtergrond deze toepassing kunnen gebruiken.

Onze planningstool maakt dat we Multipress gedeeltelijk kunnen uitsluiten. Ook kunnen de medewerkers voor het afwerken van het drukwerk op de werkvloer via een pc checken welke werken er op een bepaalde dag uitgevoerd moeten worden. De technische gegevens kunnen geraadpleegd worden en ook de tijd die nodig is om een drukwerk klaar te krijgen.

Als het werk afgewerkt is, kunnen we via een nacalculatie controleren wie welk werk heeft uitgevoerd en of dit binnen de vooropgestelde tijd is gebeurd. De verworven data kan gebruikt worden voor controle en bijsturing, maar het biedt ook een mogelijkheid om oplossingen te geven voor toekomstige orders.

## Requirements

Bij het maken van een programma komen heel wat vereisten. We hebben een onderscheid gemaakt tussen enerzijds business & functional requirements, anderzijds design requirements.

### Business & Functional Requirements

* Systeem moet op basis van externe Filemaker database gegevens importeren in eigen database
* Gegevens in database moeten bijgewerkt kunnen worden
* Systeem moet a.d.h.v. in de database de geschatte werktijden kunnen berekenen
* De applicatie moet beschikken over een beveiligd identificatieproces
* Er moet een onderscheid gemaakt worden tussen gebruiker en verantwoordelijke
* Nieuwe gebruikers moeten kunnen worden aangemaakt
* Nieuwe afdelingen en machines moeten kunnen worden aangemaakt in geval van uitbreiding van het bedrijf
* Gebruikers moeten een planplaats kunnen kiezen
* Systeem moet drukwerken aan de juiste planplaats toewijzen
* Gebruikers moeten af te werken drukwerk kunnen kiezen volgens levertijd en bestemming.
* Systeem moet werktijden registeren
* Systeem moet onderbrekingen registeren
* Gebruikers moeten opmerkingen kunnen toevoegen aan elke tijdsregistratie
* Systeem moet alle verzamelde gegevens naar de database schrijven en linken aan de gekozen drukwerken
* Systeem moet in realtime kunnen tonen welke gebruikers, welke drukwerken hebben opgestart
* Systeem moet queries kunnen uitvoeren op de database om nacalculaties uit te voeren volgens werknemer en drukwerk

### Design Requirements

* De applicatie zal in eerste instantie standalone draaien. Later kan nog bekeken worden of verweven in de Filemaker database tot de mogelijkheid behoort.
* De applicatie dienst geschreven te worden in .NET
* Het resultaat is een web-applicatie
* De database wordt een mySQL

## Technologie

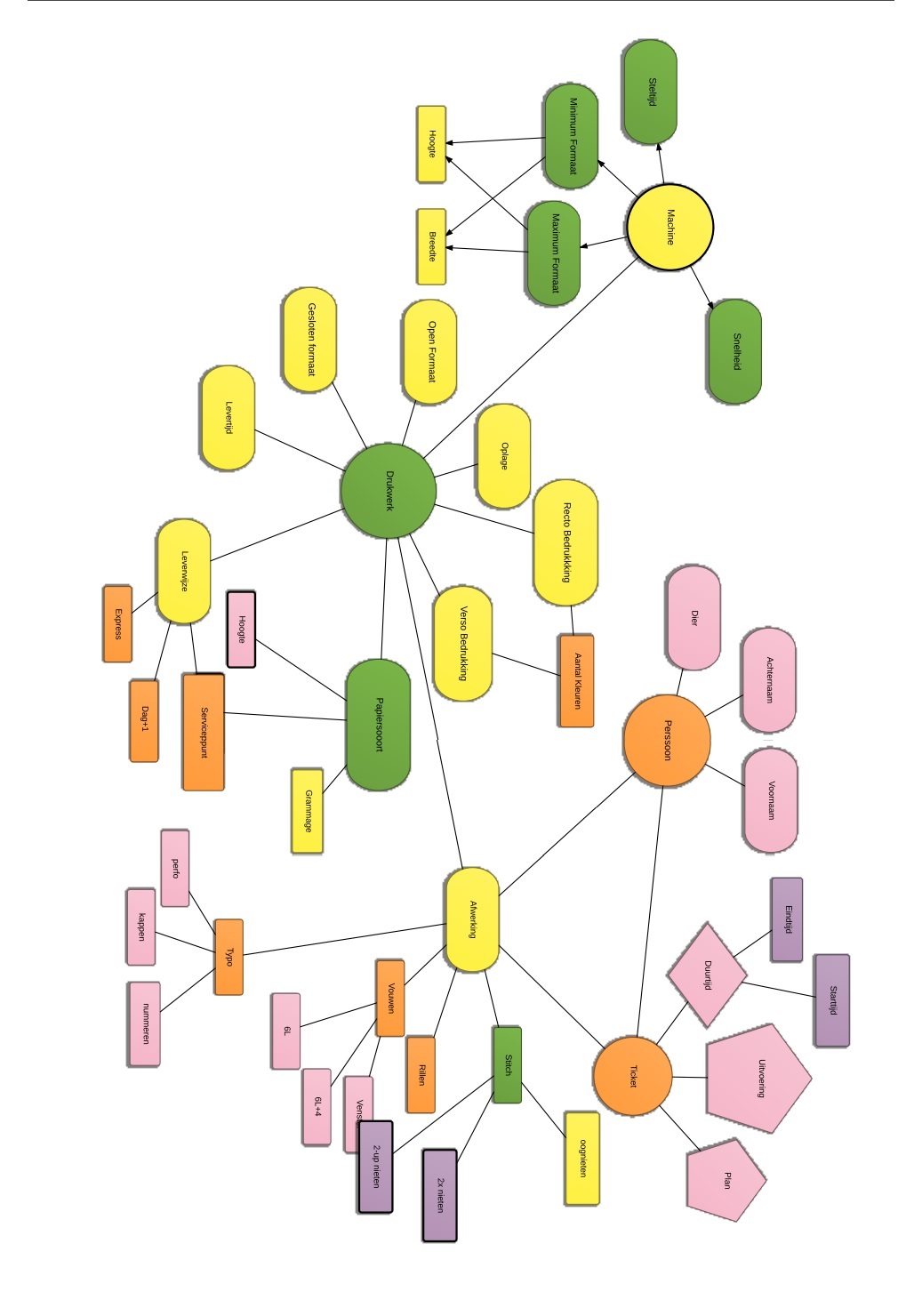
Doordat er in het huidige IT-landschap al gebruik wordt gemaakt van Linux based services (Ruby) en er een MySQL database beschikbaar is de keuze gevallen om technologieën in deze lijn te gebruiken. De data die door de planning tool gebruikt wordt kan worden teruggevonden in Filemaker.

* Asp.net Core c#
* MySQL
* Filemaker
* Razor

# Analyse

## Mindmap

We zijn onze analyse gestart met het maken van een mindmap. Dit gaf ons een bredere kijk op ons project. Waar denken we aan, wat moeten we allemaal voorzien,… Door het maken van een mindmap werden niet enkel de hoofdbestanddelen, maar ook de verbanden er tussen duidelijker.



## User Stories & BPMN

### Inloggen

#### User story - Inloggen

|  |  |
| --- | --- |
| Hoofddoel | * Als gebruiker kan ik inloggen met een e-mailadres/username en paswoord. |
| Bijkomende vereisten | * De username en paswoord worden gevalideerd door het systeem en geeft een melding indien de validatie mislukt is. * Het systeem kijkt welke rol de gebruiker heeft die is ingelogd. Deze kan een verantwoordelijke zijn of een gewone gebruiker. * Bij een correcte validatie wordt de gebruiker geauthentiseerd en de hoofdpagina geopend. |

#### BPMN - Inloggen

### Gebruikers beheren

#### User story – Gebruikers beheren

|  |  |
| --- | --- |
| Hoofddoel | * Als verantwoordelijke kan ik nieuwe gebruikers aanmaken en bestaande gebruikers bewerken en verwijderen. |
| Bijkomende vereisten | * Er zijn twee gebruikers niveaus, “verantwoordelijke” die gebruikers kan beheren en instellingen doen in het systeem. Daarnaast is er een gewone gebruiker die gebruik kan maken van de planningstool. * Om het per ongeluk verwijderen van een gebruiker tegen te gaan wordt er eerst een bevestiging gevraagd. |

#### BPMN – Gebruikers beheren

### Afdelingen beheren

#### User story – Afdelingen beheren

|  |  |
| --- | --- |
| Hoofddoel | * Als verantwoordelijke kan ik nieuwe afdelingen aanmaken en bestaande afdelingen bewerken en verwijderen. |
| Bijkomende vereisten | * Om het per ongeluk verwijderen van een departement tegen te gaan wordt er eerst een bevestiging gevraagd. * Indien er nog een machine aan de afdeling gekoppeld is, kan deze niet verwijderd worden |

#### BPMN – Afdelingen beheren

### Machines beheren

#### User story – Machines beheren

|  |  |
| --- | --- |
| Hoofddoel | * Als verantwoordelijke kan ik nieuwe machines aanmaken en bestaande machines bewerken en verwijderen. |
| Bijkomende vereisten | * Om het per ongeluk verwijderen van een machine tegen te gaan wordt er eerst een bevestiging gevraagd. * Bij het aanmaken van een nieuwe machine moet deze aan een bestaande afdeling gekoppeld worden |

#### BPMN – Machines beheren

### File Importeren

#### User Story – File importeren

|  |  |
| --- | --- |
| Hoofddoel | * Als verantwoordelijke kan ik een bestand met werkgegevens inlezen. |
| Bijkomende vereisten | * Het bestand is een csv-file afkomstig uit FileMaker Pro. * De gegevens worden naar de database gestuurd naar meerdere tabellen |

#### BPMN – File importeren

### Job inplannen

#### User story – Job Inplannen

|  |  |
| --- | --- |
| Hoofddoel | * Als verantwoordelijke ik een job inplannen. |
| Bijkomende vereisten | * Juiste leverdatum moet geselecteerd zijn |

#### BPMN – Job inplannen

### Job Starten/Stoppen

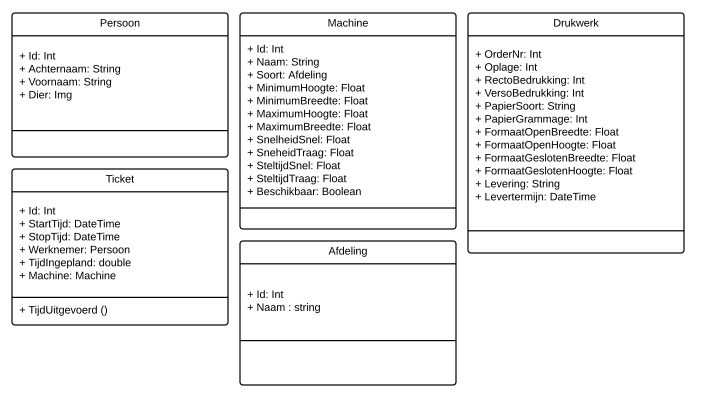
#### User story – Job Starten/Stoppen

|  |  |
| --- | --- |
| Hoofddoel | * Als gebruiker kan ik de tijdsregistratie van een drukwerk starten |
| Bijkomende vereisten | * Job bestaat en is nog niet afgewerkt. * Lijst met afgewerkte jobs wordt getoond op scherm |

#### BPMN – Job Starten/Stoppen

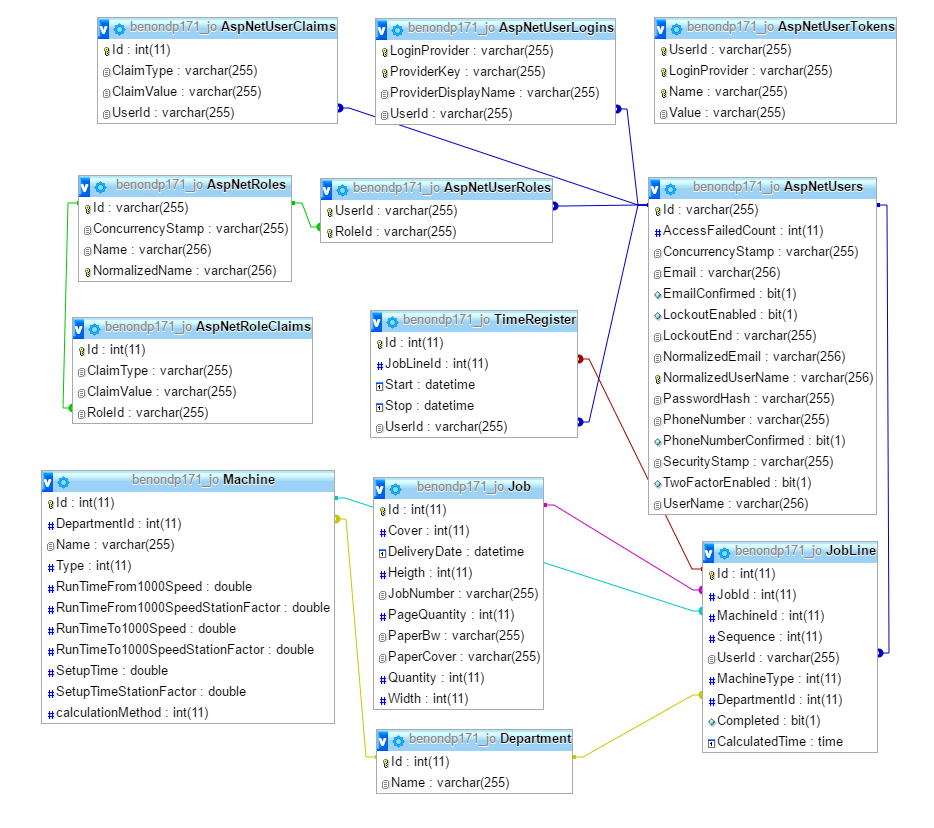
## UML klassendiagramma

Bij het uitwerken van het UML klassendiagramma waren we gestart met het noteren van onze hoofd entiteiten. Jammer genoeg zijn we door alle drukte door vergeten hier mee verder te gaan. Door met het project bezig te zijn, werd dit als bijna automatisch meer uitgewerkt. Als we hier op terugblikken is dit zeker iets waar de in de toekomst anders mee zouden omgaan. Door het goed uitwerken van een klassendiagramma was er meer duidelijkheid geweest, wat ook weer positieve voordelen gehad zou hebben op gebied van tijd.

We geven ons origineel klassendiagramma wel graag mee, ten volste beseffend dat dit niet voldoende is.

Een klassendiagram vanuit het project zelf, op basis van onze code, hebben we jammer genoeg niet kunnen genereren. Dit omdat de volledige functionaliteit voor klasse generatie in .net Core niet goed werkt. We meerdere pogingen ondernomen om hier een oplossing voor te vinden, maar kwamen telkens weer problemen tegen. Aangezien er nog steeds gewerkt wordt aan het .net Framework hopen we dit op een later tijdstip nog te kunnen genereren.

## Gegevensmodel ERD



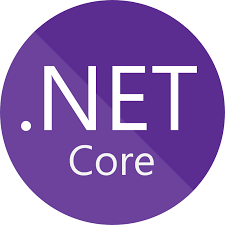
# Proof of concept

Een hele periode van ons project werd gespendeerd aan een “Proof of Concept”. Deze proof of concept was nodig om na te gaan of de geselecteerde technologieën goed samenwerken. Deze technologieën werden geselecteerd op basis van de huidige werking van de klant.

De klant heeft een server draaien met daarop een MySQL database. Momenteel draaien alle servers een linux besturingssysteem, iets wat de klant graag wou behouden. Op basis van deze criteria en de gezien leerstof hebben wij onze volgende technologieën gekozen.

Omdat C# goed bij ons lessenpakket aansloot, hebben we gekozen voor .net Core. Deze versie van .net draait ook onder Linux. Als database kozen we voor een MySQL database.

Het was een hele uitdaging om alle gekozen dependencies goed onderling met elkaar te laten samenwerken. De extra moeilijkheid was dat er nog geen officiële MySQL adapter was toen wij ons project begonnen zijn.

****

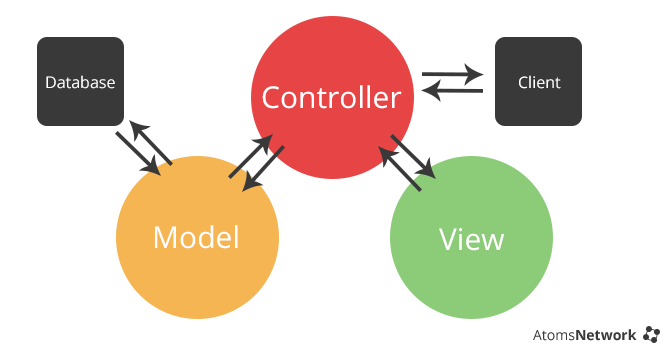


# Technologische concepten

## MVC

### Introductie

MVC staat voor Model-View-Controller. Dit design pattern deelt een applicatie op in drie onderling verbonden delen. Door de opdeling in deze drie componenten zal je een applicatie krijgen die makkelijker te onderhouden is. Uiteindelijk zal dit ervoor zorgen dat de code ook makkelijker opnieuw gebruikt kan worden. Ook zullen meerdere mensen makkelijker samen kunnen werken aan de applicatie, aangezien iedereen duidelijk overzicht heeft van de structuur.



### Model

Het model behandeld alle data, logica en bewerkingen. Ook is het model de component die communiceert met de database.

### View

De View is eigenlijk de presentatie laag. Deze ontvangt de nodige gegevens van de Controller en toont de weergave aan de user.

### Controller

De controller is de schakel tussen het model en de view. Deze geeft de nodige informatie aan de view, handelt interacties af en werkt het model bij. Best practise is dat de View en het Model nooit rechtstreeks met elkaar communiceren.

### Voordelen

* Developers kunnen gemakkelijk gelijktijdig aan de applicatie werken
* Duidelijke structuur
* Herbruikbaar
* Gemakkelijk aanpasbaar
* Een model kan meerdere views hebben

### Nadelen

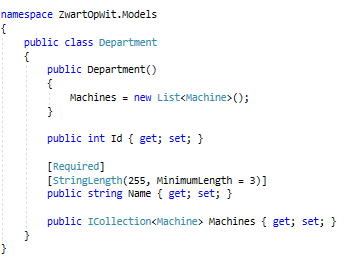
* Voor zeer grote applicaties kan deze structuur moeilijk te onderhouden worden en moeten er extra lagen woren toegevoegd om te vermijden dat de controllers gigantische cluwes van code worden

### MVC in .net Core

Een voorbeeld uit onze code over hoe wij MVC hebben toegepast. In dit specifiek deel behandelen we de departement entiteit (model, controller en view).

#### Model

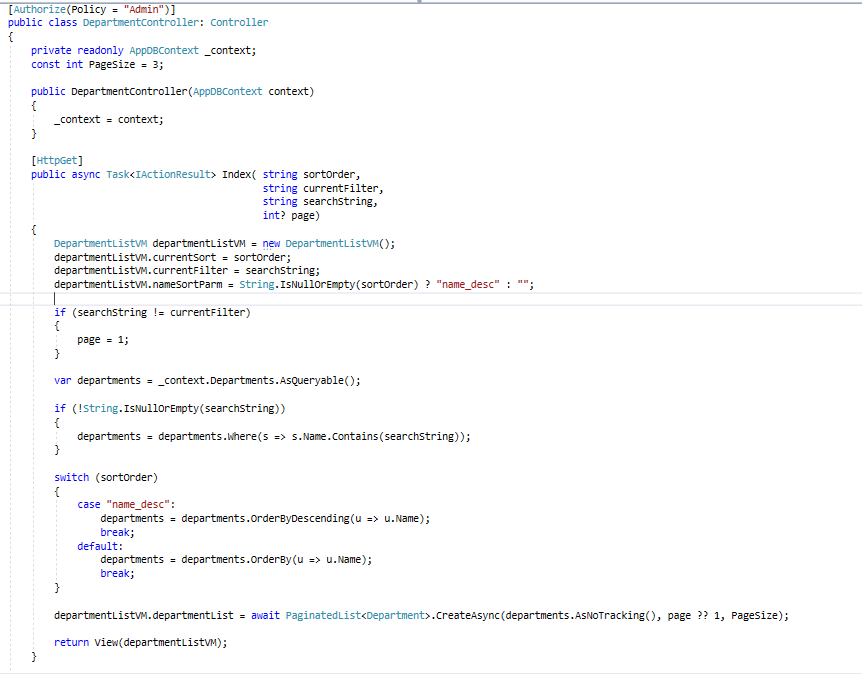
Het model is de representative van de table Department in code. Door code first te werken hebben wij eerst het model aangemaakt wat later naar de database werd gemapt via Entity framework.



MVC – voorbeeld Department model

#### Controller

In de DepartmentController maken we gebruik van het Model “Department” de data die we willen weergeven in de view klaar te zetten. In dit geval de lijst van departementen waarop een filter en een rangschikking kunnen staan. Deze data wordt dan met behulp van het viewModel “DepartmentListVM” doorgegeven aan de view.



MVC – Voorbeeld Department Controller

#### View

In het view wordt de data uit het viewmodel gebruikt om de pagina op te bouwen. De view zal nooit rechtstreeks met het Model (Database) communiceren. Het zal altijd langs een controller gaan. Als best practice wordt alle data die tussen controller en view gestuurd wordt in een view model geplaatst.



MVC – Voorbeeld Department View

## Git

### Introductie

Git is een Source Control systeem voor het beheren van code. Dit maakt het makkelijk om de broncode van een programma op een gedeelde plaats te hebben, zodat wijzigen over alle kopieën automatisch meegedeeld worden.

### Basishandelingen

#### Commit

Git bekijkt welke wijzigingen er zijn gebeurd t.o.v. de lokale repository en geeft deze weer in een lijst. Als je deze commit dan door voert worden deze opgenomen in de lokale repository. Belangrijk is wel dat je aanduidt om “alle files” mee te nemen, aangezien nieuwe files niet standaard mee ingecheckt worden.

#### Push

Een push volgt na een commit. Alle commits die gebeurd zijn sturen we dan ook door naar onze remote repository, zodat deze ook zichtbaar worden voor andere medewerkers.

Als jou lokale versie achter loopt op de versie van de remote, zal je een foutmelding krijgen. Daarom is het belangrijk om eerst altijd te pullen en daarna de pushen.

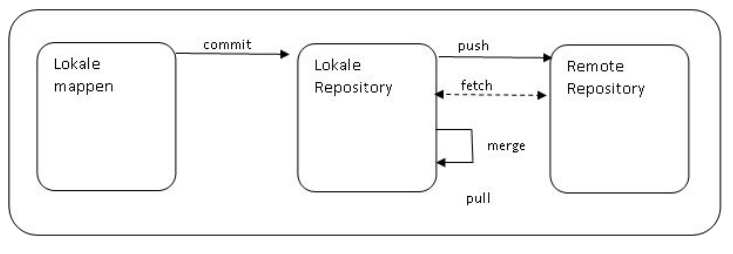
#### Pull

Bij een Pull gaat Git kijken of er wijzingen staan op de remote repository. Indien je hier mee verder gaat, worden de wijzigingen zowel op de bestanden zelf als op de lokale repository toegepast.

#### Fetch

Bij een fetch ga je enkel de wijzigingen ophalen van de remote repository en deze toepassen op de lokale repository. In tegenstelling tot een pull worden de bestanden niet gewijzigd.

#### Merge

Een merge is een manier om manueel wijzigingen door de voeren aan de repository of bestanden. Het maakt het mogelijk om twee branches samen te voegen. 

### Branching

Bij branching maak je een aftakking van de repository aan. Deze aftakking krijgt alle wijzigingen mee die voorheen zijn gebeurd op de hoofdtak. Eenmaal afgetakt is het aan de user zelf om deze up to date te houden. Je tak blijft naast de hoofdtak bestaan. Je kan deze ook op de remote repository zetten, zodat deze zichtbaar is voor anderen. Dit is makkelijk voor bijvoorbeeld code reads.

Desondanks dat het systeem van branching zeer handig is, hebben we besloten om enkel met een “master” te werken. Aangezien ons team uit slechts twee personen bestaat, was dit voldoende.

## Bootstrap

Bootstrap is een open source front-end framework voor het designen van websites en webapplicaties. Het bevat HTML en CSS templates voor zowel tekts, knoppen, navigatie, andere componenten en optioneel zelfs javascript extensies.

Bootstrap maakt het de front-end developper gemakkelijk om snel een mooi resultaat te krijgen.

Wij hebben gekozen om met bootstrap te werken voor het snelle maar toch ordelijke visuele resultaat. Daarnaast maakt bootstrap het gemakkelijk om repsonsive te werken. Momenteel wordt er bij zwart op wit enkel gebruik gemaakt van een desktops en laptops. Wil het bedrijf later overschakelen op tablets of andere, vormt dit geen enkel probleem. De applicatie is er al helemaal klaar voor.

## Rendering engine (razor)

Razor wordt gebruikt voor het creëren van dynamische webpagina’s. Razor is een syntax die gebruikt kan worden om een html pagina in een model te gieten. De razor syntax maakt gebruik van het “@” teken en vereist geen sluitingstag.

De syntax van Razor maakt de html pagina’s korter en zorgt er voor dat de code compacter wordt. Met andere woorden, meer functionaliteit, door minder code.

## Identity

### Introductie

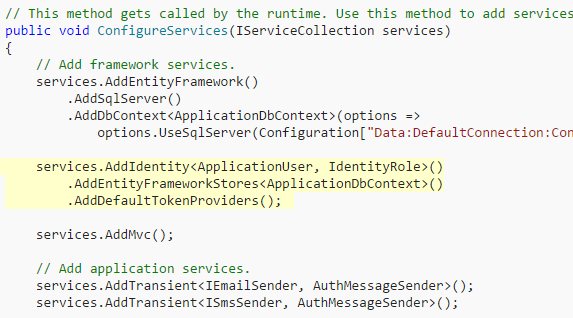
Doordat het project nood heeft aan identificatie van gebruikers en deze verschillende rollen hebben is er nood aan een autorisatie en authenticatie framework. Hier hebben we geopteerd voor een implementatie van het .net Core Identity Framework.

Gebruikers krijgen een account die ze kunnen gebruiken om in te loggen met een username en paswoord. Het identity framework biedt ook de mogelijkheid om te werken met een externe authenticatie provider zoals Facebook, Google, Twitter,… Deze hebben we voor ons project niet toegepast aangezien het om een interne applicatie gaat.

Om gebruik te kunnen maken van “Identity” moet er eerst het Nuget Package “Microsoft.AspNetCore.Identity.EntityFrameworkCore” geïnstalleerd worden.

### Identity activeren

Om Identity te activeren moet deze als een service toegevoegd worden aan de startup.cs klasse (Identity, Screenshot 1). Deze klasse zorgt er voor dat deze service door middel van dependency injection beschikbaar wordt in de applicatie.



Identity, Screenshot 1: Identity Service toevoegen aan startup.cs

Naast het activeren van de service moet ook de optie “UseIdentity” in de “Configure” methode van de startup klasse worden toegevoegd. Dit zorgt tegelijk ook voor cookie-based authenticatie.

In de configure klassen kunnen ook andere configuraties worden toegevoegd zoals het aantal log in pogingen, het pad naar de loginpagina, het log out pad,… (Identity, screenshot 2)



Identity, Screenshot 2: Toevoegen van configuraties

### Registreren + Inloggen

Identity beschikt over een registratiepagina voor gebruikers. Deze gebruiken we niet omdat we willen dat een admin alle gebruikers aan maakt.

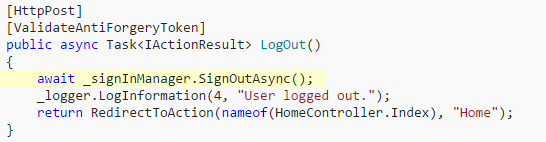
Een login wordt afgehandeld via de loginmethode in de accountController. Deze maakt achterliggend gebruik van de klasse SignInManager. Dit is een identity framework klasse die alle logica afhandelt in verband met inloggen. Aan deze methode wordt een e-mail en paswoord mee gegeven waarna deze bepaald of dit e-mail adres bestaat en dat dit daarvoor het juist paswoord is. Achterliggend wordt hier ook gewerkt met encryptie en salts (Identity, Screenshot 3).



Identity, Screenshot 3: Login

### Uitloggen

Er bestaat natuurlijk ook de mogelijkheid om uit te loggen door op de Log Out knop te duwen. Deze roept de LogOut actie op in de Account controller. Deze verwijdert de gebruiker uit de cookie. Dit loopt ook weer via de SignInManager. (Identity, Screenshot 4)



Identity, Screenshot 4: Logout

### Uitbreiding userklasse

Het toevoegen van .net Core Identity, zorgt ook voor wijzigingen in het datamodel. Zo worden er verschillende tabellen toegevoegd waaronder de usertabel “AspNetUsers”. Deze tabel bevat alle users. Door onze eigen Userklasse hier van te laten overerven kunnen we extra velden aan deze tabel toevoegen. (Identity, Screenshot 5)



Identity, Screenshot 5: Eigen Userklasse erft over van IdentityUser

### Rollen

In de zwart op wit applicatie wordt er gebruik gemaakt van 2 rollen. Een rol Employee (Werknemer) en een Admin rol. Deze zullen beide verschillende rechten hebben. De Admin rol zal dezelfde rechten hebben als de employee rol met daarboven op nog extra rechten om het systeem te configureren.

Elke gebruiker zal in het systeem maar 1 rol hebben en zal om het systeem te gebruiken zich moeten authentiseren via een login formulier.

#### Definiëren van de rollen

Om de rollen te definiëren maken we gebruik van een static class die uitgevoerd wordt in de “configure” methode van de klasse “startup.cs”. Deze methode gaat kijken of de rollen gedefinieerd in de lijst al bestaan. Bestaan ze nog niet worden ze aangemaakt. Op deze manier zorgen we ervoor dat de rollen altijd aangemaakt zijn ook al is de database leeg.



Autorisatie – role seed klasse

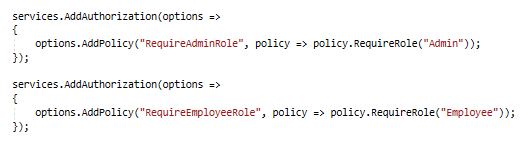


Autorisatie – role seed

#### Definiëren van Policies

Policies zijn verzamelingen van 1 of meer rollen. Door policies uit te werken worden rollen losgekoppeld van het autorisatie systeem. Een policy krijgt de rechten en aan een policy kunnen rollen worden toegevoegd.

Policies worden aangemaakt in de “configureService” methode van de klasse “startup.cs”. Deze policies worden later gebruikt in de view om te kijken of delen gerendered of niet gerenderd moeten worden.



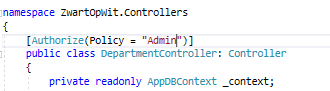
Autorisatie – policies declareren

#### Toepassen van autorisatie

Autorisatie binnen het identitity framework kan worden afgehandeld in zowel de view als de controller. Voor de ervaring van de gebruiker is het beste om van beide gebruik te maken zodat de gebruiker visueel enkel ziet waar de gebruiker toegang toe heeft.

##### Controller autorisatie

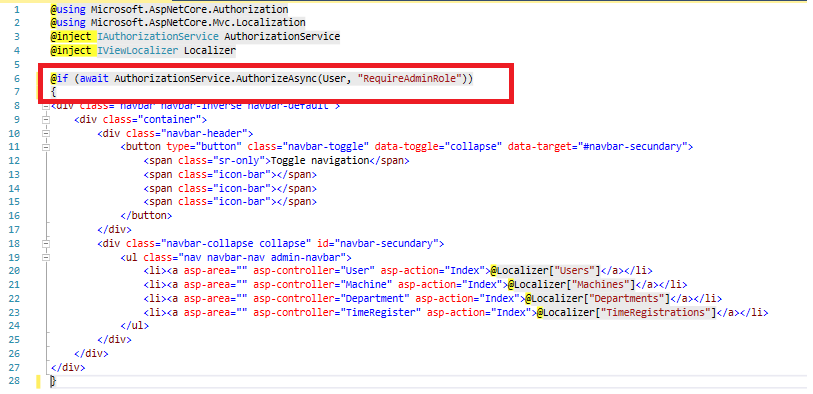
In een controller kan autorisatie op 2 niveaus afgedwongen worden. Op controller niveau of op methode niveau. Voor beide gebeurt dit met dezelfde annotatie. Voor het zwart op wit project doen wij dit enkel op controller niveau. Daarnaast maken we hier ook gebruik van policies i.p.v. rollen.



Autorisatie – Controller security

##### Autorisatie in views

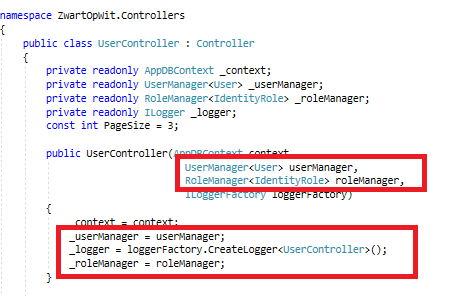
In views kan ook gekeken worden welke rollen en policies er aan de huidige gebruiker heeft. Dit maakt het mogelijk op bepaalde delen van de view juist wel of juist niet te renderen.



Autorisatie – view rendering

### Gebruikers beheren

Omdat gebruikers en rollen onder het Identity framework vallen is het niet mogelijk om deze op dezelfde manier te beheren als entiteiten die we zelf hebben aangemaakt. Het identity framework heeft hiervoor een aantal helper classes die gebruikt moeten worden. Deze classes zijn via depency injection beschikbaar in de controllers.



Gebruikers beheren – depedency injection user en rolemanager

#### UserManager

De usermanager klasse wordt gebruikt om alle handelingen met gebruikers af te handelen zoals het aanmaken, wijzigen en verwijderen van gebruikers. Maar ook kijken of de gebruik al bestaat.

#### RoleManager

De rolemanager klasse wordt gebruikt om alle handelingen met rollen af te handellen zoals kijken welke user heeft welke rol maar ook het aanmaken, wijzigen en verwijderen van rollen.

### Forgot password

#### Mail service

##### Introductie

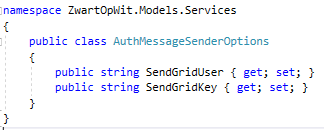
Om e-mails te kunnen sturen met onze applicatie hebben we gekozen om gebruik te maken van een externe service. Dit wordt aangeraden vanuit de best practises van Microsoft. Wij hebben gekozen van om gebruik te maken van SendGrid. SendGrid is een mail services die alle mailing afhandelt. Deze service zorgt ervoor dat je mail correct aankomen en bijvoorbeeld niet in de spam filter terecht komen. Ook geeft een mail service inzicht over wat er met de mails gebeurt die via jou account gestuurd worden

##### Mail Account configureren

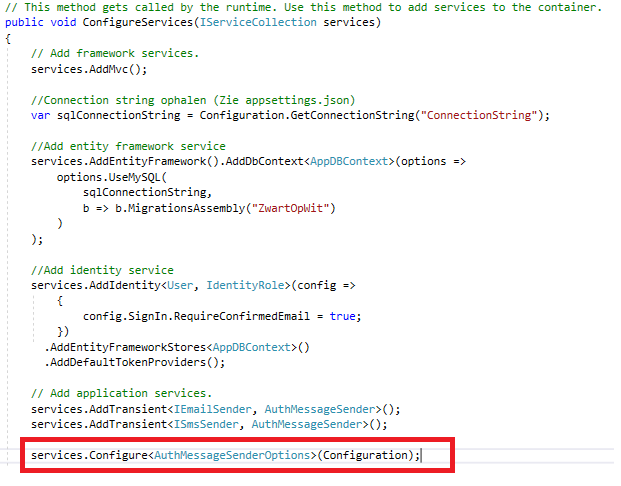
De sendgrid mail account verwacht een username (SendGridUser) en wachtwoord (SendGridKey) deze gaan we instellen door gebruik te maken van het de secret manager store in windows.

Gegevens worden met het volgende command toegevoegd aan de secret magager store: “dotnet user-secrets set SendGridUser MySendGridUserName”

Via het option pattern worden deze dan toegevoegd aan de startup.cs klasse. Hiervoor maken we gebruik van de klasse AuthMessageServiceOption die onze SendGridUser en SendGridKey meegeeft aan de startup.cs klasse



Forgot password – Options class



Forgot password - configuration

#### Forgot password form

Het “Forgot password” form vraagt het email adres van de gebruiker en stuurt dan een link met een activatie code naar het email van de gebruiker mocht het e-mail adres gekend zijn. De gebruiker kan deze link dan gebruiken om het wachtwoord opnieuw in te stellen.

## Globalisation and localization

### Introduction

Globlisation en localization is een belangrijk gegeven als je je project wil kunnen gebruiken in verschillende talen en omgevingen. Door hier rekening mee te houden in je programma kan zonder wijzingen in je code gemakkelijk afstemmen op de noden van de (toekomstige) gebruiker.

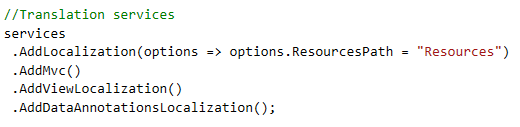
Globalisatie is een proces waarbij je applicaties maakt die verschillende culturen ondersteunen. Bij lokalisatie ga je rekening houden met een bepaalde cultuur, bijvoorbeeld een taal.

### Gebruik in ons project

Om globalisation en localisation te activeren moeten er verschillende stappen ondernomen worden. Standaard bevat .net Core hier functionaliteit voor, maar deze moet wel geconfigureerd worden.

#### Stap 1: Toevoegen van de service

De eerste stap is het toevoegen van de configuratie voor de services in de startup.cs klasse. Hier moet aan de addMVC methode ook een methode addViewLocalisation meegegeven worden met daarop de methode addDataNotationLocation (Globalisation en Localization, Screenshot 1).



Globalisation en Localization, Screenshot 1: Toevoegen van de service

Addlocalization voegt de localization service toe aan de service container.

AddViewLocalization voegt ondersteuning toe voor localized files (resources die eindigen op .nl.resx, .en.resx, .fr.resx, …)

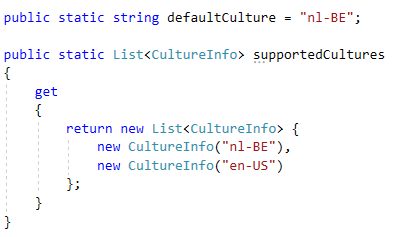
AddDataAnnottations geeft ondersteuning voor de localized DataAnnotaties in de views. (@Localizer["DepartmentName"])

Door deze methodes de implementeren heb je de volledige translations service geactiveerd.

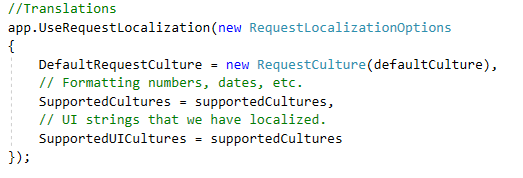
#### Stap 2: Het configureren van Localizations

De localizations waarvan je gebruikt maakt moeten ook gedefinieerd worden. Dit doe je in de startup.cs klasse in de methode configure.

Hier gaan we een lijst zetten van default (localizations) (Globalisation en Localization, Screenshot 2). In ons geval Engels en Nederlands (Globalisation en Localization, Screenshot 3).



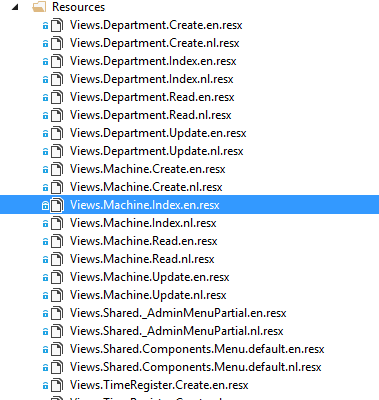
Globalisation en Localization, Screenshot 2: Het configureren van localizations



Globalisation en Localization, Screenshot 3: Het configureren van localizations

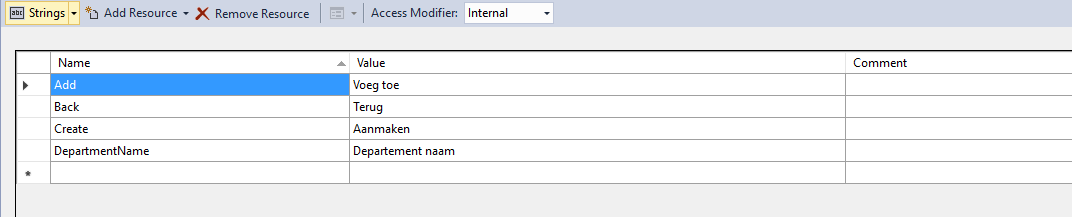
#### Stap 3: Het aanmaken van resource files

De derde stap is het aanmaken van de resource files. Deze files worden aangemaakt per view per culture. De map waar deze staan hebben we eerder al geconfigureerd in stap 1. (Globalisation en Localization, Screenshot 4)



Globalisation en Localization, Screenshot 4: Resource mappen structuur

Een resource file wordt altijd aangemaakt met op het einde een culture info. In dat geval .nl, zo weet het systeem welke file bij welke taal hoort. Een resource file werkt op de basis van een key en een value. De key blijft in iedere culture hetzelfde, terwijl de value de eigenlijke vertaling is. (Globalisation en Localization, Screenshot 5)



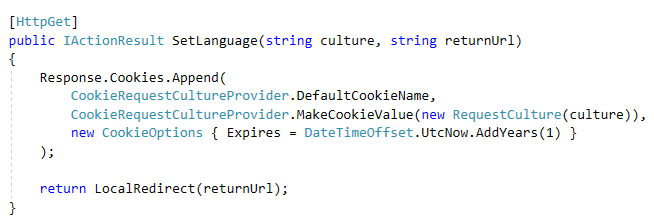
Globalisation en Localization, Screenshot 5: Resource file

#### Stap 4: Select language, partial view

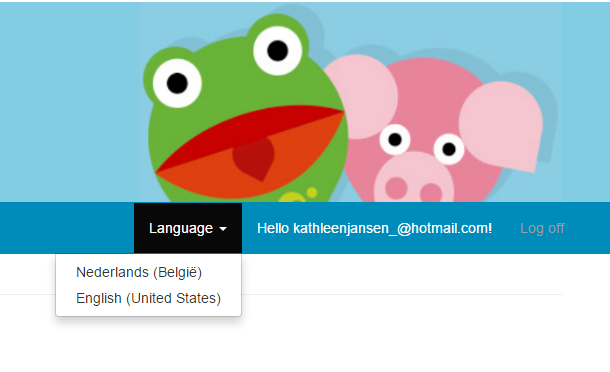
Omdat je natuurlijk eenvoudig van taal wil kunnen wisselen in het programma hebben we een partial view toegevoegd (Globalisation en Localisation, Screenshot 6). Deze gaat de culture info in een cookie opslaan (Globalisation en Localisation, Screenshot 7).



Globalisation en Localization, Screenshot 6: Partial View



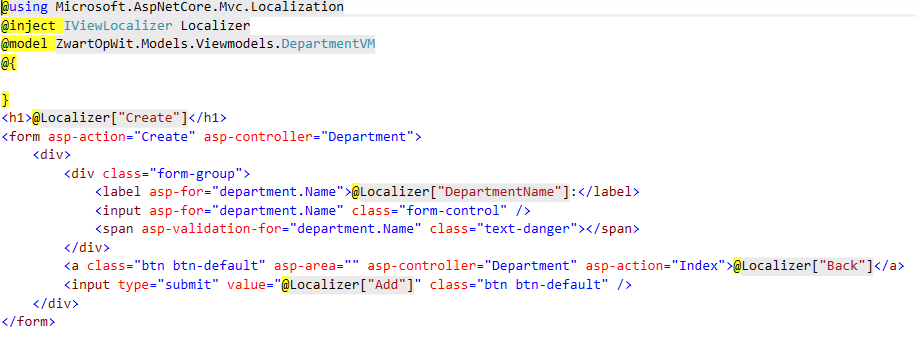
Globalisation en Localization, Screenshot 7: Controller die de culture cookie zet.



Rendering van partial view

#### Stap 5: Localization toepassen in een view

Na dat alles geconfigureerd is, kunnen we translations gaan gebruiken. We doen dit in onze view aan de hand van de @Localizer annotatie. Deze annotatie handelt alles af rond vertalingen, als er een key gevonden wordt in de resource file (Globalisation en Localisation, Screenshot 8).



Globalisation en Localization, Screenshot 8: Implementatie Localizer

## Components

### Introduction

View Components zijn een feature die pas beschikbaar zijn sinds .net Core. Hierdoor zijn ze wellicht ook iets minder bekend. Ze zijn echter wel zeer handig en makkelijk in gebruik.

### Components vs. Partial View

Hoewel components vergelijkbaar zijn met partial views zijn components een pak krachtiger. Zo gebruiken components geen model binding en hangen ze enkel af van de data die jij geeft als je ze oproept.

Een partial view kan geen business logica bevatten buiten de view om. Als je hier wel logica in wilt gebruiken ben je afhankelijk van de controller die de partial oproept.

Omwille van deze voordelen hebben we gekozen om gebruik te maken van een component bij onze menubalk. Dit was specifiek nodig omdat onze menubalk eigenlijk bestaat uit een lijst van machines en deze worden gehaald uit de database.

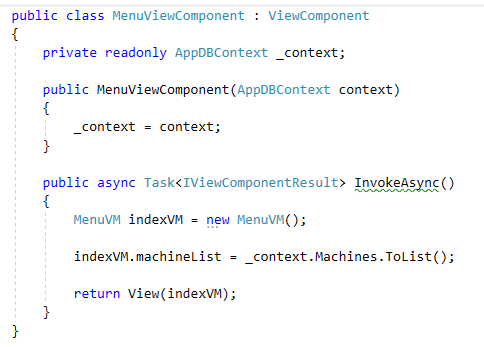
Kenmerken van een view component:

* Rendert een gedeelte in plaats van een hele repsons
* Kan zowel parameters als business logica bevatten
* Bestaat altijd uit een view en een conrtoller
* Doet meestal beroep op een layout page

Je kan view components makkelijk gebruiken in volgende situaties:

* Dynamische navigatie menu’s
* Login panelen
* Tag Clouds
* Winkelwagentjes
* …

Volgende screenshot (Component, Screenshot 1) geeft weer dat we gebruik maken van de component en niet van de controller. We doen dus een database actie buiten de controller om.



Component, Screenshot 1: Component klasse

Daarnaast hebben we de view van onze component aangemaakt (Component, Screenshot 2). Om toch de structuur te volgen wordt er hier wel gecommuniceerd via een viewmodel (menuVM). Deze wordt opgevuld door de achterliggende component klasse (Component, Screenshot 1)

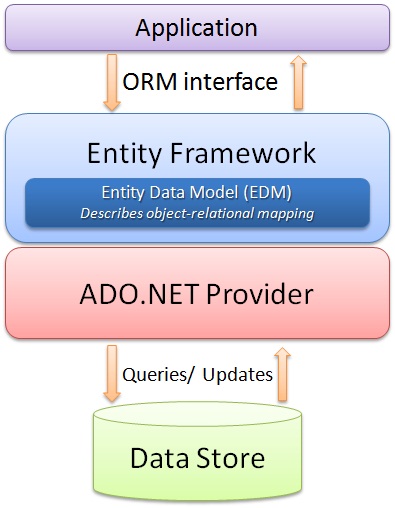


Component, Screenshot 2: View van de component

## Entity Framework

### Inleiding

Entity Framework is een open source ORM-framework. ORM staat voor Object Relational Mapping. Entity Framework is een set van technologieën in ADO.NET dat het ontwikkelen van data georiënteerde software applicaties ondersteund. Het maakt het developers gemakkelijk om het datamodel in code te gebruiken. Wij installeerde Entity Framework voor .net Core aan de hand van een Nuget Package.



### Wat doet Entity Framework?

Entity Framework genereert objecten en entiteiten volgens de tabellen in de database en voorziet het mechanisme voor het doen van data manipulaties. Ook vergemakkelijkt EF het onderhouden van relaties tussen tabellen.

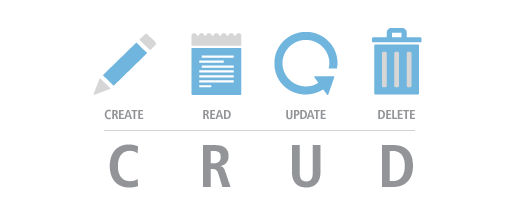
### Voordelen

Entity Framework heeft als voordeel dat we het model beter kunnen voorstellen door gebruik te maken van relaties tussen de entiteiten. Ook maakt EF het mogelijk om logica in verband met data acces niet meer te schrijven op database niveau aan de hand van SQL, maar wel in programmeer code, in ons geval c# in combinatie met Linq. Dit geeft ook als voordeel dat de onderliggende data store makkelijker kan vervangen worden zonder te veel overhead, aangezien alle data acces logica zich op een hoger niveau bevindt.

## CRUD

Voor het beheren van data maken we gebruik van CRUD. Dit staat voor de acties Create, Read, Update en Delete.

* Create : Het toevoegen van nieuwe gegevens
* Read: Het opvragen van gegevens
* Update: Het wijzigen van gegevens
* Delete: Het verwijderen van gegevens

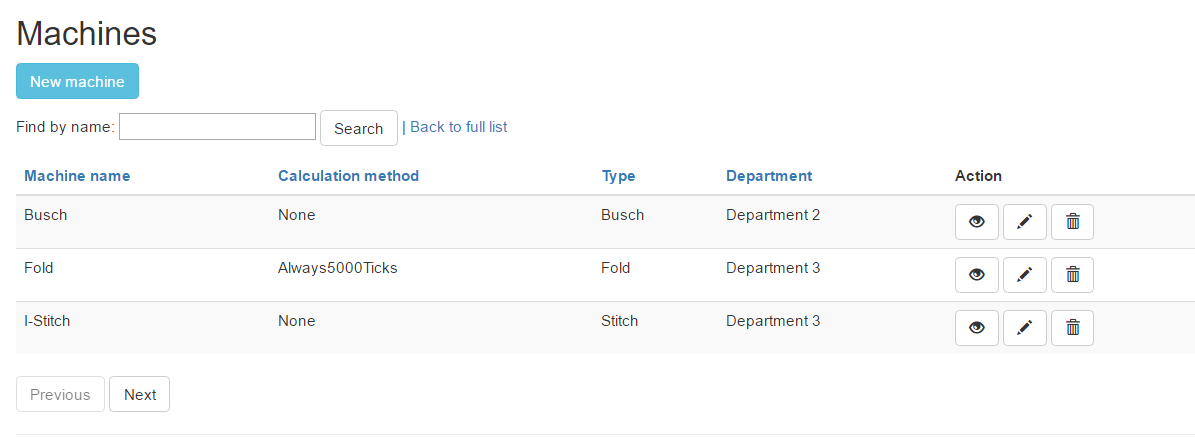


We zullen de werking van de CRUD verder verduidelijken door een uitgewerkte CRUD te behandelen. Hieronder een overzicht van de CRUD Machines.

### Index

De index pagina is de pagina waarvan alle CRUD acties beginnen. De index pagina geeft een overzicht van de records en laat toe om deze te filteren en te sorteren. Als er genoeg records zijn wordt er ook gebruik gemaakt van pagina’s. Deze pagina’s zullen dan een x-aantal records per pagina tonen om te voorkomen dat je het overzicht in de lijst verliest.

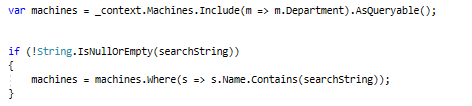
In de controller bestaat er voor de index pagina enkel een methode van het type get omdat er enkel data opgehaald wordt en geen data gemanipuleerd word. Deze methode handelt alle logica af.



CRUD – Index overzicht

#### Filteren

Het filteren gebeurt door dynamisch like functies voor de doorzoekbare velden toe te voegen als er een waarde in het zoek veld is ingevuld.



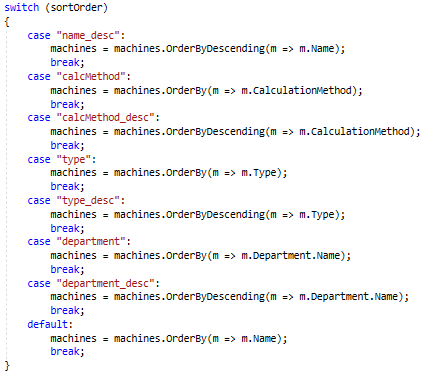
CRUD – index filter toepassen

#### Rangschikken

Het rangschikken gebeurt door te kijken of er de vorige keer al op hetzelfde veld gesorteerd is. Als dit zo is keer de sortering dan om anders sorteer aflopend op het nieuwe veld.



CRUD – Index rangschikking bepalen

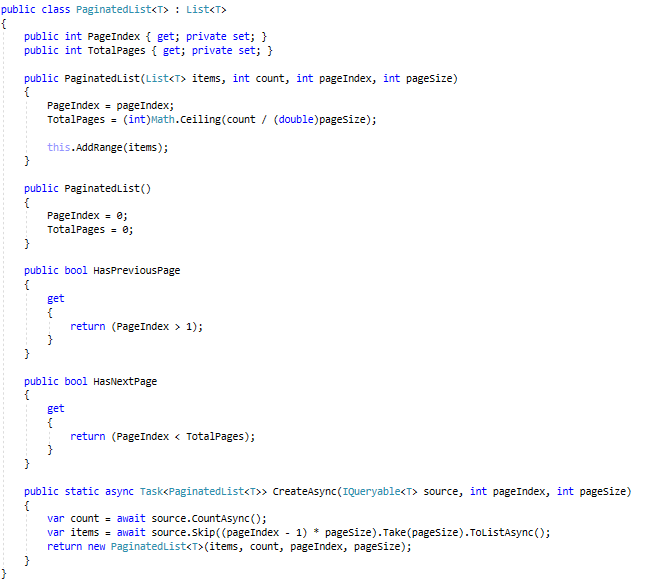


CRUD – index rangschikken doen

#### In pagina’s verdelen

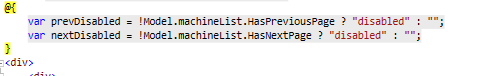
Voor het verdelen van de lijst in pagina’s maken we gebruik van een helper klasse. Deze helper klasse paginatedList is een extensie op de gewone list. De paginated list bevat enkel extra attributen die nodig zijn voor het beheren van de pagina’s. Het totaal aantal pagina’s en de huidige pagina.

Er zijn ook extra methodes aangebracht om te kijken of er nog een vorige of volgende pagina is. Ook is er een constructor toegevoegd om de nieuwe attributen correct in te vullen.

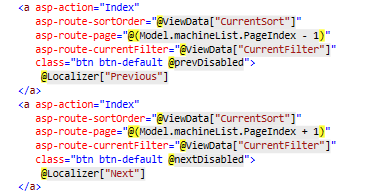


CRUD – Index paginated list class

In de view wordt aan de hand van de paginatedList bepaald of er nog een volgende of vorige pagina is. Als de knop volgende of vorige geactiveerd wordt zal opnieuw de index methode aangeroepen worden met de correcte parameters en de volgende pagina geladen worden.



CRUD – Index pagination determination has next, has previous page

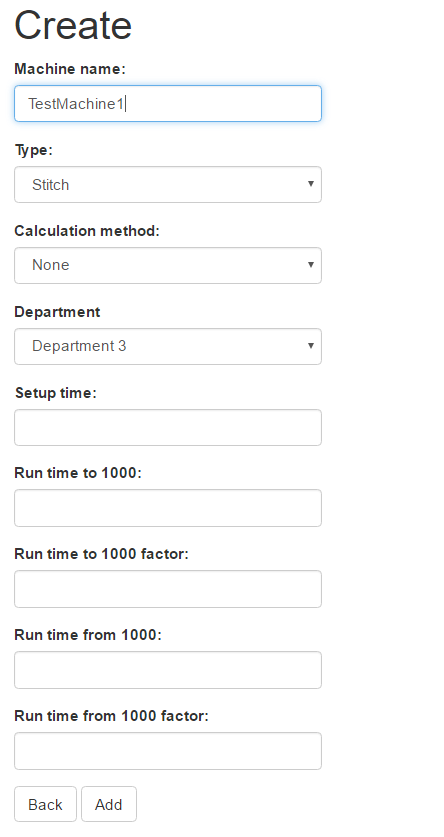


CRUD – Index pagination next previous

### Create

De create bestaat uit 1 form en in de controller zijn er 2 methodes voor de create een HttpGet en HttpPost methode. Deze methodes zorgen ervoor dat de create pagina geopend kan worden en het form verwerkt kan worden.

In de HttpGet methode zit er logica om de lijst van departementen op te halen om zo de selectie box op het create form te vullen.

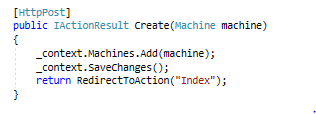


CRUD – Create form



CRUD – Create get method

In de HttpPost methode wordt de nieuwe aangemaakt. Voor wegschrijven van het machine record naar de database gebruiken we entity framework.



CRUD – Create post method

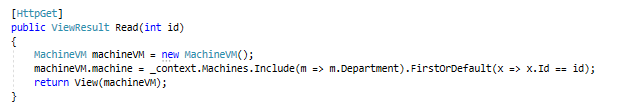
### Read

De read pagina laat alle parameters van een machine zien. Het is niet mogelijk om hier parameters van de machine te wijzigen. Deze pagina is handig als gebruikers geen rechten mogen hebben om de entiteit aan te passen maar wel om deze te zien.



CRUD – Read view

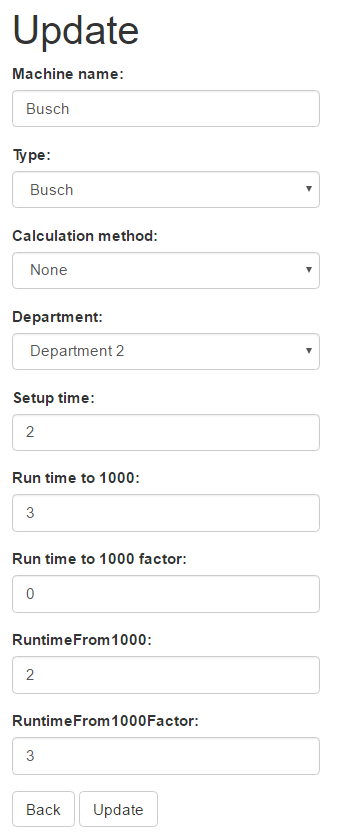
Om de department naam weer te geven maken we gebruik van een include hiermee wordt het gerelateerde department record ook in de query opgehaald.



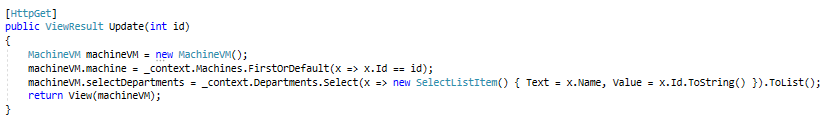
CRUD – Read method

### Update

Bij de update wordt er net als bij de create gebruik gemaakt van een HttpGet en HttpPost methode. De HttpGet methode zal het formulier vullen maar t.o.v. de create wordt nu ook het geselecteerde record opgehaald. Ook hier wordt opnieuw de data opgehaald om de selectieboxen te vullen.

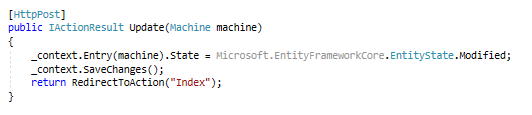


CRUD – Update from



CRUD – Update get method

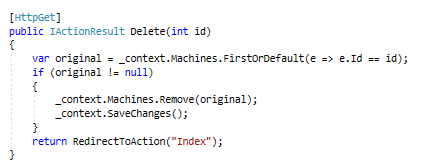
In de HttpPost methode wordt het formulier verwerkt en de wijziging bewaard in de database. Hier gebruiken we Entity Framework voor.



CRUD – Update post method

### Delete

De delete bestaat uit 1 methode als deze van de index pagina wordt opgeroepen word het huidig record verwijderd. Om dit te kunnen doen maken we gebruik van enity framework. Voor de delete wordt uitgevoerd wordt er eerst gekeken of de machine wel bestaat.



CRUD – delete method

## Patterns

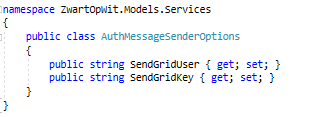
### Inleiding

Design patterns zijn stukken code die te herbruiken zijn om gekende problemen op te lossen. In ons project hebben wij er enkele gebruikt omdat dit aan de ene kant gemakkelijk was en aan de andere kant omdat dit werd verplicht door .net core wegens best pratices regels.

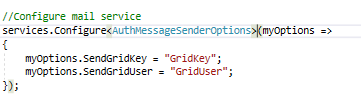
### Options pattern

Het options pattern is een pattern dat gebruikt wordt om via een klasse een groep van geralteerde settings weer te geven. Deze klasses kunnen dan bijvoorbeeld vertaald worden naar JSON. Voor het options pattern is het belangrijk dat de klasses een parameterloze constructor hebben. Deze opties zijn dan via dependency injection beschikbaar in andere klasses.

Dit pattern is toegepast voor het configureren van de mail service.



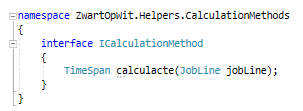
Options pattern – Options klasse



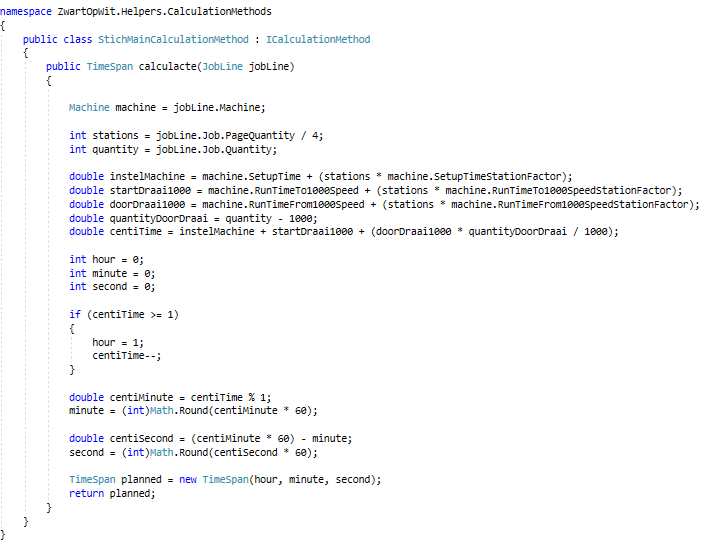
Options pattern – Options klasse vullen

### Simple factory pattern (Calculation method)

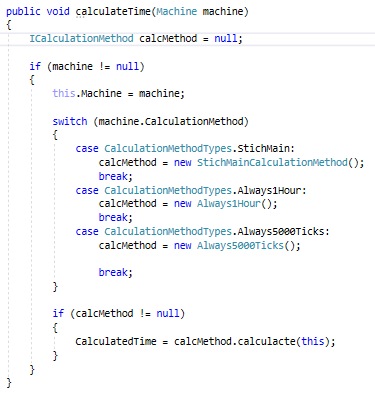
Het simple factory pattern is een pattern waar aan de hand van een enum een bepaalde method op een klasse aangeroepen wordt. Wij hebben dit gebruikt om de calculatie methodes aan de machines te koppellen. Op de machine wordt een waarde van de enum gekozen die daarna overeenkomt met een methode in de calculatie logica. Alle klassen die in de factory gebruikt worden implementeren een bepaalde interface.



Simple factory pattern – Interface



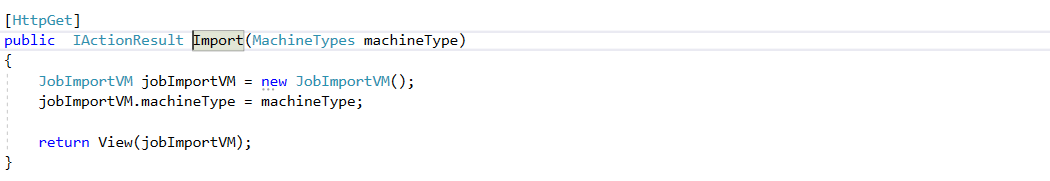
Simple factory pattern – Calculation method implementing the interface



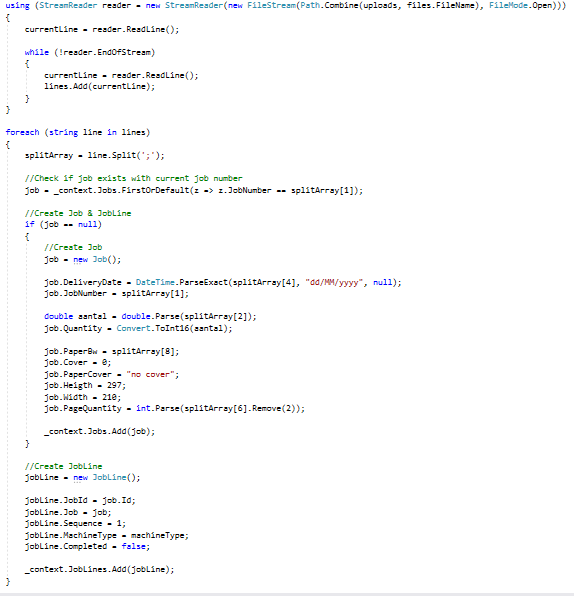
Simple factory pattern – Factory

## CSV Import

De csv bestanden die vanuit filemaker doorgestuurd worden kunnen via de code hieronder geupload worden. Via de Interface IformFile wordt het bestand naar de, in dit geval map ‘uploads’ gestuurd.



Het bestand wordt daarna opgeroepen via StreamReader, gecombineerd met FileStream, door gebruik te maken van het eerst gekozen pad. Via split worden dan de juiste gegevens in Job en JobLine gezet en aan de context toegevoegd. Als alle gegevens verzameld zijn wordt de context naar de database geschreven.



# Werking programma

## Users beheren

### Introductie

Als admin-gebruiker kan je gebruikers beheren. Gebruikers kunnen inloggen op het systeem. Er bestaan 2 rollen. Een admin-rol en een gewone gebruiker-rol.

De optie gebruikers beheren is enkel zichtbaar in de admin navigatiebalk. Deze is enkel als je een admin-user bent beschikbaar.

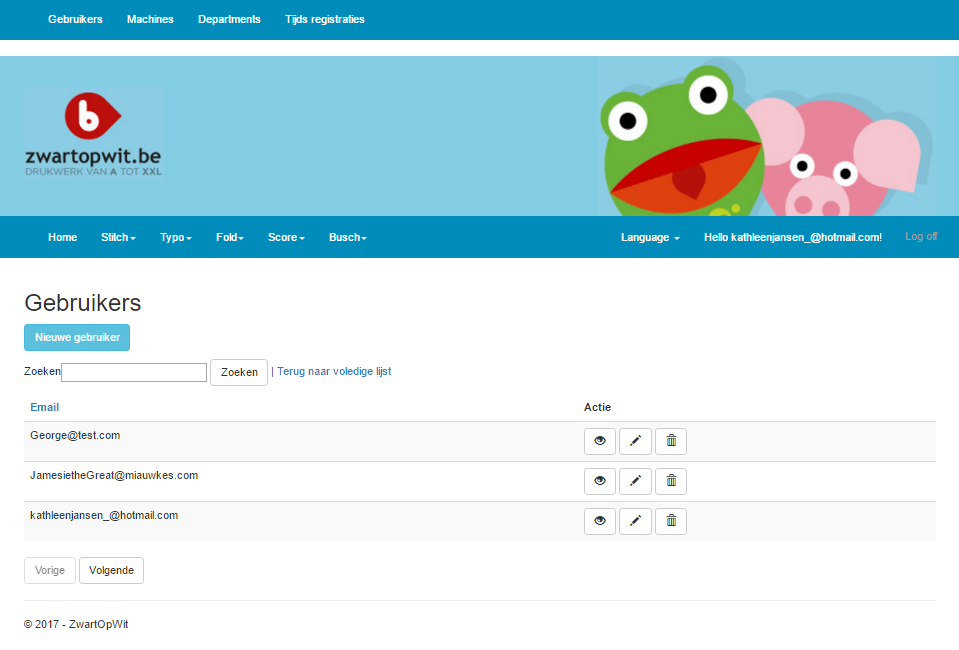
### Index pagina

Nadat je op de knop “Gebruikers” in de admin navigatiebalk hebt geklikt, kom je op de index pagina van “Gebruikers”. Deze geeft een lijst weer van alle huidige gebruikers.

Hier heb je verschillende mogelijkheden en opties. Voor het beheren van de gebruikers werken we met CRUD (zie hoofdstuk CRUD). Verder heb je de mogelijkheid om een gebruiker te filteren door gebruik te maken van “Zoeken”. Ook zijn er paginaknoppen voorzien voor als de lijst te uitgebreid wordt.



Screenshot: Navigatiebalk voor admin – ga naar Gebruikers

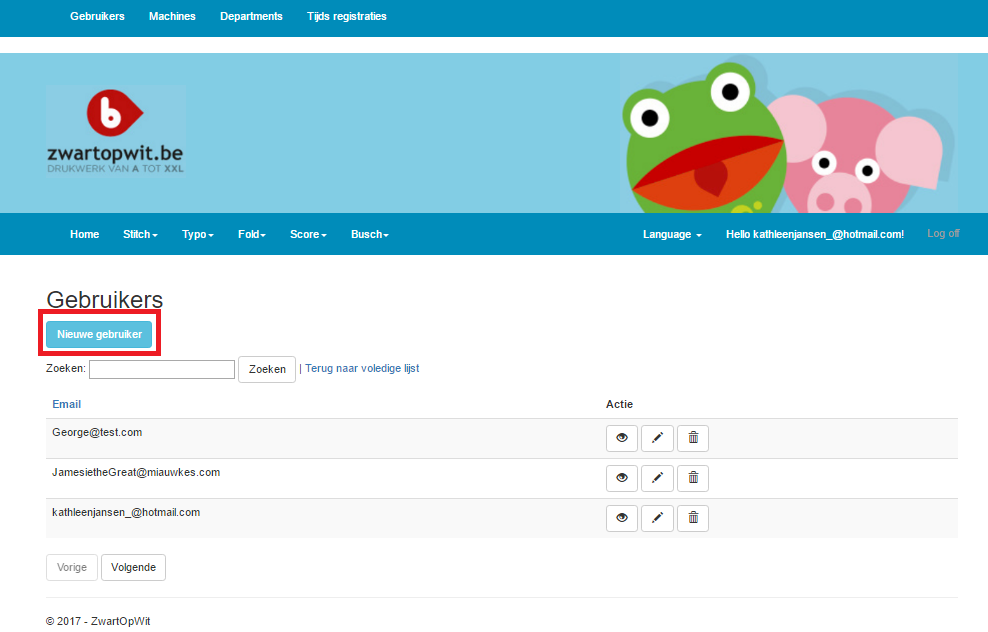


Screenshot Gebruikers – Index pagina

### User CRUD

#### Create

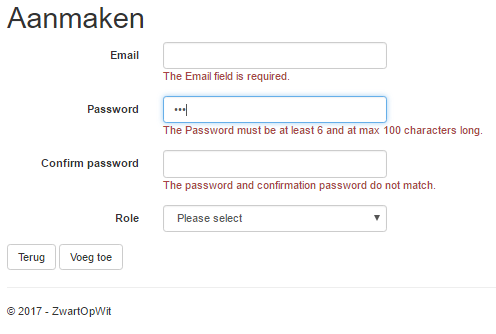
Als we op de knop “Nieuwe gebruiker” klikken komen we op de create pagina voor een nieuwe gebruiker. Hier kunnen we een nieuwe gebruiker aanmaken.



Screenshot: Index pagina – Nieuwe gebruiker aanmaken

Een gebruiker bestaat uit een e-mailadres (username), een paswoord en een rol. Om typfouten tegen te gaan wordt er 2x naar het paswoord gevraagd. Uit rol kan men kiezen tussen admin of employee. Alle velden zijn verplicht in te vullen, anders krijg je een foutmelding. Ook wordt er gevalideerd op de lengte van de ingevoerde tekst.

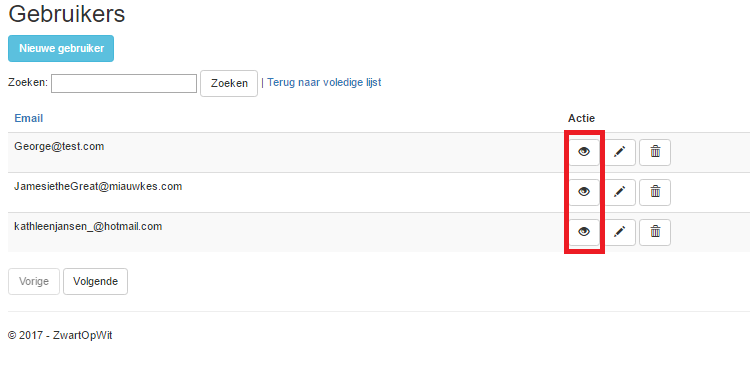
Meer informatie over het gebruik van rollen en identity vind je terug in hoofdstuk 8.6 Identity.



Screenshot: Gebruikers – Nieuwe gebruiker aanmaken + validatie

#### Read

Door op het oog – icoontje te klikken in de gebruikers lijst, krijg je een weergave van de details van de gekozen gebruiker.



Screenshot: Gebruikers - Read

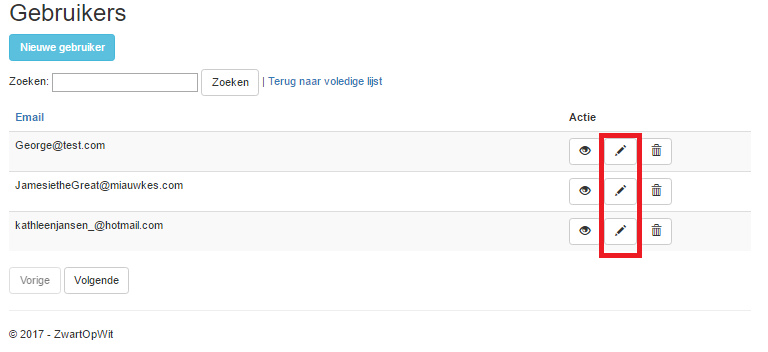
Deze worden weergegeven in een overzichtelijke lijst. Het paswoord van de gebruiker wordt natuurlijk niet getoond. Indien je wil terugkeren naar de vorige pagina klik je op “terug”.



Screenshot: Gebruikers – Read detailvoorstelling in lijst

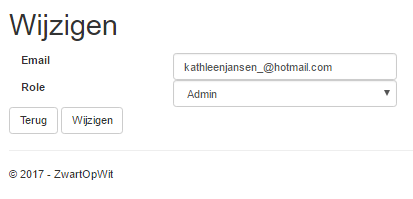
#### Update

Als je op het potlood icoontje klikt, kan je de gegevens van de gekozen gebruiker wijzigen.



Screenshot: Gebruikers - Update

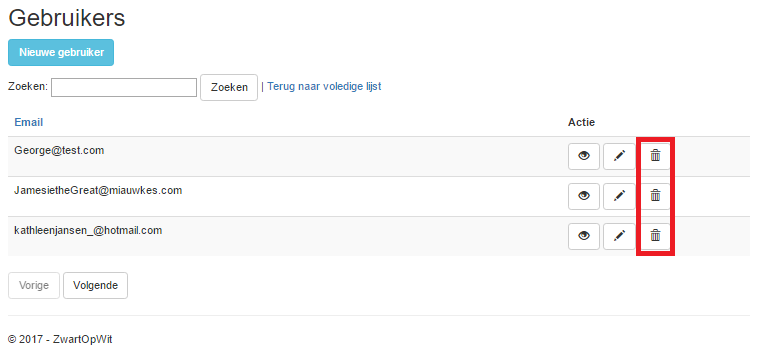
Deze heeft dezelfde invoervelden als de Create/Aanmaken pagina. De reeds bestaande gegevens zijn opgehaald en worden ineens ingevuld. Indien de gebruiker klaar is met updaten klikt hij onderaan op wijzigen en komt hij terug op de index pagina.



Screenshot: Gebruikers – Wijzigen detail pagina

#### Delete

Om een departement te verwijderen druk je op het vuilbak-icoontje op de index pagina. Het departement wordt dan verwijderd.



Screenshot: Gebruikers – Verwijderen

### Zoeken

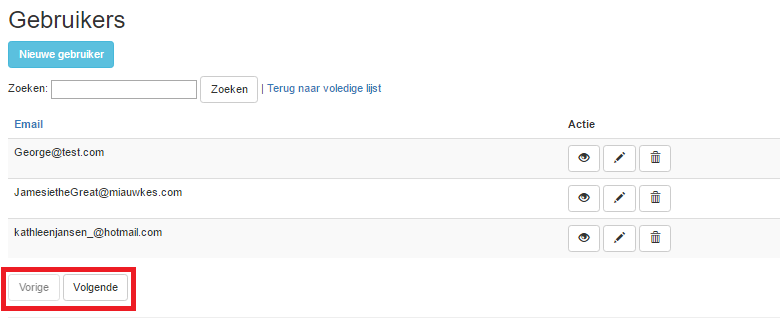
Je kan een gebruiker gemakkelijk terugvinden door de naam of een gedeelte hiervan in te geven in de zoekbalk. Deze is niet hoofdletter gevoelig. Wil je daarna terugkeren naar de volledige lijst druk je op “Terug naar volledige lijst”.



Screenshot: Gebruikers – Filteren

### Navigeren

Omdat de mogelijkheid bestaat dat de pagina veel gebruikers bevat, hebben we navigatieknoppen voorzien. Met de knoppen “Vorige” en “Volgende” kan je gemakkelijk naar de volgende pagina. De knop “Vorige” is niet klikbaar op de eerste pagina, de knop “Volgende” is niet klikbaar op de laatste pagina. Het aantal gebruikers dat wordt weergegeven is momenteel ingesteld op 3. Deze kan nog gemakkelijk aangepast worden in de code.



Screenshot: Gebruikers – Navigeren door index pagina

## Departementen beheren

### Introductie

Als admin-gebruiker kan je departementen beheren. Machines behoren tot een departement.

De optie departementen beheren is enkel zichtbaar in de admin navigatiebalk. Dus enkel als je een admin-user bent is deze beschikbaar.

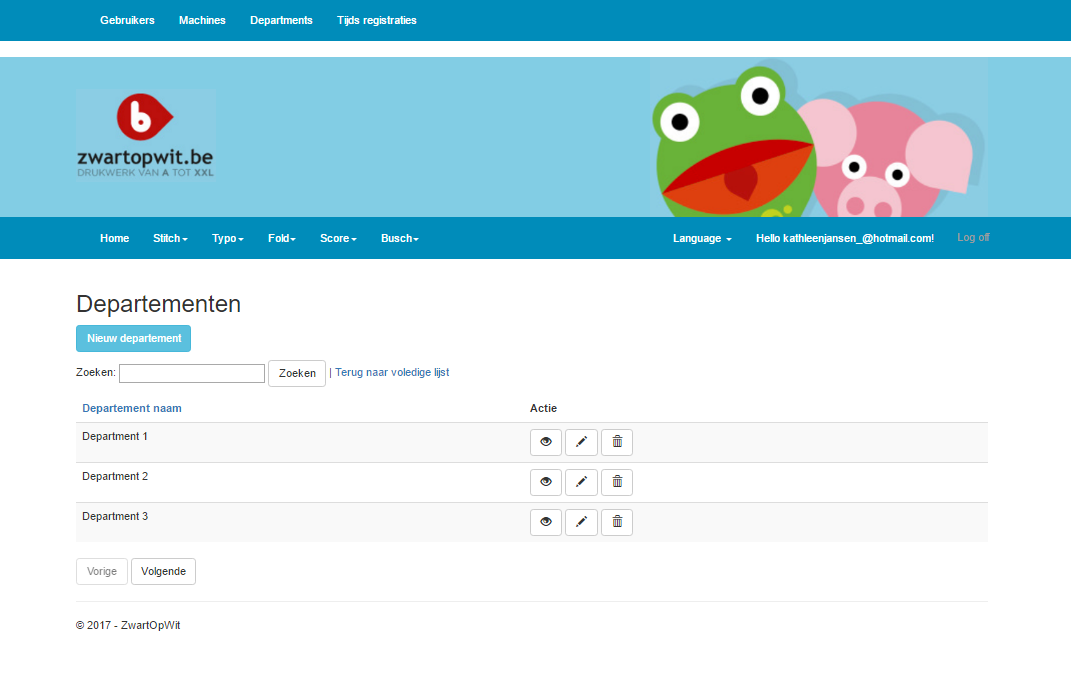


Screenshot: Navigatiebalk voor admin – Ga naar Departments

### Index pagina

Nadat je op de knop “Departments” in de admin navigatiebalk hebt geklikt, kom je op de index pagina van “Departments”. Deze geeft een lijst weer van alle huidige departementen.

Hier heb je verschillende mogelijkheden en opties. Voor het beheren van de departments werken we met CRUD (zie hoofdstuk CRUD). Verder heb je de mogelijkheid om een department te filteren door gebruik te maken van “Zoeken”. Ook zijn er paginaknoppen voorzien voor als de lijst te uitgebreid wordt.

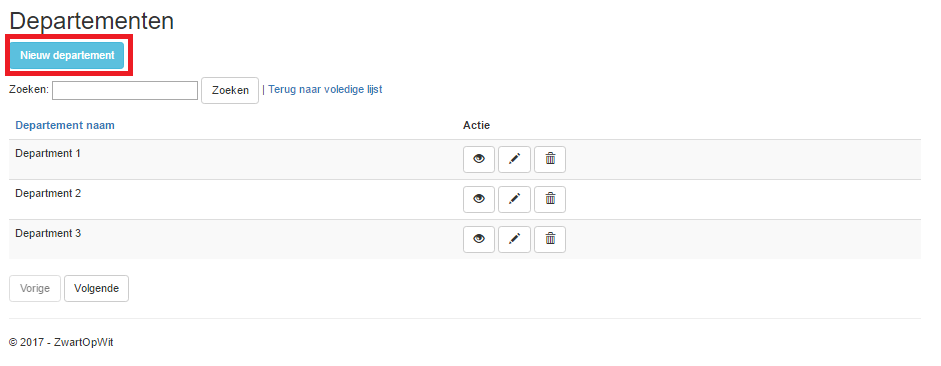


Screenshot: Departmententen – Index pagina

### Department CRUD

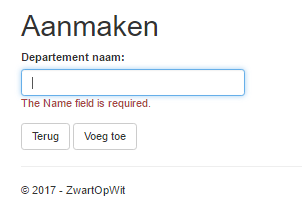
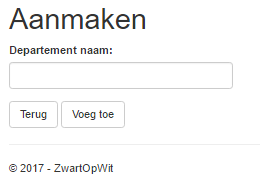
#### Create

Als we op de knop “Nieuw departement” klikken komen we op de create pagina voor een departement. Hier kunnen we een nieuw departement aanmaken.



Screenshot: Departementen – Nieuw departement toevoegen

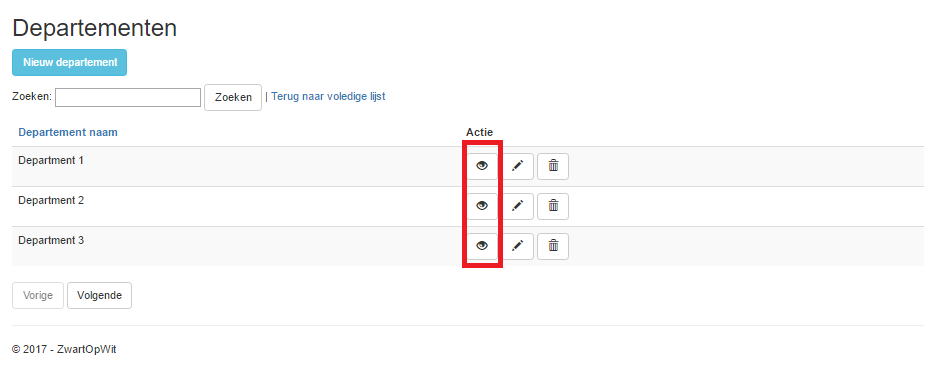
Een departement bestaat enkel uit een naam. Deze is verplicht in te vullen, anders krijg je een foutmelding. Deze moet ook minimum 3 karakters bevatten.



Screenshot: Departementen – Aanmaken + validatie

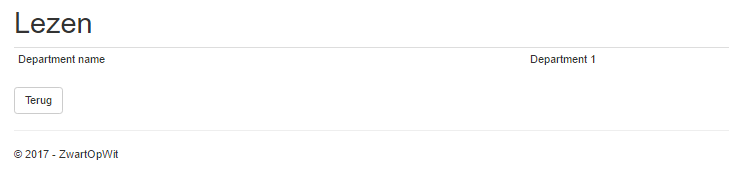
#### Read

Door op het oog – icoontje te klikken in de departementen lijst, krijg je een weergave van de details van het departement.



Screenshot: Departementen - Read

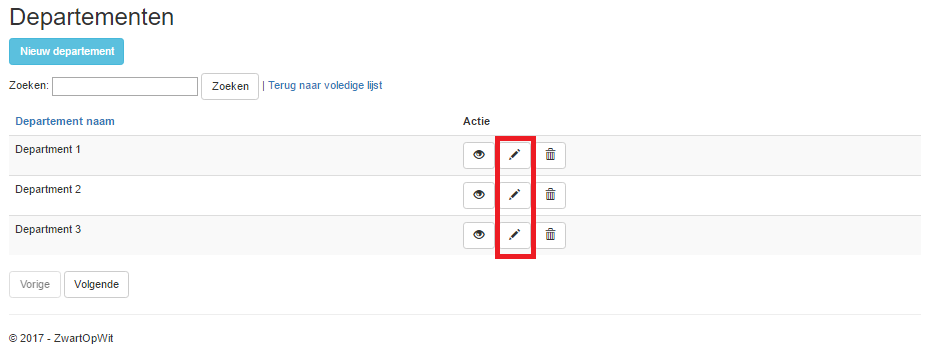
Deze worden weergegeven in een overzichtelijke lijst. Indien je wil terugkeren naar de vorige pagina klik je op “terug”.



Screenshot: Departementen – Read detail voorstelling

#### Update

Als je op het potlood icoontje klikt kan je de gegevens van het gekozen departement wijzigen.



Screenshot: Departementen - Update

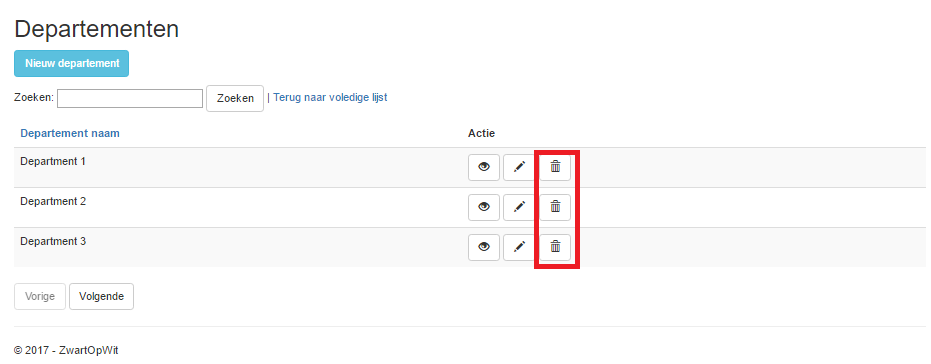
Deze heeft dezelfde invoervelden als de Create/Aanmaken pagina. De reeds bestaande gegevens zijn opgehaald en worden ineens ingevuld. Indien de user klaar is met updaten klikt hij onderaan op wijzigen en komt hij terug op de index pagina.



Screenshot: Departementen – Wijzigen detail voorstelling

#### Delete

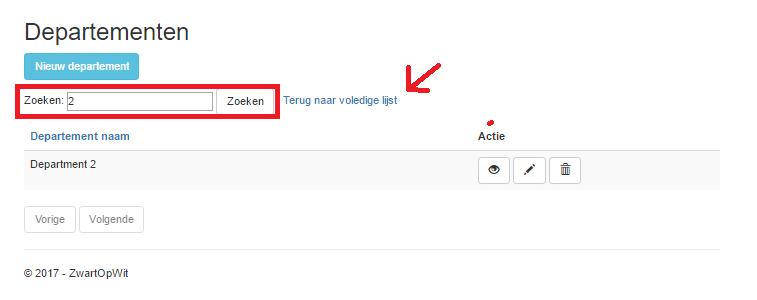
Om een departement te verwijderen druk je op het vuilbak-icoontje op de index pagina. Het departement wordt dan verwijderd.



Screenshot: Departementen - Verwijderen

### Zoeken

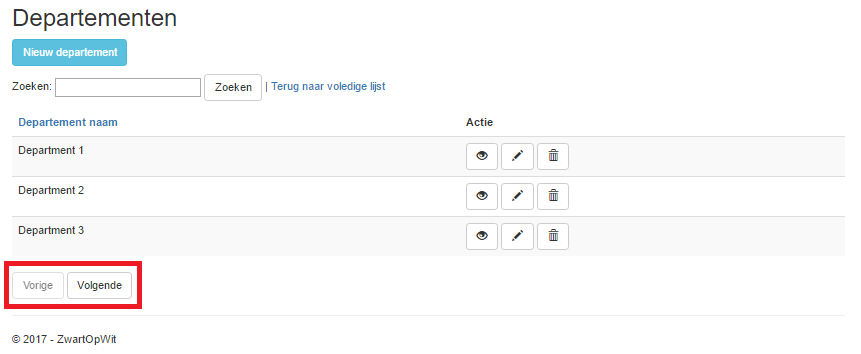
Je kan een departement gemakkelijk terugvinden door de naam of een gedeelte hiervan in te geven in de zoekbalk. Deze is niet hoofdletter gevoelig. Wil je daarna terugkeren naar de volledige lijst druk je op “Terug naar volledige lijst”.



Screenshot: Departementen - zoeken

### Navigeren

Omdat de mogelijkheid bestaat dat de pagina veel departementen bevat, hebben we navigatieknoppen voorzien. Met de knoppen “Vorige” en “Volgende” kan je gemakkelijk naar de volgende pagina. De knop “Vorige” is niet klikbaar op de eerste pagina, de knop “Volgende” is niet klikbaar op de laatste pagina. Het aantal departementen dat wordt weergegeven is momenteel ingesteld op 3. Deze kan nog gemakkelijk aangepast worden in de code.



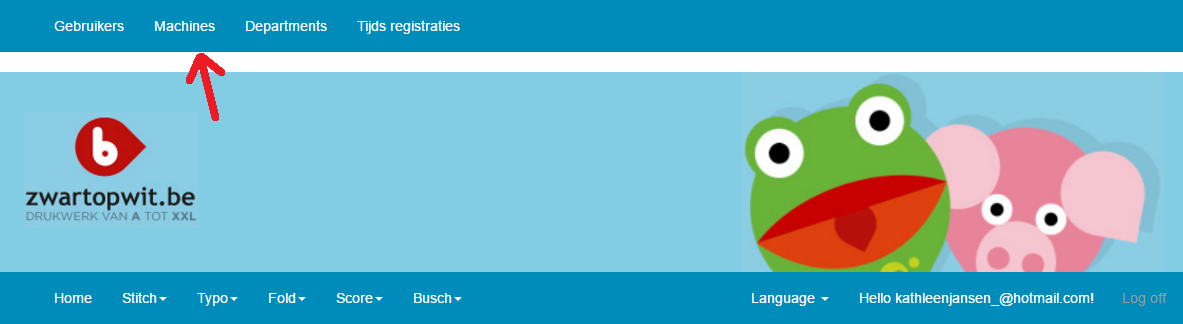
Screenshot: Departementen – Navigeren door index pagina

## Machines beheren

### Introductie

Als admin-gebruiker kan je machines beheren. Met machines bedoelen we de afwerkingsmachines van Zwart op Wit. Vaak heeft een drukwerk nog afwerking nodig. Dit kan gaan van nietjes, vouwen, rillen,… Alles om het vouwwerk tot in de details af te werken.

De optie machines beheren is enkel zichtbaar in de admin navigatiebalk. Dus enkel als je een admin-user bent is deze beschikbaar.

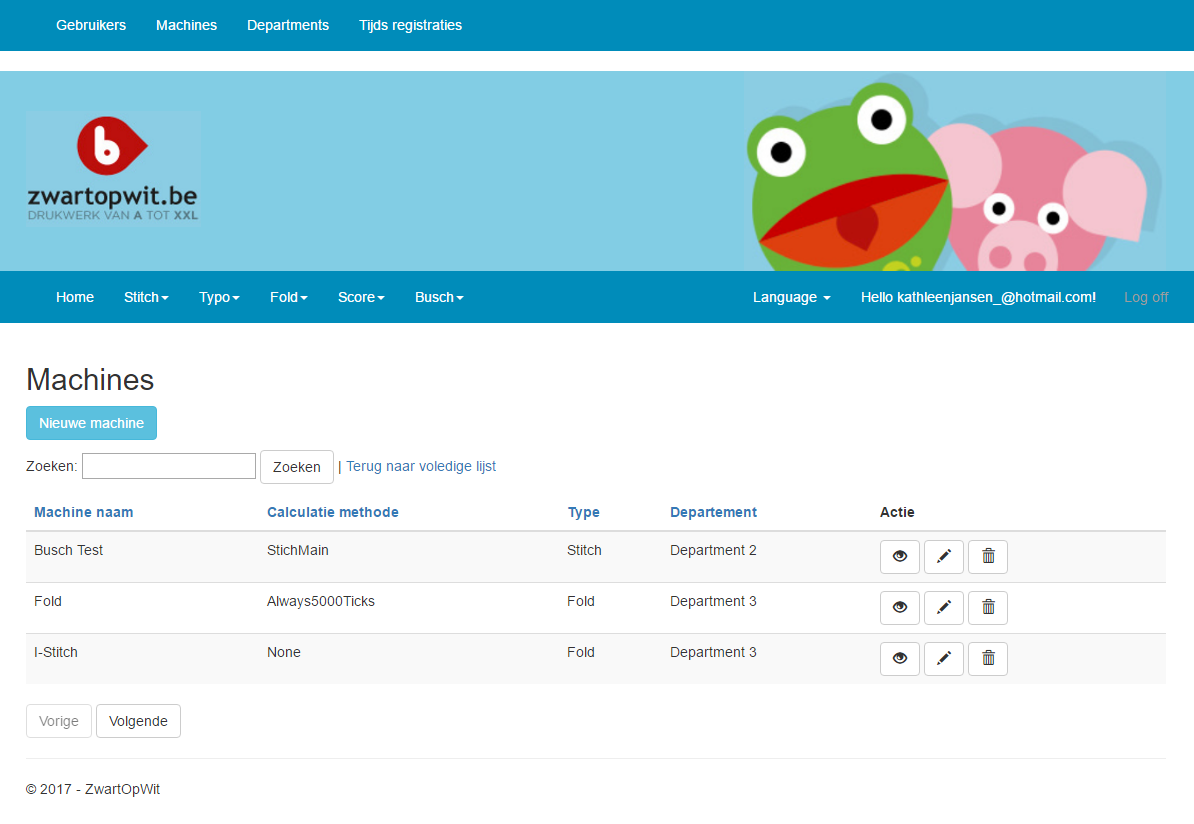


Screenshot: Navigatiebalk voor admin – Ga naar machines beheren

### Index pagina

Nadat je op de knop “Machines” in de admin navigatiebalk hebt geklikt, kom je op de index pagina van “Machines”. Deze geeft een lijst weer van alle huidige machines.

Hier heb je verschillende mogelijkheden en opties. Voor het beheren van de machines werken we met CRUD (zie hoofdstuk CRUD). Verder heb je de mogelijkheid om een machine te filteren door gebruik te maken van “Zoeken”. Ook zijn er paginaknoppen voorzien voor als de lijst te uitgebreid wordt.

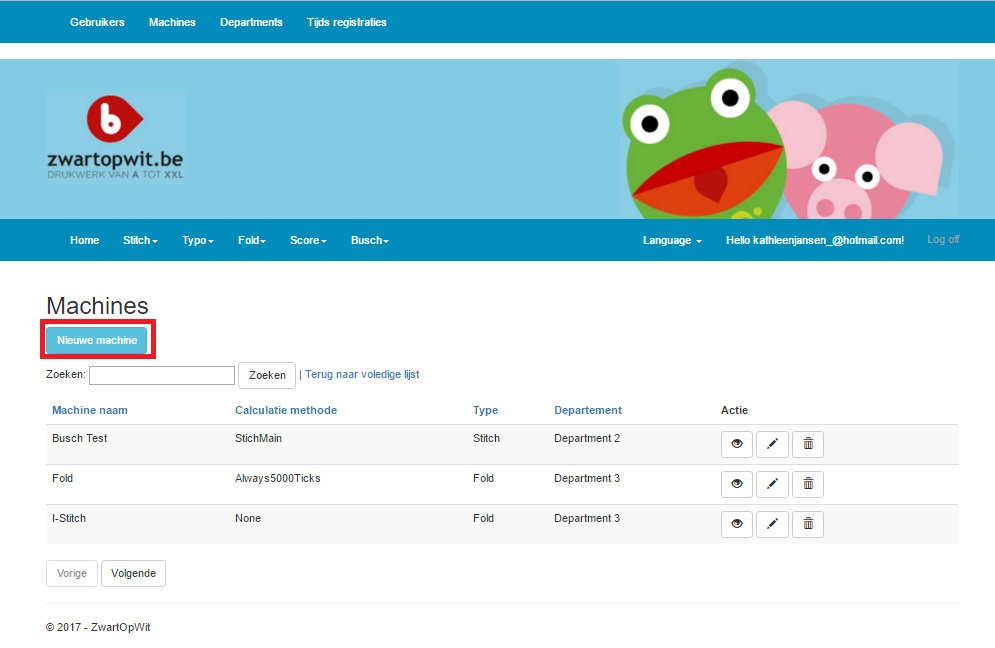


Screenshot: Machines – Index pagina

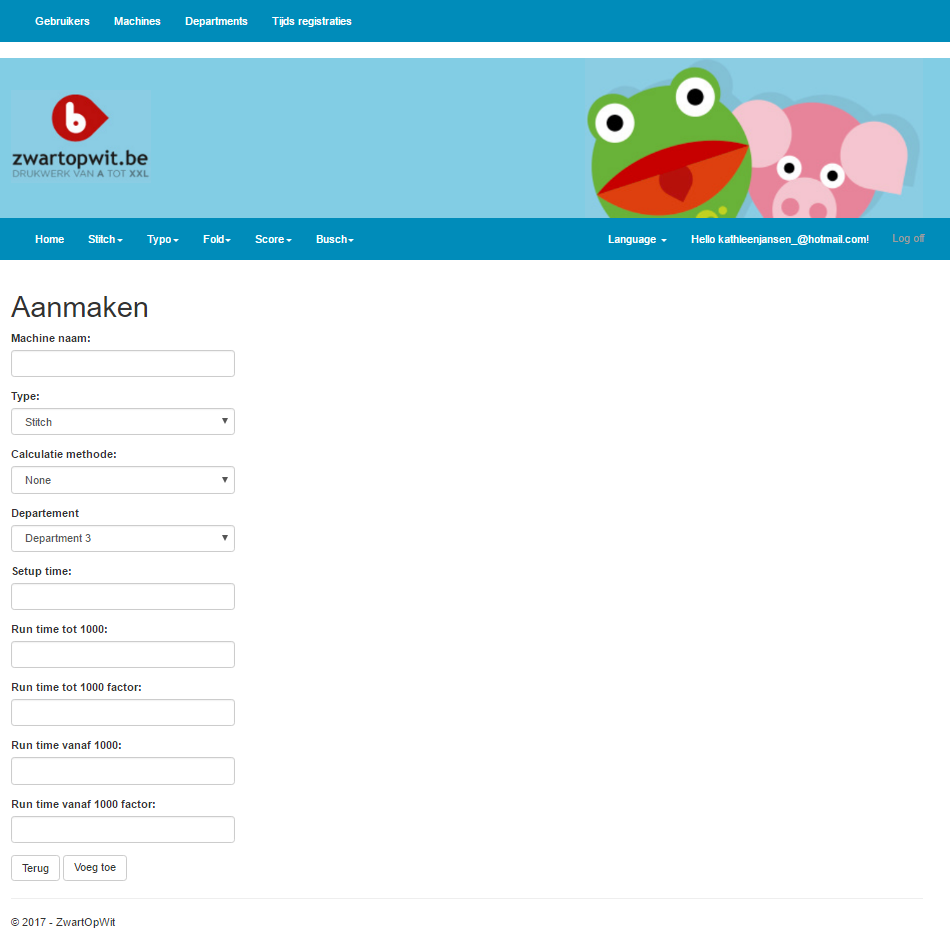
### Machine CRUD

#### Create

Als we op de knop “Nieuwe machine” klikken komen we op de create pagina voor een machine. Hier kunnen we een nieuwe machine aanmaken.



Screenshot: Machines – Nieuwe machine aanmaken



Screenshot: Machines: Nieuwe machine aanmaken detail

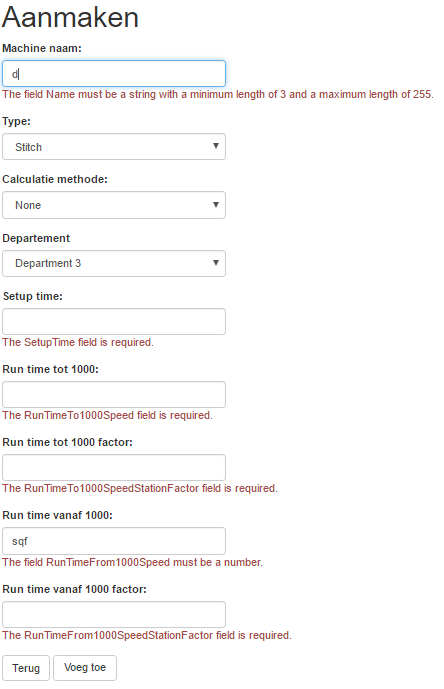
Als we een machine willen aanmaken moeten we de nodige gegevens invullen. De machine krijgt een naam, type, calculatiemethode, departement en de nodige gegevens die de calculatiemethode nodig heeft voor de berekening van de doorlooptijd van de job.

De setup tijd is de tijd die nodig is om de machine in te stellen.

Om de berekening van de doorlooptijd te doen zijn volgende velden nodig:

* De Run time tot 1000
* De run time to 1000 factor
* Run time vanaf 1000
* Run time vanaf 1000 factor

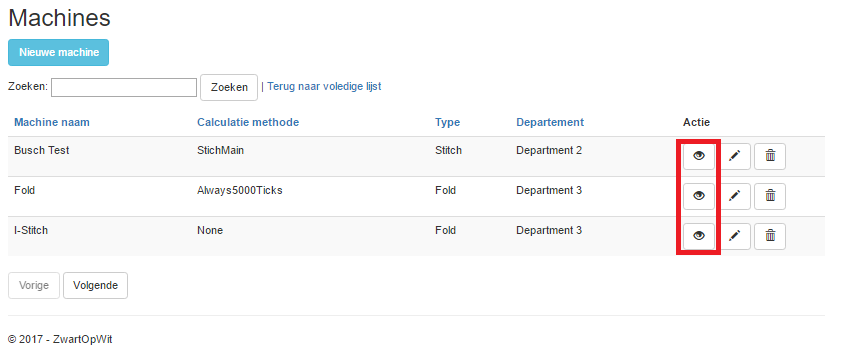
Er zit ook validatie op de velden. Indien deze niet of verkeerd zijn ingevuld geeft deze een foutmelding. Een voorbeeld van een foutieve ingave is bijvoorbeeld een letter gebruiker waar een getal moet komen, of dat de opgegeven naam niet lang genoeg is. Indien alles wel correct is, wordt deze machine met succes aangemaakt. Hij zal nu beschikbaar zijn op de indexpagina en is ook opgeslagen in de database.



Screenshot: Machines - De user krijgt foutmeldingen indien hij een foute ingave doet

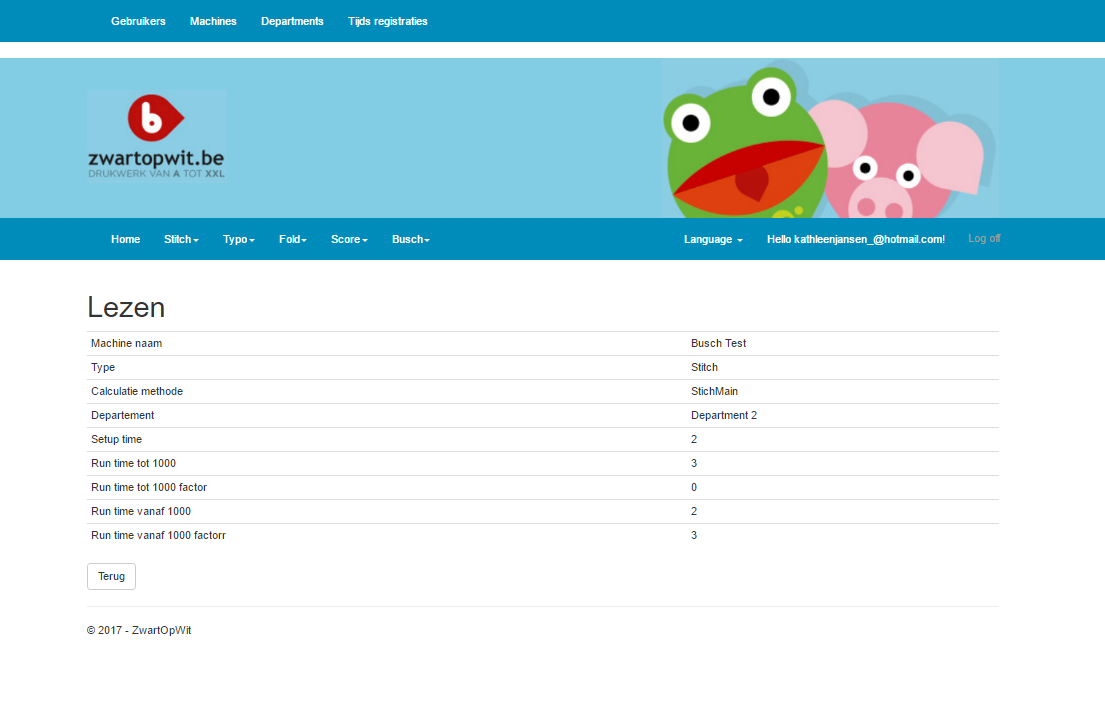
#### Read

Door op het oog – icoontje te klikken in de machine lijst, krijg je een weergave van de details van de machine.



Screenshot: Machines - Druk op het oogje om de machine details te bekijken

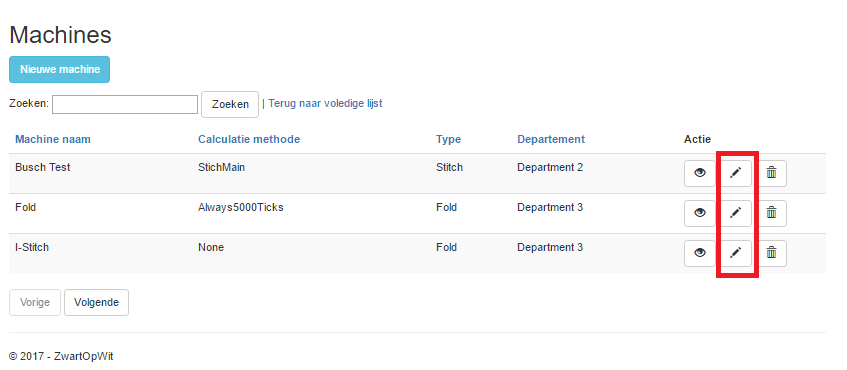
Deze worden weergegeven in een overzichtelijke lijst. Indien je wil terugkeren naar de vorige pagina klik je op “terug”.



Screenshot: Machines – Detail pagina (read)

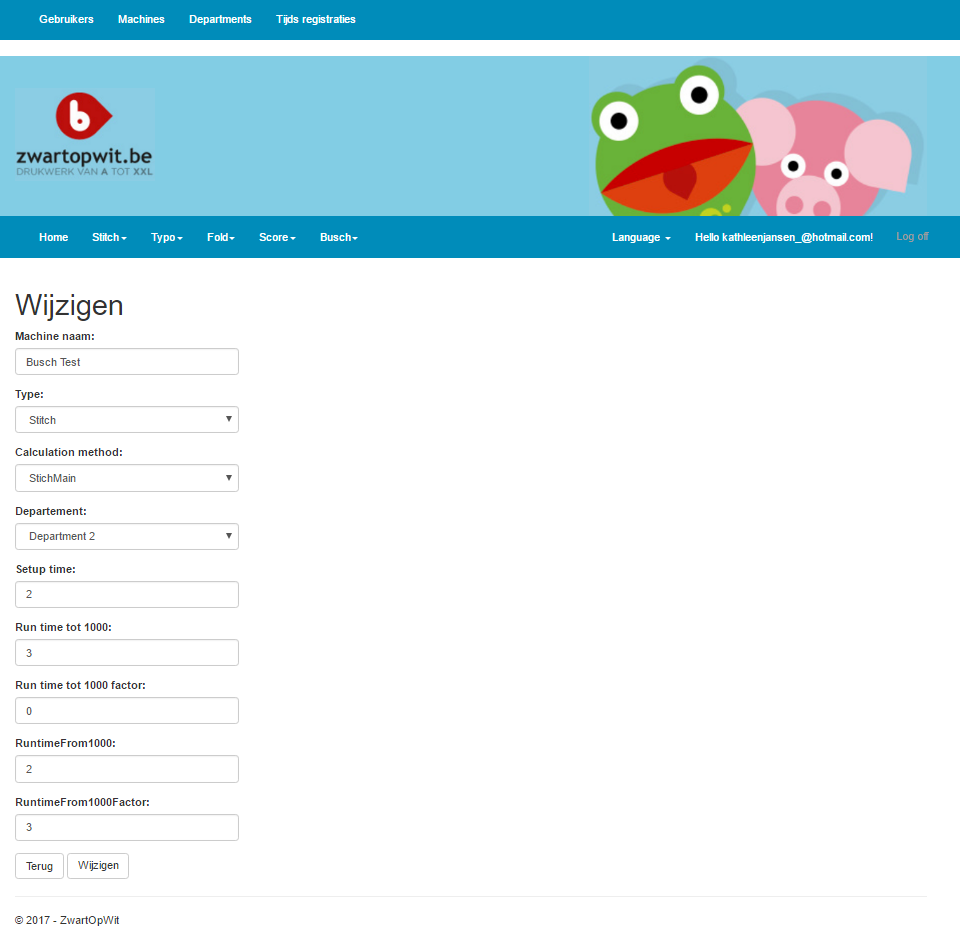
#### Update

Als je op het potlood icoontje klikt kan je de gegevens van de gekozen machine wijzigen.



Screenshot: Machines - Druk op het oogje om de machine te wijzigen

Deze heeft dezelfde invoervelden als de Create/Aanmaken pagina. De reeds bestaande gegevens zijn opgehaald en worden ineens ingevuld. Indien de user klaar is met updaten klikt hij onderaan op wijzigen en komt hij terug op de index pagina.

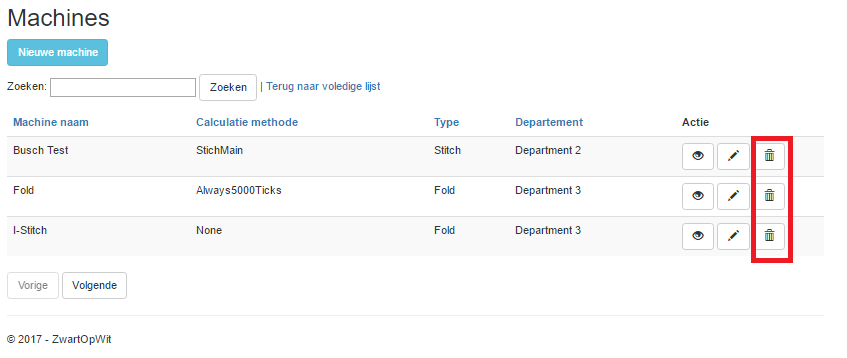


Screenshot: Machines – Wijzigen detail pagina

.

#### Delete

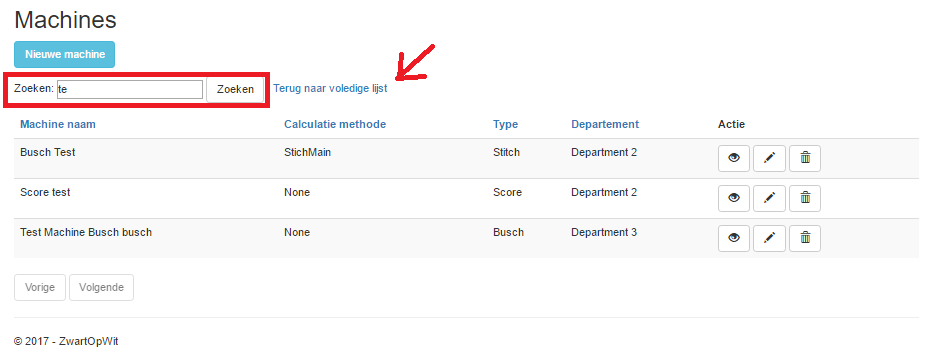
Om een machine te verwijderen druk je op het vuilbak-icoontje op de index pagina. De machine wordt dan verwijderd.



Screenshot: Machines - verwijderen

### Zoeken

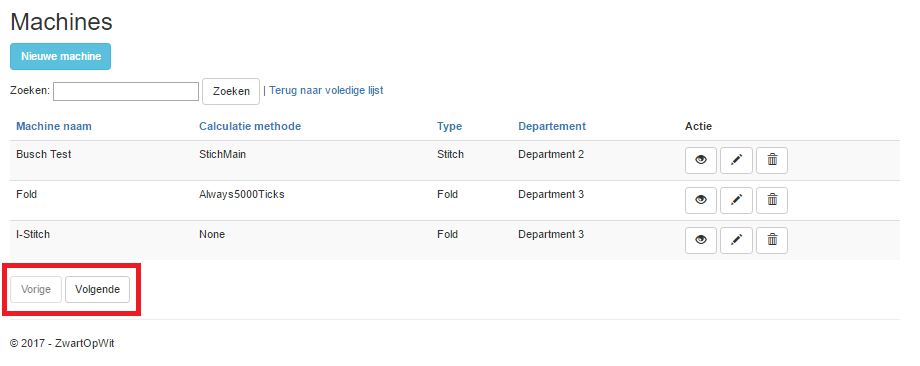
Je kan een machine makkelijk terugvinden door de naam of een gedeelte hiervan in te geven in de zoekbalk. Deze is niet hoofdletter gevoelig. Wil je daarna terugkeren naar de volledige lijst druk je op “Terug naar volledige lijst”.



Screenshot: Machines - zoeken

### Navigeren

Omdat een pagina al snel veel machines kan bevatten hebben we navigatieknoppen voorzien. Met de knoppen “Vorige” en “Volgende” kan je makkelijk naar de volgende pagina. De knop “Vorige” is niet klikbaar op de eerste pagina, de knop “Volgende” is niet klikbaar op de laatste pagina. Het aantal machines dat wordt weergegeven is momenteel ingesteld op 3. Deze kan nog makkelijk aangepast worden in de code.



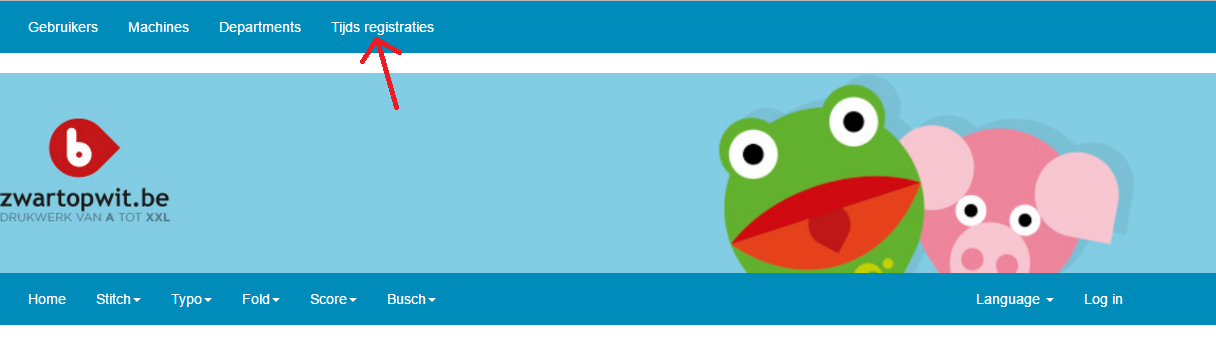
Screenshot: Machines - navigeren door index pagina

## Time registers beheren

### Introductie

Als admin-gebruiker kan je time registers (Tijds registratie) beheren.

De optie time registers beheren is enkel zichtbaar in de admin navigatiebalk. Dus enkel als je een admin-user bent is deze beschikbaar.

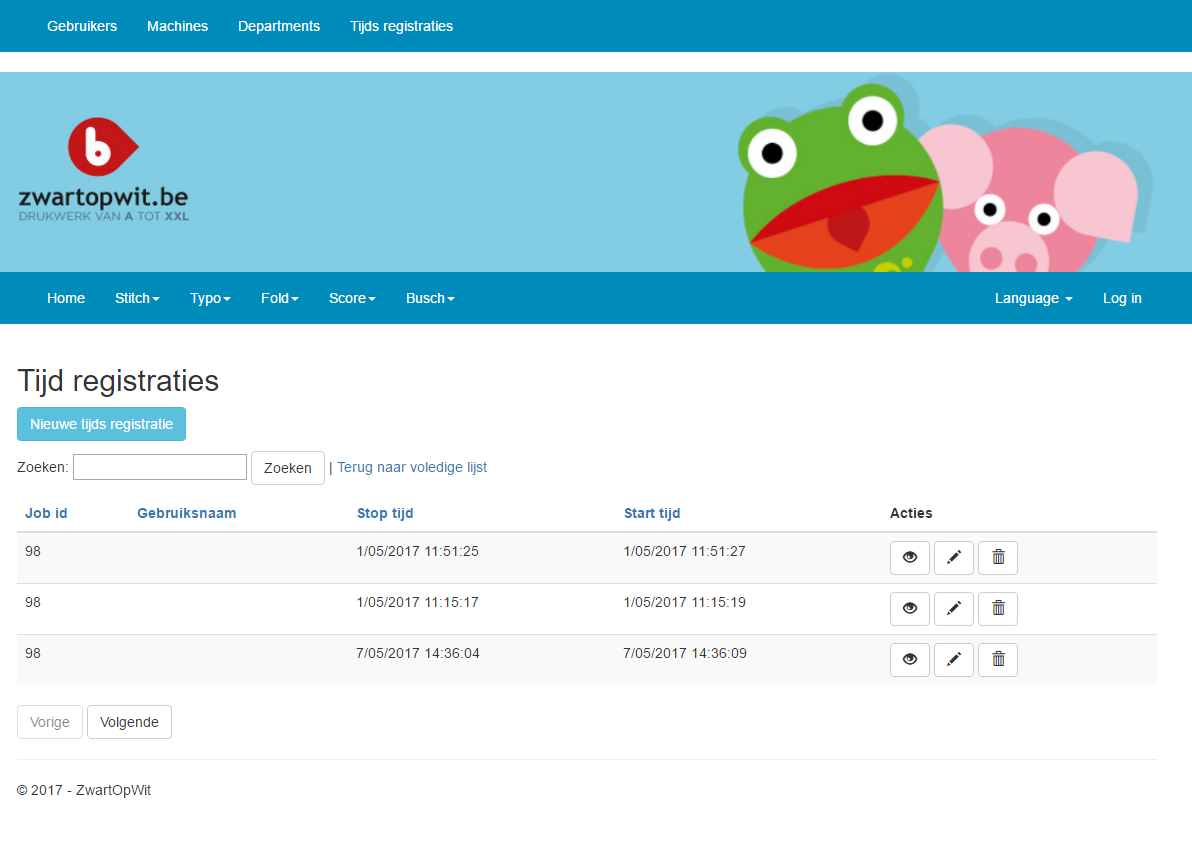


Screenshot: Navigatiebalk voor admin – Ga naar time registers

### Index pagina

Nadat je op de knop “Time registers” in de admin navigatiebalk hebt geklikt, kom je op de index pagina van “Time registers”. Deze geeft een lijst weer van alle huidige departementen.

Hier heb je verschillende mogelijkheden en opties. Voor het beheren van de time registers werken we met CRUD (zie hoofdstuk CRUD). Verder heb je de mogelijkheid om time registers te filteren door gebruik te maken van “Zoeken”. Ook zijn er paginaknoppen voorzien voor als de lijst te uitgebreid wordt.

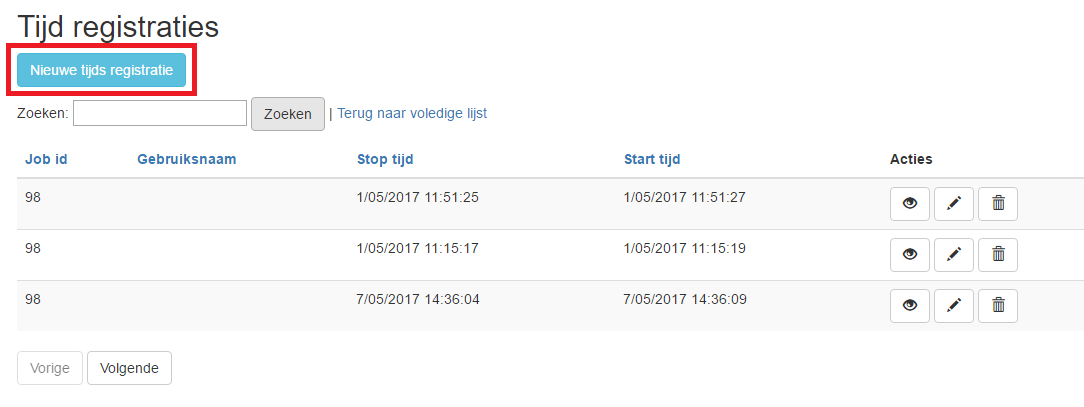


Screenshot: Time registers – Index pagina

### Time register CRUD

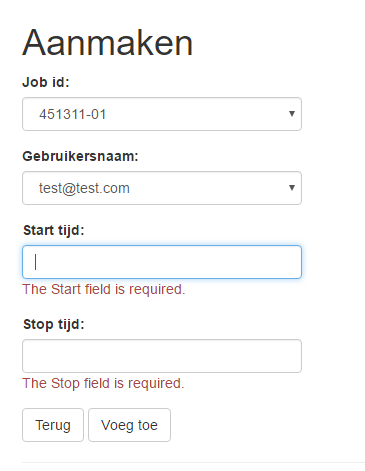
#### Create

Als we op de knop “Nieuwe tijds registratie” klikken komen we op de create pagina voor een tijds registratie. Hier kunnen we een nieuw tijds registratie aanmaken.



Screenshot: Tijdsregistraties – Nieuwe tijds registratie toevoegen

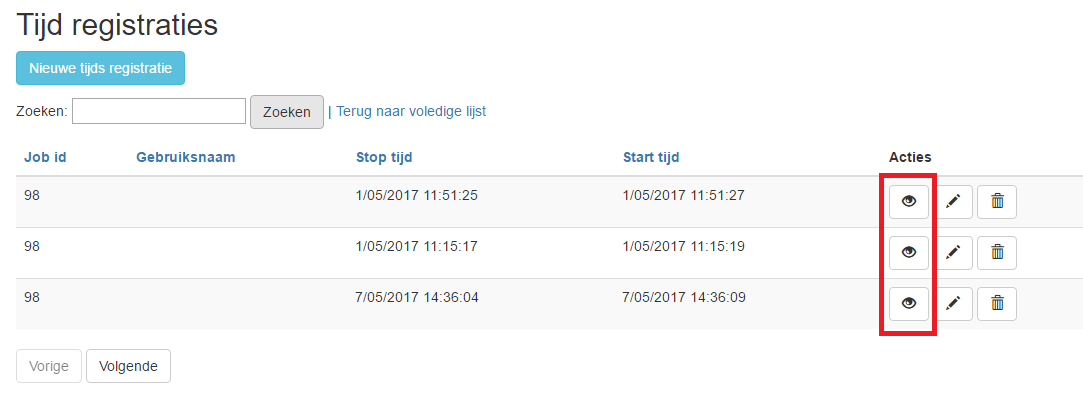
Een tijdsregistratie bestaat uit een link naar een job, user, start en einddatum. Alle velden zijn verplicht behalve einddatum. Deze kan later nog worden meegegeven.



Screenshot: Tijdregistraties – Aanmaken + validatie

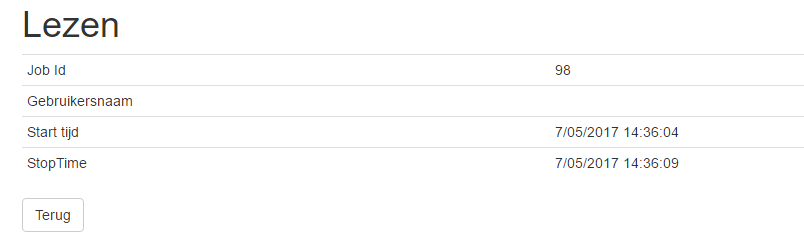
#### Read

Door op het oog – icoontje te klikken in de tijdsregistratie lijst, krijg je een weergave van de details van een tijdsregistratie.



Screenshot: Tijdsregistraties - Read

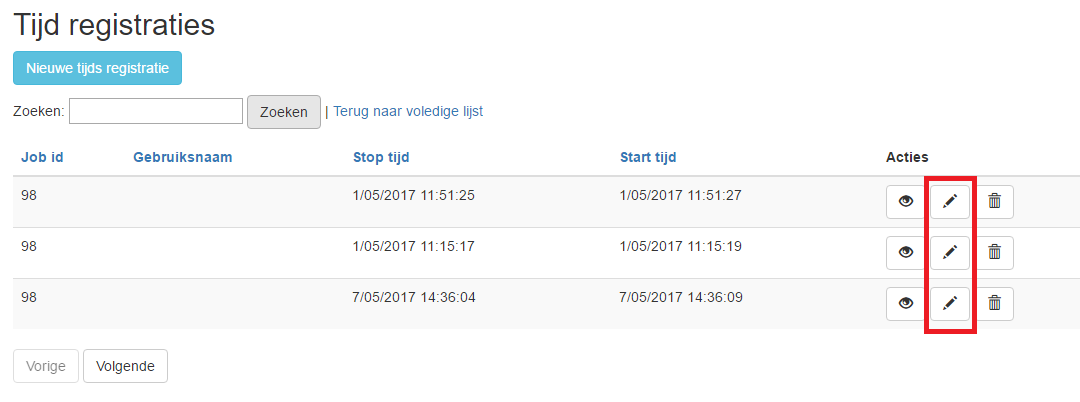
Deze worden weergegeven in een overzichtelijke lijst. Indien je wil terugkeren naar de vorige pagina klik je op “terug”.



Screenshot: Tijdsregistraties – Read detail voorstelling

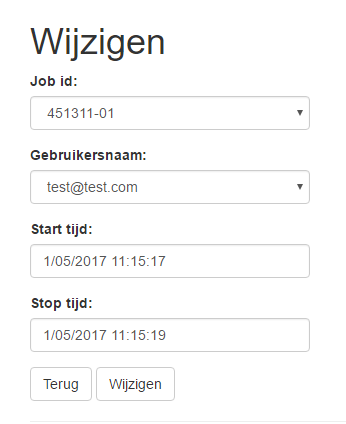
#### Update

Als je op het potlood icoontje klikt, kan je de gegevens van de gekozen tijdsregistratie wijzigen.



Screenshot: tijdsregistraties - Update

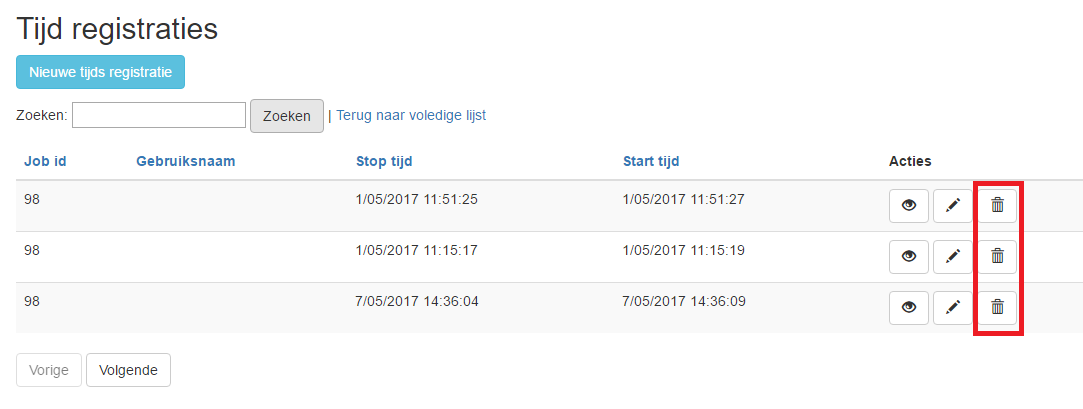
Deze heeft dezelfde invoervelden als de Create/Aanmaken pagina. De reeds bestaande gegevens zijn opgehaald en worden ineens ingevuld. Indien de user klaar is met updaten klikt hij onderaan op wijzigen en komt hij terug op de index pagina.



Screenshot: Tijdsregistaties – Wijzigen detail voorstelling

#### Delete

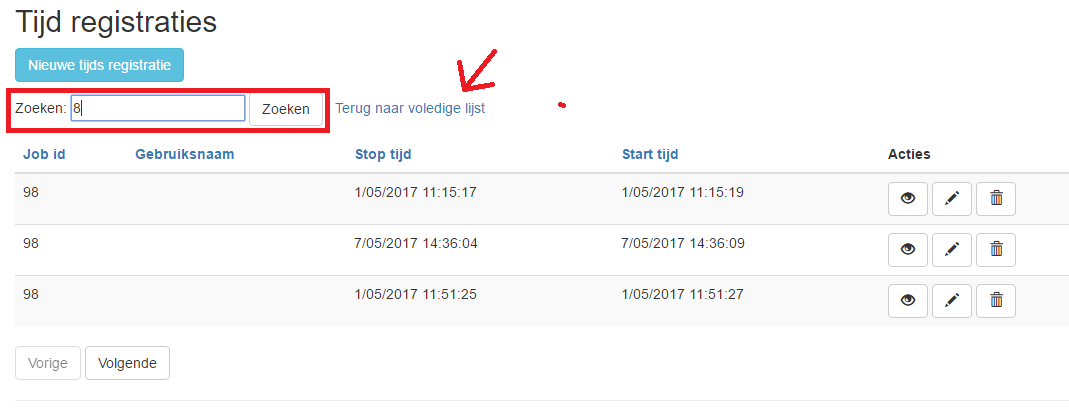
Om een tijdsregistratie te verwijderen druk je op het vuilbak-icoontje op de index pagina. De tijdsregistratie wordt dan verwijderd.



Screenshot: Tijdsregistraties - Verwijderen

### Zoeken

Je kan een tijdregistratie gemakkelijk terugvinden door de naam of een gedeelte hiervan in te geven in de zoekbalk. Deze is niet hoofdletter gevoelig. Wil je daarna terugkeren naar de volledige lijst druk je op “Terug naar volledige lijst”.



Screenshot: Tijdsregistraties - zoeken

### Navigeren

Omdat de mogelijkheid bestaat dat de pagina veel tijdsregistraties bevat, hebben we navigatieknoppen voorzien. Met de knoppen “Vorige” en “Volgende” kan je gemakkelijk naar de volgende pagina. De knop “Vorige” is niet klikbaar op de eerste pagina, de knop “Volgende” is niet klikbaar op de laatste pagina. Het aantal tijdregistraties dat wordt weergegeven is momenteel ingesteld op 3. Deze kan nog gemakkelijk aangepast worden in de code.



Screenshot: Tijdsregistraties – Navigeren door index pagina

## Job planning

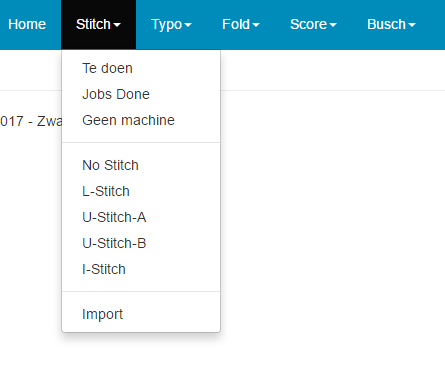
### Jobs menu items

Er zijn verschillende pagina’s voorzien om job lijnen weer te geven. Deze pagina’s worden dynamisch opgebouwd vanuit de controller. Het menu bestaat uit de 5 machine types:

* Stitch
* Type
* Fold
* Score
* Busch

Voor ieder machine type wordt dan de volgende menu structuur gehandhaafd

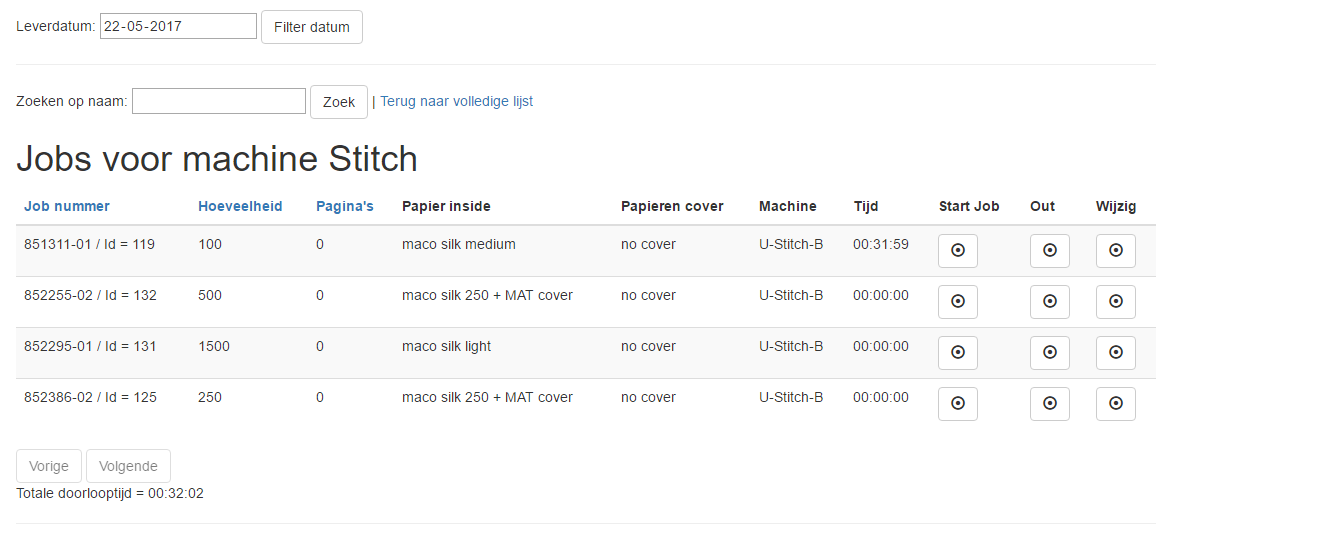
* To do
* Jobs Done
* No machine
  + Machine 1
  + Machine 2
  + …..
* Import



Menu voorbeeld stitch

#### To do

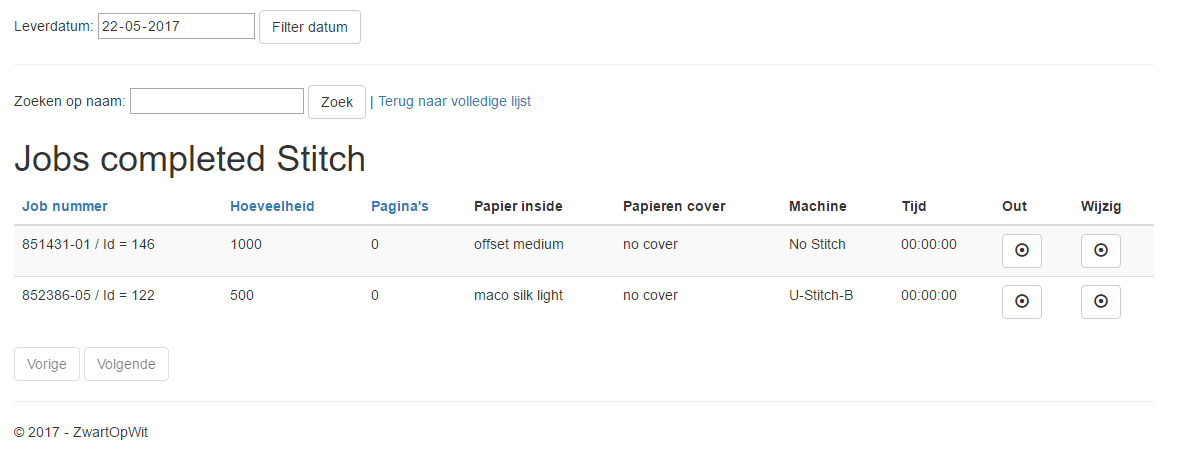
Deze pagina gaat voor het machine type dat geselecteerd is alle ingeplande jobs tonen.



Jobs to do overzicht

#### Jobs done

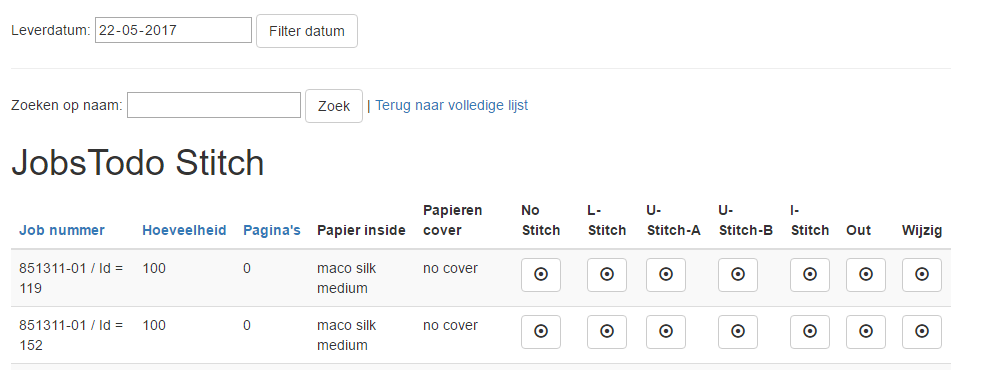
Deze pagina toont alle jobs voor het machinetype waar het menu onder valt die al in de status “completed” zijn.



Jobs completed overzicht

#### No machine

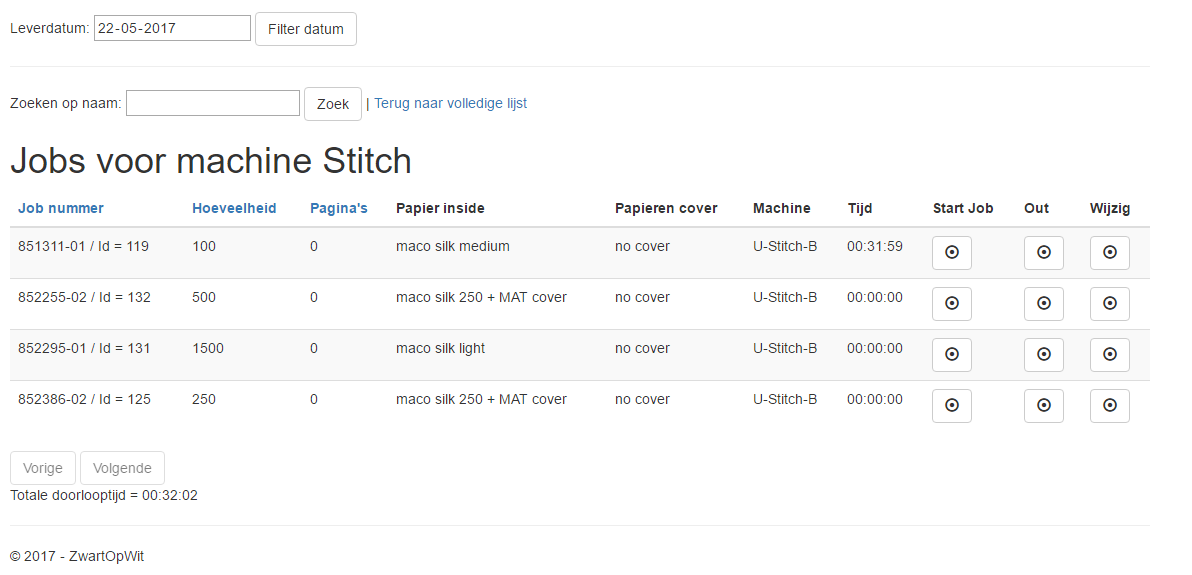
Deze pagina toont alle jobs die zijn opgeladen voor het machine type onder het gekozen menu, waarvan die nog niet zijn toegewezen aan een machine.



Jobs zonder machine

#### Machine detail

Dit zijn de detail pagina’s per machine en deze laten alle jobs zien die gereed staan voor een bepaalde machine. Vanuit deze pagina kan ook een job gestart of gestopt worden.



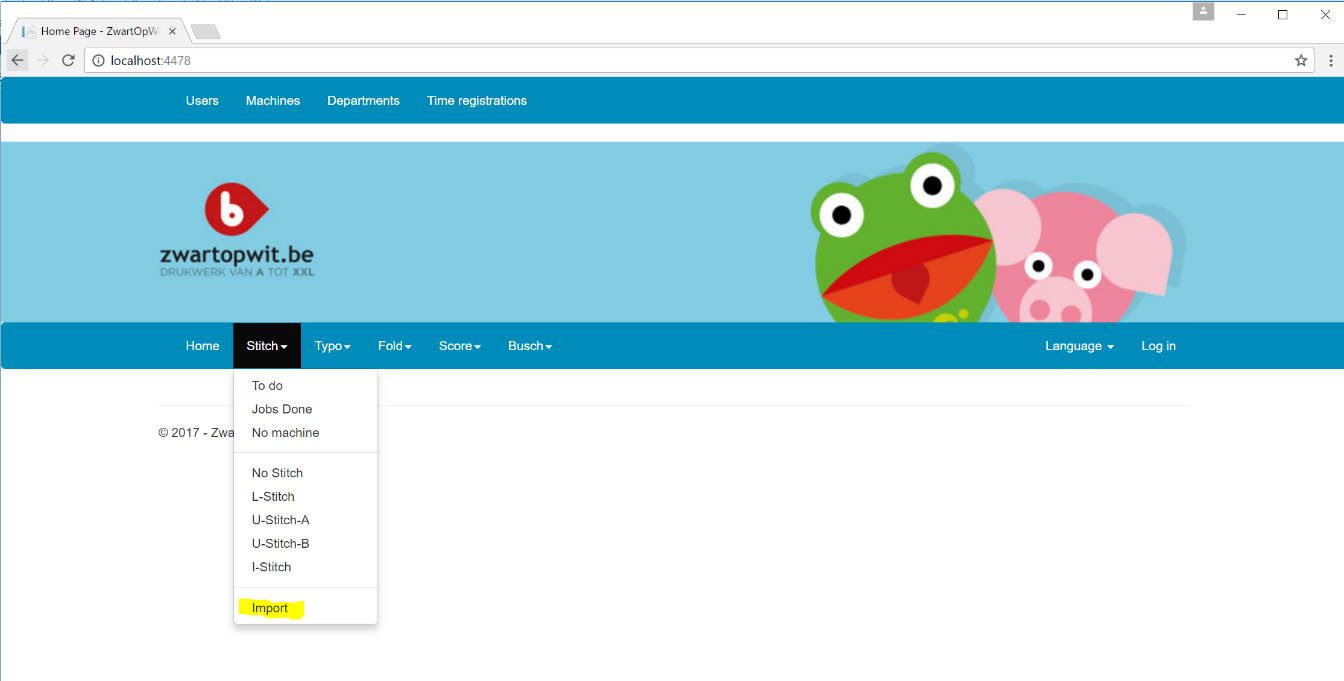
Machine detail overzicht

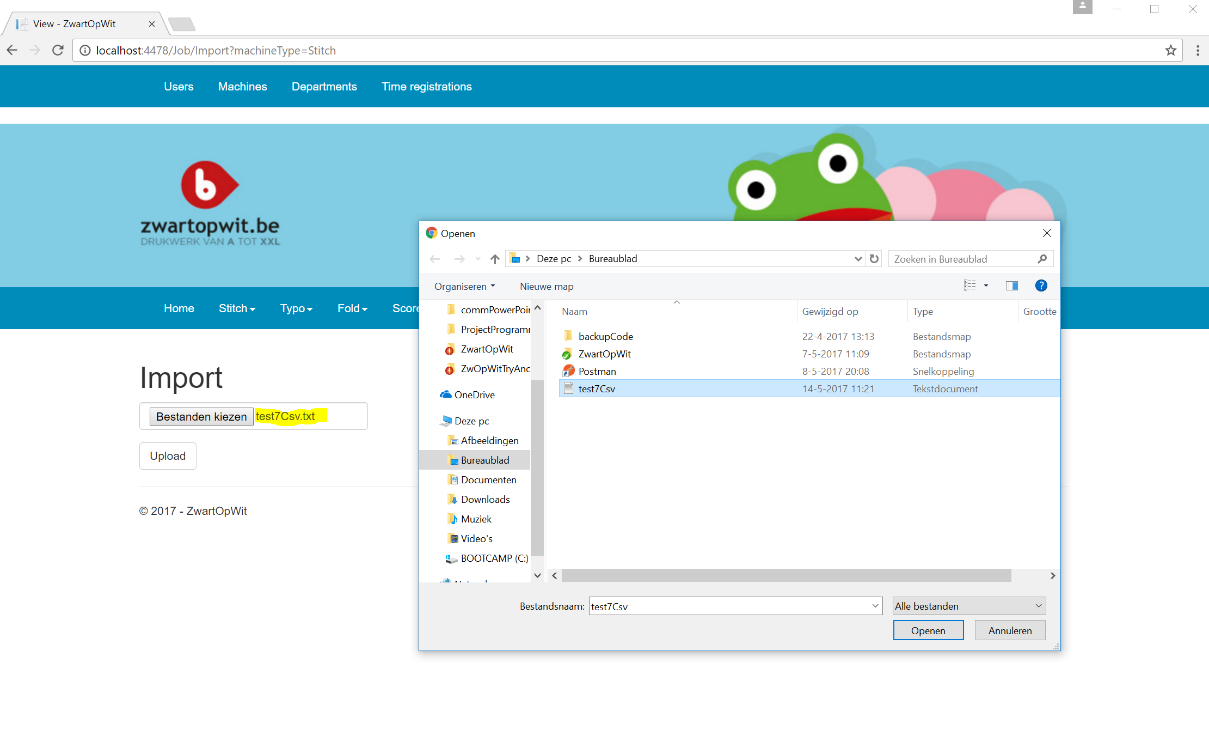
#### Import

Via het programma Filemaker kan een csv-file gemaakt worden met de gegevens van de jobs. Deze plaats je best op het bureaublad. Via Import kan je het bestand opzoeken en zo halen we de jobs binnen en kopiëren we de gegevens naar de Database. De scheiding tussen jobs en joblines gebeurt hier.

Per machinetype wordt er een andere file aangeleverd. Het machine type wordt automatisch ingevuld afhankelijk van het menu waaruit de import gestart wordt.

Voor een latere uitbreiding wordt hier al de opsplitsing gemaakt tussen jobs en joblines. Een job kan bijvoorbeeld een “stitch” deel en een “fold” deel omvatten. Deze acties behoren tot dezelfde job daarom de opsplitsing.

Selecteer het juiste bestand en druk op Upload.



### Jobs starten en stoppen

Vanuit het machine detail scherm en het to do scherm is het mogelijk om jobs te starten en te stoppen. Als er een job gestart word wordt er achterliggend een tijdsregistratie record aangemaakt waarop de begindatum ingevuld is.

Als de medewerker voor stoppen kiest gaat de medewerker moeten kiezen of de job gedaan is of niet. Er zijn verschillende scenario’s waardoor 1 persoon de job niet kan afmaken.

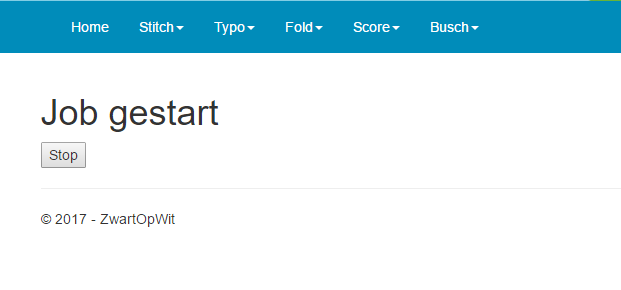
### Job flow

De job starten gebeurt door te klikken op start job.



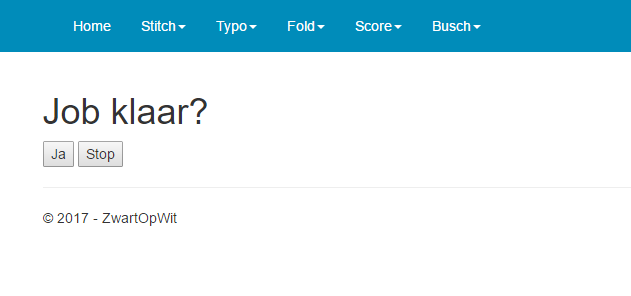
Start job vanuit jobs to do

Als de job gestart is komt de medewerker op het scherm uit om de job te stoppen



Stop job

Als er op stop wordt geklikt krijgt de medewerker de vraag of de job klaar is indien ja wordt de job lijn op completed geplaatst anders blijft deze open staan. Het stoppen zorgt ervoor dat de tijdsregistratie een einddatum krijgt.



Stop job opties

# Algemeen Besluit

Als algemeen besluit willen we graag concluderen dat we heel wat hebben bijgeleerd. Enerzijds zijn we toch wel trots op het product dat we hebben gerealiseerd na heel wat zwoegen en zweten.

We hebben geleerd dat een goede analyse belangrijk is en dat het op termijn toch wel heel tijdbesparend zou geweest zijn. Iets wat we jammer genoeg iets te laat bij achter kwamen. Dit is iets wat we zeker willen meenemen naar een toekomstig project.

Ook communicatie is belangrijk. Met twee aan één project werken is niet altijd even simpel. Je moet samen conclusies maken en zien dat je op dezelfde golflengte zit. We zijn hier als een sterk team aan de slag gegaan, dat elkaar ondersteunde en aanvulde waar nodig.

Het uitwerken van een dergelijk project vraagt organisatie en inspanning. We kwamen tot de conclusie dat het eigenlijk nooit af is, en dat je je verder in kan blijven verdiepen.

Algemeen kunnen we besluiten dat we fier zijn op onze planningstool. Deze vergt nog wel wat werk en inzet, maar geeft een mooie basis om in de toekomst verder uit te breiden. Dit rekening houdend met zijn specifieke noden en wensen.

# Bijlages

1. Plan van aanpak

18-11-2016

Kathleen Jansen & Jo Brunau

# Plan van Aanpak

Tool voor planning en afwerking drukwerk

www.zwartopwit.be

Inleiding

Dit plan van aanpak is opgemaakt in het kader van de implementatie van een nieuwe tool voor planning van afwerking van drukwerk bij zwartopwit.be / drukkerij Bulckens.

Probleemstelling

Bestellingen van drukwerk komen binnen via de website. Nadat deze bestellingen zijn ontvangen worden zij verwerkt. De vertaling van het bestelde pakket drukwerk naar de werkvloer toe, is echter in de huidige omgeving erg complex georganiseerd.

Op een vlottere en lean-gerichte manier te werk kunnen gaan, is er nood aan een programma dat de planning en verwerking van de verschillende drukwerken doorheen de afwerking van de drukkerij op elk moment kan volgen.

Business and Functional Requirements

* Systeem moet op basis van externe Filemaker database gegevens importeren in een eigen database.
* Gegevens in de database moeten kunnen bijgewerkt worden.
* Systeem moet a.d.h.v. de gegevens in de database geschatte werktijden kunnen berekenen.
* De applicatie moet beschikken over een beveiligd identificatieproces.
* Er moet een onderscheid gemaakt kunnen worden tussen medewerker en verantwoordelijke.
* Nieuwe gebruikers moeten kunnen worden aangemaakt.
* Gebruikers moeten planplaats kunnen kiezen.
* Systeem moet drukwerken aan de juiste planplaats toewijzen.
* Gebruikers moeten af te werken drukwerk kunnen kiezen volgens levertijd en bestemming.
* Systeem moet werktijden registreren.
* Systeem moet onderbrekingen registreren.
* Gebruikers moeten opmerkingen kunnen toevoegen aan elke tijdsregistratie.
* Systeem moet alle verzamelde gegevens naar de database schrijven en linken aan de gekozen drukwerken
* Systeem moet in realtime kunnen tonen welke gebruikers, welk drukwerken hebben opgestart.
* Systeem moet queries kunnen uitvoeren op de database om nacalculaties uit te voeren volgens werknemer en drukwerk

Design Requirements

* De applicatie zal in eerste instantie standalone draaien. Later kan nog bekeken worden of verweven in de Filemaker database tot de mogelijkheid behoort.
* De applicatie dient geschreven te worden in .NET
* Het resultaat is een web-applicatie.
* De database wordt een MySQL database.

Technologie

Doordat er in het huidige IT-landschap al gebruik wordt gemaakt van Linux based services (Ruby) en er een MySQL database beschikbaar is de keuze gevallen om technologieën in deze lijn te gebruiken. De data die door de planning tool gebruikt kan teruggevonden worden in Filemaker.

* Asp.net Core c#
* MySQL
* Filemaker
* Razor

Kostprijs

Ontwikkelingskosten:

* Deze planningstool wordt uitgewerkt a.d.h.v. een schoolproject waardoor deze 0 euro bedraagt.

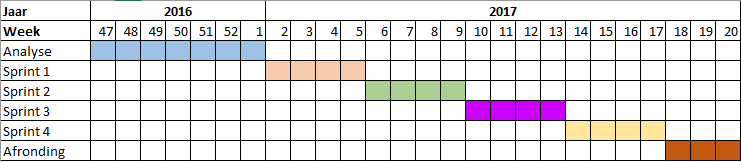
Onderhoudskosten

* Om de planningstool te laten draaien worden de kosten geschat op 0 euro. Dit omdat we draaien op de bestaande infrastructuur gaan er geen extra licentie en hardware kosten zijn.
* Op termijn kan er een investering nodig zijn in extra hardware naarmate de applicatie groeit.

Planning

Er is gekozen om Agile te gaan werken. Dit omwille van volgende redenen:

* Flexibel opleveren
* Makkelijk omgaan met scope uitbreiding
* Alles wordt in hapklare delen opgeleverd voor einduser



Verwachte sprintplanning:

Analyse:

* Het analyseren van de planningtool + uitwerking userstories
* Uitwerken van proof of concept (testen of het systeem goed kan communiceren met andere systemen in het IT-landschap)

Sprint 1:

* Uitwerken van basis van de user-interface
* Ophalen van data uit filemaker
* Uitwerken van form-templates
* Authenticatie en autorisatie (login)

Sprint 2:

* Uitwerken van de planningtoolschermen
* Uitwerken van planning engine

Sprint 3:

* Te bepalen na analyse

Sprint 4:

* Te bepalen na analyse

Afronding:

* Bugfixing
* Detail afhandeling

UAT

User Acceptance Testing. Voor alles live gaat in productie moeten de key users uitgebreid de tool testen en goedkeuring geven. Alle testen worden beschreven in een te bepalen testing tool. Na een testcycli wordt er een document met de resultaten verdeeld.

Op te leveren:

* End-to-end test-scenario’s
* Go-live approval

Training

Training bestaat uit het verzamelen/produceren van trainingsmateriaal en het geven van de effectieve training aan users. Enkel key users worden getrained. Deze worden verwacht hun kennis over te dragen naar de eindgebruikers (Train the trainer).

Op te leveren:

* Trainingsmateriaal
* Getrainde key users

# Bronnen

WARMER, J. en KLEPPE, A., Praktisch UML 4e Editie, 2e druk, Pearson Education Benelux, Amsterdam, 2008, 253 pagina’s

Microsoft (2017), MVC ASP.NET, [www.asp.net/mvc](http://www.asp.net/mvc)

W3schools (2017), ASP.NET Razor, <https://www.w3schools.com/asp/razor_intro.asp>

Damienbod (2016), ASP.NET Core with MySQL and Entity Framework Core, <https://damienbod.com/2016/08/26/asp-net-core-1-0-with-mysql-and-entity-framework-core/>

Microsoft Docs (2017), View Components, <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/view-components>

Code Project/Rahul Rajat Singh, R.R.S. (2012), An Introduction to Entity Framework, <https://www.codeproject.com/Articles/363040/An-Introduction-to-Entity-Framework-for-Absolute-B>

Microsoft (2017), Introduction to Entity Framework, <https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx>

Microsoft (2017), Introduction to Identity, <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity>

VAN DE PERRE L., Source Control met Git (2016), 16 slides

Options pattern <https://andrewlock.net/how-to-use-the-ioptions-pattern-for-configuration-in-asp-net-core-rc2/>

Microsoft Docs (2017), Globalisation and Localization, <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/localization>

GitHub (2017), Github, <https://github.com>

Bootstrap (2017), Bootstrap, <http://getbootstrap.com/>

.net core policies (2017), Turotorial: Policy-based Authorization in ASP.NET Core, <https://stormpath.com/blog/tutorial-policy-based-authorization-asp-net-core>

Roles setup (2017), Roles Setup <http://stackoverflow.com/questions/39934201/asp-net-core-identity-add-custom-user-roles-on-application-startup>

Option pattern (2017), Configurations in .net Core, <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration#using-options-and-configuration-objects>