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

# Een Customer Service Portaal in ASP.NET Core 2.0 met Entity Framework

Aertgeerts Nathan

academiejaar 2017-2018

AP Hogeschool Antwerpen
Wetenschap &Techniek
Elektronica-ICT

ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

# Table of Contents

# Abstract

Een customer service portaal in ASP.NET Core 2.0 met Entity Framework Core. Een bachelorproef geschreven door Nathan Aertgeerts, student Elektronica-ICT aan Artesis Plantijn Hogeschool te Antwerpen. Door middel van voorgaande technologie stack creëren we een portaal waarop de klant zichzelf kan helpen. Het voorgaande onderzoek was bepalend voor de noodzakelijke features op te lijsten in de analyse. Vervolgens zijn de features agile en sprintmatig aangepakt.

Het ticketingsysteem met email integratie staat de klant toe om eenvoudig via het webportaal of mail een ticket aan te maken. Support medewerkers kunnen vervolgens op deze tickets antwoorden en klanten zijn op de hoogte van wijzigingen met behulp van een notificatie. Support medewerkers kunnen dankzij de Visual Studio Team Services integratie een Bug of Feature aanmaken, koppelen, bekijken of updaten naargelang het ticket. De integratie van Exact Online zorgt ervoor dat support medewerkers alle bedrijfsgegevens bij de hand hebben wanneer ze een ticket behandelen. Aan de hand van het contractlevel kunnen we bedrijven en klanten gepersonaliseerde support mogelijkheden aanbieden. Tenslotte voorzien we een knowledgebase en document management systeem waarbij geregistreerde klanten toegang hebben tot verschillende documenten en downloads. We willen de gebruikers informeren van nieuwigheden en veranderingen door notificaties te tonen wanneer nodig.

De gecombineerde features resulteren in een uniek customer service portaal dat zowel klant als support medewerker een overzichtelijke en uitgebreide oplossing biedt.

# Dankwoord

Graag wil ik mij met het schrijven van dit dankwoord richten tot iedereen dat heeft bijgedragen aan de realisatie van deze bachelorproef.

Ik wil graag Intation bedanken voor de fijne samenwerking. In het bijzonder mijn stagebegeleider Baetens Cedric voor de professionele begeleiding en technische ondersteuning om dit project tot een succesvol einde te brengen.
Van Hoorenbeke Nico en Roks Daan voor de fijne samenwerking en de kans die ik bij Intation heb gekregen om mijn onderzoek uit te voeren en mijn scriptie te schrijven.

Daarnaast wil ik graag mijn stagementor Smets Marc bedanken voor de begeleiding en feedback.

*Antwerpen, Juni 2018*
*Aertgeerts Nathan*

# 1. Introductie

## 1.1 Het bedrijf: Intation

Intation is een bedrijf dat zich specialiseert in de industriële automatisatie. De grootste werkvorm bestaat uit consultancy en wordt uitgebreid met inhouse software development. Intation heeft een assortiment van producten die beschikbaar zijn via licenties.

## 1.2 Situering

In de internetstructuur van Intation is er vandaag nog geen bestaand service portaal.

De huidige website wordt geoutsourcet, draait op wordpress en bevat een klantenzone. Dit is een eenvoudig contact formulier waarbij bestaande klanten vragen kunnen stellen over een van de bestaande producten. Wanneer het formulier is ingevuld komt dit in de Intation-mailbox terecht, daarna wordt dit verder behandeld vanuit de mailbox.

Klanten kunnen niet terecht op de website voor FAQ te bekijken of om zichzelf te helpen. Een klant zal niet goed op de hoogte zijn van zijn support-request en wanneer het klantenbestand zal uitbreiden wordt het een intensieve taak om dit in een mailbox te onderhouden. Bepaalde zaken zullen verloren gaan en dit heeft tot gevolg dat het supportteam het overzicht verliest en de klant ontevreden achterblijft.

Het klantenbestand wordt vandaag beheerd in exact online, een uitgebreide business software - CRM tool die enkele REST API's ter beschikking stelt. Deze tool wordt onder andere ook gebruikt voor de boekhouding en financiële gegevens.

## 1.3 De opdracht

Een Customer Service Portaal opstellen voor Intation waarop de klant kan registreren en inloggen. Daarna kan de klant support-tickets creëren en reeds aangemaakte tickets bekijken. Deze tickets worden op hetzelfde portaal door het supportteam behandeld. De ontwikkelaars kunnen met de visual studio team services integratie eenvoudig een bug of work-item aanmaken om hun software development te sturen en te voldoen aan de behoefte van de klant. Het support-ticket systeem moet een geïntegreerde email functionaliteit bevatten en een knowledgebase systeem, zodat klanten zichzelf kunnen helpen en op de hoogte worden gehouden van belangrijke updates.

Deze structuur moet ervoor zorgen dat klanten efficiënt en snel geholpen kunnen worden. Waardoor het supportteam zijn tijd optimaal kan benuttigen. Het doel is de noodzakelijke features te implementeren en te streven naar een tool die kan ingezet worden naarmate het klantenbestand van Intation groeit, waarbij zowel de klant, de werknemer en de werkgever voordelen van zullen ondervinden.

# Structuur scriptie

De scriptie is opgedeeld in 5 hoofdstukken. Volgende titels en paragrafen beschrijven de inhoud van de hoofdstukken.

## Introductie

Dit deel vormt de algemene inleiding van de scriptie en ontvermt zich over de opgave en dus de inhoud en situering van de bachelorproef.

## Hoofdstuk 2: Technologieën

Dit hoofdstuk beschrijft de gebruikte technologieën en tools die aanbod zullen komen. De beschreven tools zijn hulpprogramma's die essentieel waren om het customer service portaal te ontwikkelen. De technologieën zijn frameworks en andere waarop de basis van het customer service portaal is ontwikkeld

## Hoofdstuk 3: Onderzoek

Dit deel toont ons het voorafgaande onderzoek om het eindresultaat te bereiken en het design op te stellen. Het onderzoek heeft enkele tijd in beslag genomen en is noodzakelijk geweest voor het bepalen van de eigenschappen van het customer service portaal. Het design toont de kern van het customer service portaal.

## Hoofdstuk 4: Resultaten

De resultaten zijn een uitgebreide beschrijving van elk functioneel blok in het project. We beschrijven wat, waarom en hoe deze functionaliteit geïmplementeerd is. Dit is aangevuld met de problemen die zijn ondervonden en de resultaten die behaald zijn. Dit hoofdstuk behandelt de volgende onderwerpen (in volgorde van voorkomen):

- Basis
- Ticket Systeem
- Knowledgebase
- Document Management Systeem
- Notificaties
- Email integratie
- Visual Studio Team Services integratie
- Exact Online integratie

## Hoofdstuk 5: Conclusie

Dit hoofdstuk reflecteert over de resultaten en of deze voldoen aan de verwachtingen. Er worden aanbevelingen gemaakt voor toekomstig werk of uitbreidingen.

# Tools

## Visual Studio Team Services

Om features te kunnen plannen en verder uit te werken maakt Intation gebruik van een project management software, "Visual Studio Team Services". Deze software wordt aangeboden door Microsoft en maakt gebruik van een "Git" systeem en "Agile" methodes om software development overzichtelijk te kunnen plannen en beheren.

## Visual Studio

Om code te schrijven gebruiken we "visual studio (2017)", dit is een geïntegreerde ontwikkelingsomgeving van Microsoft. Visual Studio geeft ons de mogelijkheid om met behulp van ontwikkelingtools in verschillende programmeertalen computerprogramma's te ontwikkelen of code te schrijven. Visual Studio en Visual Studio Team Services kunnen eenvoudig gekoppeld worden. Dankzij het ingebouwde versie controle systeem van VSTS hebben we altijd een back up van onze broncode bij de hand.

## IIS

Internet Informatie Services is ontwikkeld door Microsoft en maakt het mogelijk een standaard computer om te vormen in een webserver.

## Visual Studio Code

Visual Studio Code is een broncode-editor ontwikkeld door Microsoft

# Technologieën

## ASP.NET Core 2.0

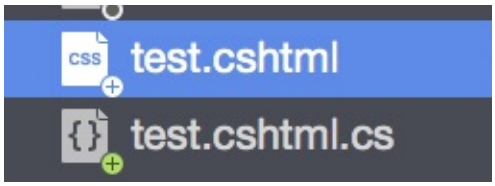
ASP.NET Core 2.0 is de meest recente versie van ASP.NET (Active Server Pages). ASP.NET is een open-source web-framework ontwikkeld door Microsoft en de community dat gebruikt wordt om moderne cross-platform en cloud-based web applicaties te bouwen.

## Entity Framework Core 2.0

Entity Framework is een open-source object relational mapper voor .NET applicaties ontwikkeld door Microsoft. Ontwikkelaars kunnen eenvoudiger omgaan met data doordat ze zich niet meer hoeven te focussen op de onderliggende database tabellen waarin de data wordt opgeslagen. Dit zorgt ervoor dat de code korter en overzichtelijker wordt. Entity Framework Core 2.0 is de meest recente versie van het framework, de belangrijkste wijziging van het Core segment is de cross-platform mogelijkheid. Entity Framework geeft ons de mogelijkheid om zowel code-first als database-first te werken. In dit project werken we code-first zodat we ons geen zorgen hoeven te maken over de database aangezien onze code en Entity Framework dit afhandelen.


## Razor Pages

Razor pages is een web-framework om eenvoudig webpagina's aan te maken. De razor pages maken two-way data binding eenvoudiger omdat de achterliggende code van controller en model samengevoegd worden. We verkrijgen een beter georganiseerde structuur ten opzichte van andere web-frameworks. Een razor page bestaat slechts uit twee bestanden, een bestand met de html code en een bestand met de C# code.




## Bootstrap 4

Bootstrap is een open-source framework bestaande uit HTML, CSS en Javascript om webpagina's te ontwikkelen en ontwerpen. Met behulp van deze stijlsjablonen kan je een consistentie leveren in 'look en feel' van je website die zich aan alle browserformaten zal aanpassen.

# Analyse

## Onderwerp

Het onderwerp van mijn bachelorproef had een vrij ruime betekenis aan de start van de stage. De opdracht was het voorzien van een customer service portaal, een webpagina waar de klant zichzelf kan helpen. Uit ervaring van projectopdrachten vroeg dit om een uitgebreid onderzoek. De eerste twee weken werden ingepland om een grondige analyse voor te bereiden.

Om mijn onderzoek te starten ben ik na advies van Daan Roks, de productowner, opzoek gegaan naar reeds bestaande oplossingen of 'tools' op de markt. De bestaande helpdesk tools bieden zeer veel features aan en zijn in grote lijnen met elkaar te vergelijken. De grootste spelers op de markt zijn aanhangsels van andere CRM of teamsupport aanbieders zoals Salesforce of Atlassian. Waardoor een ideale implementatie van deze software vaak steunt op deze aanbieder. Aangezien Intation gebruikmaakt van Office 365 en Visual Studio Team Services werd een eerste vereisten vastgesteld. *Het customer service portaal moet aansluiten bij de bestaande infrastructuur.*

## Wat is de huidige infrastructuur?

De website van Intation draait op wordpress, deze is extern opgezet en werd tot voor kort volledig uitbesteed. Wordpress is een Content Management Systeem (CMS) waarmee je zeer eenvoudig een website kan opzetten en onderhouden. Intation heeft intern een voorkeur voor C# en dat is duidelijk waarneembaar aan de huidige producten en projecten binnen Intation. Het meest recente product is in C# geschreven op een basis van ASP.NET Core 2.0 met het Entity Framework Core. Deze termen komen aan bod in het hoofdstuk "Technologieën".

## Bestaande oplossingen en features

Features zijn eigenschappen die meestal een positieve toevoeging hebben aan een bepaald product of software. Door features van grote spelers in de markt op te lijsten kunnen we nadien filteren op noodzaak van Intation. De grootste spelers op de markt van customer-support profiteren van hun aandeel en zijn zeer prijzig. Daarom heb ik mijn zoekopdracht wederom verfijnt en ben ik specifiek opzoek gegaan naar open-source mogelijkheden. De open-source mogelijkheden voorzien geen database hosting zodat zij zelf geen servicekosten hebben. Het grootste aantal open-source mogelijkheden zijn op een basis van PHP geschreven. *Aangezien het customer service portaal moet aansluiten bij de bestaande infrastructuur is er geen voorkeur voor PHP.*

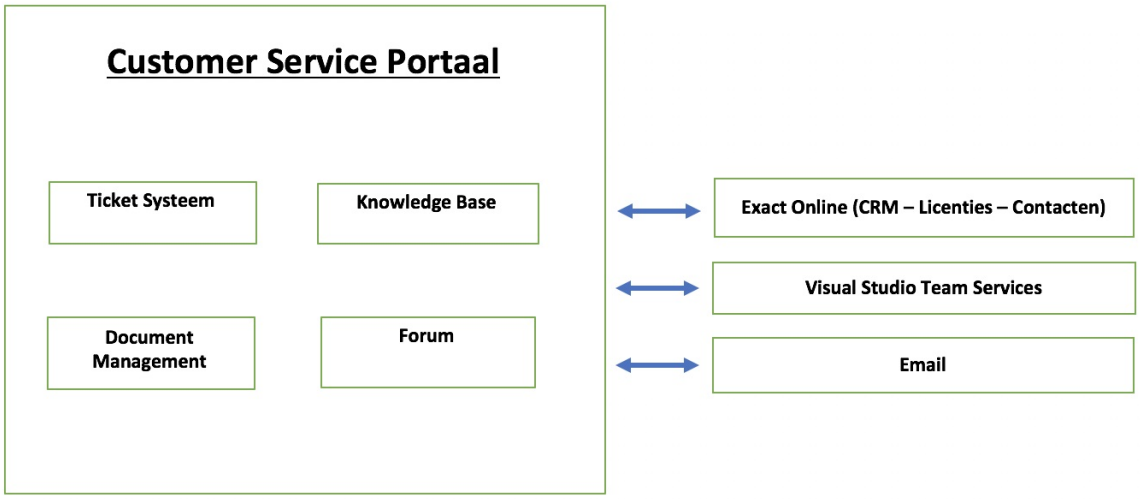
Ik ben vervolgens de mogelijkheden gaan bekijken die aansluiten bij de huidige website. Wordpress biedt een tal van plug-ins om digitale downloads te verkopen, helpdesk-systeem toe te voegen of een knowledgebase op te zetten. De ontwikkelaars van deze plug-ins verdienen hier echter hun brood mee en al snel was het duidelijk dat ze vele features enkel aanbieden als een premium-service. Met de resultaten van mijn onderzoek gebundeld hebben we een eerste meeting gehouden. Tijdens deze meeting heb ik alle mogelijke oplossingen overlopen en eventuele 'uitblinkers' zijn aan bod gekomen met een demo.

Op basis van de analyse en demo's hebben we tijdens de vergadering de richtlijnen van het project proberen nauwkeuriger te definiëren.

- Ticket Systeem
- Email Integratie met ticket systeem
- Integratie met Visual Studio Team Services
- Integratie met Exact Online
- Knowledge base met informatie
- Downloads en documenten voor geregistreerde gebruikers
- Notificaties voor updates

## Design

Aan de hand van de lijst met features kunnen we een eenvoudige maar duidelijke opdeling maken dat ons volgend schema oplevert:




We zien een duidelijke scheiding tussen features die we alleenstaand moeten implementeren en features waarvoor we gebruik maken van communicatie met externe services.

## De conclusie van de analyse

Er zijn verschillende oplossingen mogelijk: Bestaande helpdesk-tools, servicedesk-tools, wordpress plug-ins en IT-software Management-tools die voor vele features oplossingen bieden of gecombineerd kunnen worden om een resultaat te bekomen. Het was zeer een leerrijk onderzoek, voor zowel mezelf als Intation. We hebben features ontdekt, ondervonden en getest. We hebben bestaande tools op de markt gevonden die al dan niet geschikt zouden zijn, maar vooral een goede achtergrond opgebouwd om zelf een project te starten. We hebben gekozen om een asp.net core (2.0) met Entity framework te gebruiken omdat dit sterk aanleunt bij de bestaande stack van Intation

## Het resultaat

Het resultaat dat we willen bereiken is een Customer Service Portaal voor Intation waarop de klant kan registreren en inloggen. Daarna kan de klant support-tickets creëren en reeds aangemaakte tickets bekijken. Deze tickets worden op hetzelfde portaal door het supportteam behandeld. De ontwikkelaars kunnen met de visual studio team services integratie eenvoudig een bug of work-item aanmaken om hun software development te sturen en te voldoen aan de behoefte van de klant. Het support-ticket systeem moet een geïntegreerde email functionaliteit bevatten en een knowledgebase systeem, zodat klanten zichzelf kunnen helpen en op de hoogte worden gehouden van belangrijke updates. Waardoor het supportteam zijn tijd optimaal kan benuttigen. Het doel is de noodzakelijke features te implementeren en te streven naar een tool die kan ingezet worden naarmate het klantenbestand van Intation groeit.

# Resultaten

# Basis

De basis voor het project is een ASP.NET Core 2.0 applicatie waarbij een gebruiker eenvoudig kan inloggen en registreren. Vervolgens moeten er rollen aangemaakt worden en gebruikers aan bepaalde rollen worden toegekend. Deze basisfunctionaliteit moet uitgebreid worden met verificatie van het account met een bestaand emailadres. Nadien kunnen we de gebruikers en rollen permissies opleggen. Om de opstart van onze applicatie te vereenvoudigen en vullen met data moeten we deze 'seeden' bij de opstart van het project.

# Implementatie

## Gebruikers en opstart project

Wanneer je in Visual Studio een project aanmaakt heb je de mogelijkheid om gebruik te maken van een set basis tools, hierdoor is het opzetten van een eenvoudige webapplicatie en het toevoegen van individuele useraccounts een klein werk.

## Email verificatie

Het is vanzelfsprekend dat een gebruiker zich moet verifiëren op het portaal met een bestaand emailadres. Hiervoor was het noodzakelijk om een klasse op te stellen die zich enkel en alleen focust op het behandelen van emails. Aangezien we de functionaliteit van deze klasse op andere plaatsen zullen aanspreken bundelen we deze code in een service.

De email service klasse voegen we toe aan de startup klasse van het project voor dependecy injection. De gegevens die noodzakelijk zijn om een mail te versturen groeperen we in een nieuwe EmailSender klasse die deze data zal doorgeven aan de service. Als laatste heb ik de EmailSender klasse geïnjecteerd in de AccountController om beroep te kunnen doen op de SendMailAsync methode ervan.

Debugging was vereist aangezien de implementatie niet functioneerde naar behoren. Na onderzoek kwam een veel voorkomende fout naar boven, Gmail blokkeert inlogpogingen van apps of apparaten die oudere securityinstellingen gebruiken. De oplossing hiervoor was deze instellingen aanpassen zodat Gmail hier wel toegang tot verleent. Als laatste bleek echter hoe belangrijk de emailconfiguratie gegevens zijn, de smtp service is afhankelijk van je email provider. Wanneer je mails wil versturen van een office 365 emailadres zal deze instelling anders zijn dan bij Gmail (smtp.gmail.com).

## Rollen en permissies

De ingebouwde rolemanager van ASP.NET zorgt ervoor dat we eenvoudig rollen kunnen aanmaken en gebruikers hieraan toekennen. Het was even experimenteren en onderzoeken welke functionaliteiten er allemaal beschikbaar zijn. Ik heb vervolgens gekozen om 3 soorten rollen aan te maken: Admins, Agents en Customers. Vanzelfsprekend kan een admin rol alle functionaliteit uitoefenen die beschikbaar is op het webportaal. Een Agent is beperkt in het verwijderen en aanpassen van zaken maar kan alles bekijken. Een customer is nog beperkter en kan enkel en alleen zijn eigen tickets bekijken. Deze restricites kunnen we opleggen aan een role of meerdere rollen door met behulp van authorisatie de toegang tot een controller te weigeren, vervolgens bepaalt de rol welke html code te gebruiker te zien krijgt. Dit is uitgebreidt met een dashboard voor agent en admin waarin zij een overzicht krijgen van belangrijke zaken en instellingen kunnen aanpassen.

### Database seeden

Om niet telkens opnieuw gebruikers aan te maken, rollen te creëren en gebruikers hieraan toe te kennen heb ik beslist om de database te seeden wanneer er nog geen data in de database zit. Hiervoor heb ik een DBInitializer klasse aangemaakt die controleert of er al data in de database zit. Indien er geen data is gevonden worden er als eerst rollen aangemaakt, vervolgens creeëren we gebruikers waarvan we de emailverificatie reeds op true zetten, tenslotte voegen we de gebruikers toe aan een bepaalde rol.

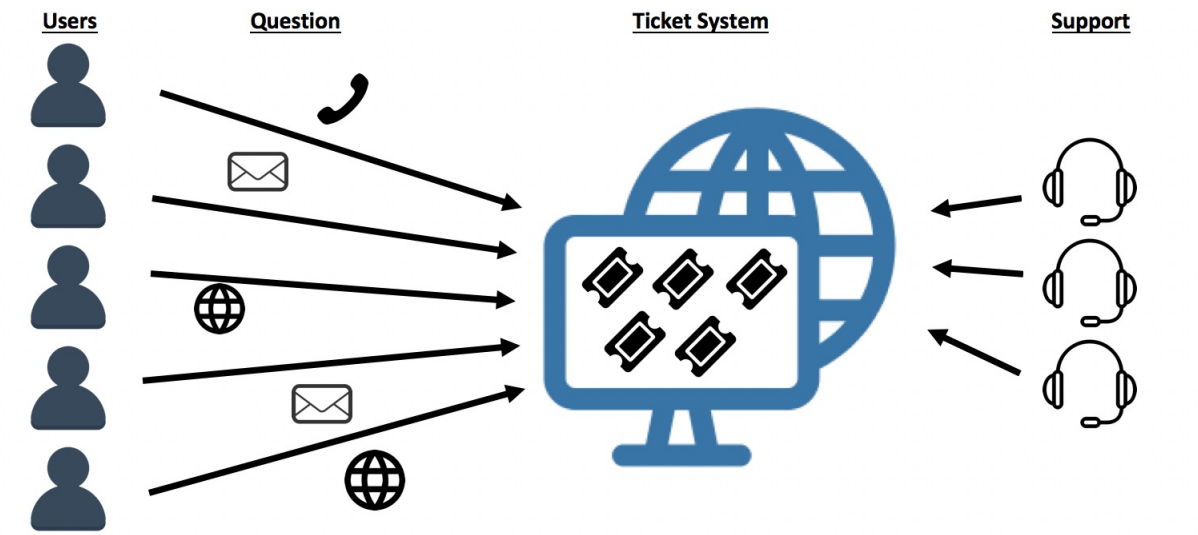
## Resultaat

Het resultaat is een eenvoudige webpagina waarop de gebruiker kan inloggen en registreren, de layout is aangepast aan de Intation vormgeving en stijl.

# Ticket Systeem

## Wat is een ticket systeem

Normaler wijze contacteer je een bedrijf via mail of telefoon om een antwoord te krijgen op je vraag. Wanneer een bedrijf of organisatie groeit zal het aantal klanten mee groeien en het aantal vragen oplopen. Om de 'support-vragen' te behandelen en een overzicht te bewaren wordt er gevraagd aan de klant om een formulier in te vullen op de website. Dit formulier wordt omgezet in een 'support-ticket', het ticket krijgt een uniek identificatie nummer zodat de klant en de support medewerker eenvoudig kunnen terugvallen op een bepaald ticket. Wanneer een ticket wordt aangemaakt krijgt dit de status 'open' mee, als het ticket in behandeling is wordt deze status omgezet in 'pending'. Vervolgens is er communicatie mogelijk tussen de klant en de support medewerker, wanneer de vraag opgelost is krijgt het ticket de status 'closed'.




## Implementatie

## Create, read, update en delete

Om van start te gaan met ticket objecten maken we gebruik van basis CRUD-operaties: create, read, update en delete. Aangezien we deze acties op verschillende plaatsen willen aanroepen zullen we deze samen bundelen in een ticket service. Om data te kunnen meegeven vanuit de razor pages moeten we invoervelden voorzien binnen een form zodat we deze met behulp van een button en databinding de data van de ingevoerde velden kunnen doorgeven aan onze backend.

Bij het updaten van een ticket vragen we eerst het gekoppelde ID op en met behulp van dit ID kunnen we het juiste ticket object opvragen en eventueel bewerken. Dezelfde flow wordt gevolgd bij het lezen of verwijderen van een object uit de database.

## Enums

Om functionele redenen en op aanraden van Cedric hebben we de status en prioriteiten als Enums opgeslagen. Waarom? Omdat deze variabelen maar uit een kleine set van mogelijkheden kan bestaan, zo kunnen er al geen fouten via strings insluipen. Als volgende heb ik een view gecreëerd waarbij we een lijst van tickets opvragen

afhankelijk van hun status. Om tickets nog eenvoudiger te kunnen behandelen hebben we na overleg geopteerd om de bestaande producten van intation toe te voegen als enum (InControl, InWave, InDocumentation).

## Categorieën

Een wijziging die later is toegevoegd is het toekennen van categorieën aan tickets, aangezien de categorieën nog niet gedefinieerd zijn willen we deze zelf kunnen toevoegen en instellen. Deze categorieën worden op dezelfde manier aangemaakt als een ticket en zijn enkel instelbaar door de administrators. Een klant kan bij het aanmaken van zijn ticket dankzij een dropdown de bijpassende categorie selecteren.

## One-to-many relatie

Een ticket op zich verteld veel over het probleem maar heeft weinig nut als er niet op geantwoord kan worden. Hiervoor moeten we dus de tickets kunnen voorzien van antwoorden, dit is een one-to-many relatie. We creëren hiervoor een nieuw Reply model en passen het ticket model aan door te zeggen dat deze objecten bevat van Reply. Vervolgens voorzien we de antwoorden van de crud mogelijkheden. Dit was minder eenvoudig als de Tickets aangezien we nu voor het opvragen van een antwoord het ticket opvragen en het antwoord ID includen, de include functionaliteit is ingebouwd in asp.net core. Om antwoorden te verwijderen kunnen we ook gebruik maken van Include waardoor we geen rekening moeten houden met een cascade delete. We willen natuurlijk niet dat een customer alle antwoorden kan verwijderen, maar enkel zijn eigen. Deze feature pakken we aan in de razorpages door te verbergen wanneer de identiteit gekoppeld aan het antwoord niet gelijk is aan de identiteit van de customer.

## HTML-tekst editor

Om een ticket nog verder te verrijken met informatie is het handig om tekst een bepaalde structuur mee te geven en een afbeeldingen of screenshot in te dienen. Op aanraden van cedric moest ik voor de tekst kijken naar een HTML-tekst editor. We hebben dit onderzocht en kwamen tot een gratis versie dat compatibel was met bootstrap. Summernote wat gebruikt maakt van javascript laat ons toe om tekst als html door te sturen en afbeeldingen als bit64. Dit slaan we volledig op als een string en dankzij de razorpages functionaliteit kunnen we met HTML.raw() de string terug omvormen en deze als HTML met afbeelding kunnen laten zien bij de tickets. Vervolgens hebben we dit ook toegepast op de antwoorden.

## Resultaat

Tickets kunnen op het portaal worden aangemaakt

# Knowledge Base

Een knowledge base of kennisbank is een collectie van informatie of 'kennis'. Een knowledge base gaat vaak samen met een ticket systeem doordat herhaalde vragen kunnen omgezet worden in een artikel waardoor mensen sneller geholpen kunnen worden. De support medewerkers moeten folders en artikels kunnen aanmaken en bewerken. De klanten moeten de knowledgebase kunnen doorzoeken

## Implementatie

Om folders aan te maken en nadien artikels toevoegen maken we zoals bij het ticket systeem gebruik van de CRUD-operaties. De mappen en artikels hebben een one-to-many relatie zoals de antwoorden op tickets, waardoor we eveneens gebruik kunnen maken van de include property in ASP.NET Core.

Mappen en artikels moeten aangemaakt kunnen worden door Agents en Administrators, hiervoor voorzien we een nieuwe 'Manage-knowledge view' gekoppeld aan de knowledge base sectie op het portaal. Om artikels een rijke context te kunnen meegeven maken we gebruik van dezelfde summernote editor als bij de tickets. Hierdoor kunnen we afbeeldingen en allerlei eenvoudig opslaan als string in de database.

## Resultaat

# Document Management Systeem

Document management is hoe iemand zijn elektronische documenten beheerd. Een document management systeem zorgt ervoor dat deze documenten eenvoudig beheerd kunnen worden. Intation wilt elektronische documenten beschikbaar stellen voor zijn klanten aan de hand van hun contractlevel. De documenten worden opgeslagen in sharepoint, sharepoint is een platform van Microsoft dat het mogelijk maakt om informatie te delen en samenwerking binnen een organisatie online te bevorderen.

Om dit te voltooien moeten we een integratie voorzien zodat de documenten vanuit Sharepoint beschikbaar zijn. Vervolgens is het noodzakelijk dat de beschikbare downloads gelimiteerd kunnen worden aan de hand van het contractlevel.

## Digitale downloads

Eerst en vooral moeten de documenten die aangeboden worden als digitale download beschikbaar zijn op de server. Hiervoor hebben we een lokale sharepoint map opgezet die automatisch synchroniseert met de online wijzigingen. Dit heeft als gevolg dat onze documenten altijd lokaal beschikbaar en up-to-date zullen zijn.

Vervolgens moet een administrator een path kunnen opslaan in de Database. We kunnen helaas wegens security redenen geen 'folder-picker' mogelijkheid voorzien vanuit het portaal. Dit zou wil zeggen dat de gebruikers de volledige mappen structuur kunnen onderzoeken van de server. We doen dus een suggestie aan de administrator van het lokale path waarin de documenten zich bevinden. Binnen Intation wordt steeds dezelfde structuur voor documenten van producten en projecten gehanteerd.

## Restricties

Nu het path naar de juiste folder beschikbaar is kunnen we voor elk document in deze map het path eveneens opvragen en opslaan in de database. ASP.NET beschikt over File providers waardoor we toegang kunnen geven tot lokale bestanden en deze retourneren als download in de browser aan de gebruiker. We specifieren het bestands-type en de naam voor de download en halen het bestand op dankzij de filepath dat is opgeslagen in de database. Een volledige map stellen we beschikbaar als een zip-bestand en enkele documenten als de extensie waarmee ze op de sharepoint locatie zijn opgeslagen.

Voorlopig zijn alle documenten zichtbaar en beschikbaar voor iedereen. Om dit te limiteren hebben we geopteerd om de documenten een level toe te kennen aan de hand van een Enum zodat we deze nadien kunnen vergelijken met het contractlevel van een bedrijf. Dit geeft ons de mogelijkheid om restricties op te leggen en te specifieren welke downloads al dan niet beschikbaar zijn voor bepaalde klanten. Wanneer een map of filepath wordt opgeslagen krijgen deze de enum 'invisible' toegekend, dit zorgt ervoor dat we in de razorpages de documenten onzichtbaar kunnen maken voor de gebruiker.

Nadien kan er per document een level worden ingesteld, de beschikbare levels zijn zoals de contractlevels: 'None = zichtbaar voor iedereen, Bronze = zichtbaar vanaf een bronze contractlevel, Silver = zichtbaar vanaf een silver contractlevel en Gold = enkel zichtbaar met een gouden contractlevel'. Dit wil zeggen alvorens een gebruiker op het portaal toegang heeft tot een document hij gekoppeld moet zijn aan een bedrijf.

## Resultaat

Klanten kunnen bestanden individueel downloaden of mappen als een gecomprimeerd bestand. De administrators kunnen documenten toevoegen door het juiste path op te slaan. Vervolgens kunnen restricties worden opgelegd aan de hand van contract- en documentlevels.

# Notificaties

Notificaties zijn zeer belangrijk voor een gebruiker zodat hij onmiddelijk op de hoogte wordt gesteld wanneer er een antwoord is op zijn vraag of er updates hebben plaats gevonden. Intation wil dat een gebruiker een email ontvangt wanneer er een ticket is aangemaakt of aangepast en wanneer er documenten of artikels zijn toegevoegd. De gebruiker moet zelf zijn notificatieinstellingen kunnen aanpassen. Wanneer nieuwe documenten worden toegevoegd is het vanzelfsprekend dat enkel de gebruikers die toegang hebben tot deze documenten een mail zullen ontvangen.

## Implementatie

De notificaties en meldingen bestaan voor Intation uit "Emails op maat". We hebben ervoor geopteerd om elke gebruiker de mogelijkheid te geven om emails te ontvangen over volgende zaken: Ticket Created, Ticket Update, New Article in KB, New Document. De gebruiker kan deze instellingen aanpassen in zijn persoonlijke menu. Wanneer er dus een nieuw ticket wordt aangemaakt of aangepast door of voor de gebruiker krijgt hij hiervan een mail. Wanneer een nieuw document wordt toegevoegd aan het portaal wordt er eerst gekeken welke gebruikers er in een bedrijf zitten en of zij toegang hebben tot deze documenten. Indien dit zo is en de gebruiker zijn meldingen aanstaan zal hij hiervan een mail ontvangen.

Hiervoor is er een nieuwe view voorzien waarbij elke gebruiker in zijn persoonlijke instellingen enkele checkboxen kan aanduiden of hij al dan niet meldingen wilt ontvangen van een bepaalde verandering. De checkboxen zijn gekoppeld aan een boolean die we controleren alvorens we een melding versturen. Er wordt gecontroleerd op de boolean van notificatie van de gebruiker voordat we de mailmanager de mail laten opstellen.

## Resultaat

De notifcaties zijn te vinden in het persoonlijke menu van de gebruiker. De instellingen kunnen eenvoudig worden aangepast met een enkele klik.

# Email Integratie

Email integratie in een ticketsysteem betekent dat een gebruiker kan antwoorden op zijn ticket via mail. Dit wil zeggen dat wanneer een ticket wordt aangemaakt op het portaal, de gebruiker een email zal ontvangen met als onderwerp een uniek identificatie nummer. Wanneer de gebruiker op de mail antwoord moet de context mee bij het juiste ticket verschijnen. Als er een email wordt verstuurd van een onbekend emailadres of het uniek ID is incorrect dat moet het systeem hierop kunnen anticiperen. Tenslotte wilt Intation dat de emails een persoonlijke touch hebben en de vormgeving en stijl aansluiten bij de website en het customer service portaal

## Implementatie

Een algemeen emailadres is noodzakelijk zodat emails steeds in dezelfde inbox terechtkomen, hiervoor hebben we een nieuw emailaccount aangemaakt via office 365 (support@intation.eu). Als eerste heb ik de mailing service die de confirmatiemails stuurt overgezet van mijn privé Gmail account naar het nieuwe support-mailadres.

## Nuget packages

Een NuGet package is een bestand met code gedeeld door een ontwikkelaar. Dankzij de ingebouwde NuGet tools in Visual Studio kan je eenvoudig NuGet packages downloaden en toevoegen aan je project, vervolgens kan je gebruik maken van de functionaliteit. Er zijn verschillende NuGet packages beschikbaar die deze mail functionaliteit ondersteunen. Ik heb gekozen voor Mailkit/Mimekit een NuGet Package met een MIT-licentie en een hoge rating aangezien deze beschikt over een goede documentatie en makkelijk uit te breiden is.

Als eerste slaan we de nieuwe inkomende mails op in de huidige directory als .EML-bestand. Ik heb hiervoor een nieuw model aangemaakt in de database om deze mails te kunnen inlezen en de jusite data eruit te halen die we nodig hebben. Al de voorgaande functionaliteit heb ik gebundeld in een een MailReceiver Service.

## Background tasks

Wanneer een ticket wordt aangemaakt krijgt dit een uniek identificatie nummer, dit nummer gebruiken we eveneens als onderwerp in de mails zodat deze eenvoudig worden teruggevonden door gebruiker en supportmedewerker. Aangezien het binnenhalen en uitlezen van emails een taak is die enkele seconden nodig heeft kan onze applicatie vertragen. Om dit te vermijden kan je gebruik maken van een backgroundtask die deze opdracht zal uitvoeren in de achtergrond.

De nieuwe inkomende mails worden in een lijst gestoken zodat we over deze lijst kunnen itereren. Intation heeft beslist dat niet voor elke inkomende mail een ticket moet worden aangemaakt, dit is beslist om spam of andere praktijken te vermijden. Een ticket kan dus alleen worden aangemaakt op het portaal. Wanneer een mail van een onbekende afzender wordt ontvangen stuurt het systeem een antwoord om eerst een account aan te maken op het portaal en vervolgens een ticket aan te maken. Als een gebruiker reeds gekend is wordt er gevraagd om een ticket aan te maken via het portaal. Wanneer de afzender, het subject en unieke ID gekend zijn zal de mail verder worden uitgelezen en wordt de context opgeslagen als antwoord in de database. Tenslotte markeren we deze mail als gelezen en wordt de mail verwijdert uit de inbox.

## Email templates

Om de emails persoonlijk te maken hebben we een ontwerp gemaakt dat zich baseert op de stijl en vormgeving van de Intation website. Emails bestaan uit html code, om consistentie te garanderen werken we met een tabel in html code zodat de mails die we willen versturen steeds een identieke vormgeving hebben. Vervolgens stijlen we de tabel aan de handt van de Intation kleuren. De email templates bevatten placeholders afhankelijk van hun onderwerp, deze

placeholders worden ingevuld het moment dat ze verstuurd worden. Zo kunnen we de naam van de gebruiker, url van de webpagina, ticket onderwerp en vele andere zaken meesturen waardoor de gebruiker een duidelijk overzicht kan behouden.

## Resultaat

Nieuwe mails worden uitgelezen en automatisch door het systeem behandelt. Wanneer bepaalde zaken in het portaal een mail triggeren wordt de mail verstuurd met bijhorende data. De email templates werken naar behoren.

# Visual Studio Team Services Integratie

Visual Studio Team Services is een cloudservice specifiek voor het samenwerken aan de ontwikkeling van software. VSTS beschikt over ingebouwde functionaliteit zoals een versie controle systeem, agile scrum methodes en meer. Om connectie te maken met Visual Studio Team Services heeft een support medewerker een persoonlijke API-key nodig die hijzelf kan aanmaken in het webportaal van VSTS. Wanneer een ticket behandeld wordt moet de support medewerker kunnen zien of hier reeds gekoppelde bugs of features aan toebehoren. Een support medewerker kan zelf een bug of feature aanmaken vanuit het customer service portaal.

## Implementatie

Visual Studio Team Services beschikt over een uitgebreide en goed gedocumenteerde API-bibliotheek waar ik mezelf eerst moest in verdiepen. Vervolgens hebben we samen de gewenste functionaliteiten samengesteld gebaseerd op de mogelijkheden met behulp van de API's. Als eerste moeten we API-token aanvragen om op een veilige manier GET en POST Request te kunnen sturen met onze VSTS account. Ten tweede heb is het antwoord van een API-call een JSON, we moeten dit omzetten in een object zodat we deze data kunnen uitlezen en toekennen. Met behulp van de NewSoft JSON library kunnen we de JSON deserializen in een object en deze data opslaan in de database.

Vervolgens hebben we een POST Request geschreven om een bug of feature te creëren. Deze functionaliteit hebben we ook gebundeld in een service. Het is ook van belang om een link te kunnen leggen tussen een ticket en een bug of feature. Hiervoor hebben we een bug model aangemaakt en het ticket model aangepast zodat deze objecten van een bug kan bevatten. Wanneer een koppeling wordt gemaakt wordt de database het ticket ID opgeslagen bij de bijhorende bug zodat we deze eenvoudig kunnen opvragen en laten zien bij het juiste ticket. De API-calls heb ik eerst getest via postman en nadien met behulp van een C# voorbeeld van VSTS geschreven in C# in het asp.net project.

Om de feature ook gebruiksnut te geven hebben we ervoor gezorgd dat de bugs worden weergegeven bij het ticket met hun onderwerp, ID, 'Assigned To' en door hier op te klikken eenvoudig genavigeerd kan worden naar de juiste VSTS-bug of feature. De retournerende informatie in het bug object was niet altijd even duidelijk waardoor ik bepaalde strings gesplit heb in C# met behulp van de split methode om relevante data duidelijker weer te geven op de razorpages. Als je op verschillend bugs klikt moet je ook het project meegeven wat verschillend kan zijn, dit heb ik opgelost door dit op te vragen wanneer een bug of feature wordt aangemaakt. Het kan ook zijn dat een ticket betrekking heeft tot een reeds bestaande bug of feature. Daarvoor heb ik de optie voorzien om deze manueel te koppelen door het ID van een bestaande bug mee te geven.

Voorlopig was mijn eigen API-token manueel geconfigureerd in de C# code. Omdat elke Agent een verschillende API-token heeft moeten we dit variabel kunnen instellen. In het dashboard heb ik daarom een invoerveld voorzien waarbij we een token kunnen invoeren en koppelen via de include functionaliteit van ASP.NET Core aan onze ApplicationUser. We willen ook dat de gebruiker zijn API-key kan verwijderen wanneer gewenst en een nieuwe toe voegen wanneer deze vervallen is. Ik heb dit efficiënt gemaakt door in de UI links toe te voegen waardoor het voor de gebruiker eenvoudig is om te navigeren naar de instellingen.

## Resultaat

We kunnen met behulp van REST API's met deze service communiceren, hiervoor moeten we eerst een token genereren. Deze token is specifiek voor een gebruiker en je kan hieraan verschillende permissies toekennen. De functionaliteit die we geïmplementeerd hebben in de webapplicatie zorgt ervoor dat tickets eenvoudig gekoppeld kunnen worden aan bugs of features. Wanneer een support medewerker een ticket behandelt kan hij eenvoudig de status en andere gegevens van de bug of feature bekijken om een gedetailleerd antwoord te kunnen geven.

# Exact Online Integratie

Het klantenbestand wordt vandaag beheerd in exact online, een uitgebreide business software - CRM tool voor bedrijven en zelfstandigen die enkele REST API's ter beschikking stelt. Deze tool wordt onder andere ook gebruikt voor de boekhouding en financiële gegevens. Wanneer een supportticket behandelt wordt geven we de supportmedewerker een overzicht in het portaal van de bijhorende bedrijfsgegevens. Om de bedrijfsgegevens te koppelen aan de juiste gebruikersaccounts van het customer service portaal wordt er een relatie opgezet in de database die kan worden ingesteld door de admin.

## Implementatie

De integratie met Exact Online was eerst gevraagd om te kijken naar API-mogelijkheden aangezien deze documentatie ook beschikbaar is op de website van Exact Online. Helaas kan een gegenereerde token geen verschillende beperkingen opleggen dan het account waarop deze gecreëerd is. Intation beschikt maar over enkele accounts bij exact online waarop de leidinggevende toegang tot alles hebben. Aangezien dit wil zeggen dat ik via de API-calls dan toegang heb tot alle vertrouwelijke data en deze ook kan verwijderen hebben we beslist om geen token te creëren met volledige toegang voor testing/development purposes. Het aanmaken van eender welke extra account met al dan niet eventuele beperkingen resulteert in een maandelijkse extra fee voor deze account. Om deze extra kost te vermijden bij het ontwikkelen van een applicatie hebben we geopteerd om gebruik te maken van de exportfunctie van Exact Online naar een XML of CSV-file.

## XML-upload

De data die we hieruit verkrijgen is te vergelijken met de data die we van een API-call zouden terugkrijgen. Waardoor we onze functionaliteit verder kunnen uitwerken, maar gebruik maken van een statisch dataformaat dat we handmatig kunnen updaten. Het XML-bestand bevat veel klanteninformatie dat we moeten filteren op noodzaak in onze applicatie. Hiervoor hebben we het XML-bestand onderzocht, dit bestand bevat Accounts, deze Accounts zijn bedrijven of klanten en hebben soms een lijst van contact objecten. ASP.NET Core heeft een ingebouwde functionaliteit om XML-bestanden te parsen.

Ik heb een object aangemaakt dat de data van deze file kan opslaan en vervolgens kunnen we een lijst weergeven van alle bedrijven. Om het XML bestand te uploaden zetten we dit eerste om in een ByteArray met de IFormFileExtension, de extensie zorgt ervoor dat we via databinding onze file kunnen uploaden in de razor pages en vervolgens omzetten in een ByteArray om op te slaan, echter kunnen we deze array niet zomaar uitlezen en zetten we deze om in een stream, deze stream valt perfect uit te lezen en data op te slaan in onze database. Het bestand wordt vervolgens geserialized naar een JSON object zodat dit overeen zou komen met de data van onze API-calls, deze gaan we dan deserializen om vervolgens onze data aan ons object toe te kennen en op te slaan in de database.

Dit JSON object is ook eenvoudiger om mee te werken dan het XML-bestand. Om het bedrijf en zijn data te kunnen koppelen aan een gebruiker heb ik een extra tabel aangemaakt die het companyID en UserID opslaat. Dit valt te vergelijken met hoe de rollen worden opgeslagen in ASP.NET Core. Vervolgens heb ik ervoor gezorgd dat we deze data manueel kunnen koppelen door een button actie en de juiste informatie kunnen weergeven bij de tickets door het opvragen van de gebruikers in het bedrijf object.

## REST API

De exact online integratie was succesvol afgerond met het uploaden van een XML-file, maar aan het einde van deze sprint is er beslist om toch een apart development account aan te maken en met de API te werken. De omvorming was niet geheel eenvoudig. Het waren maar kleine aanpassingen om de data te beheren, maar de moeilijkheid zat hem in de connectie met de API.

We communiceren met exact online aan de hand van API-calls, we kunnen klanten gegevens opvragen en hun details laten zien. We kunnen deze klanten objecten van Exact koppelen aan accounts van het customer service portaal. Het is zeer belangrijk wanneer met dit repliceert om de redirect-url te veranderen. Dit moet je ook specifiëren wanneer je een applicatie toevoegd in het Exact Online portaal.

Het opvragen van de klantengegevens gaat in verschillende stappen:

- Eerst moet je jezelf authentiseren met Exact Online, hiervoor gebruik je de redirect-url naar Exact Online.
- De response is een ACCESSTOKEN en Token Type (Bij een foute call vervalt de ACCESSTOKEN)
- Vervolgens moet je met behulp van de A-token de ‘Divisie’ opvragen, dit bepaald je toegang
- Response is een nieuwe token en je divisie (Deze token kan gebruikt worden voor meerdere calls)
- Vervolgens vragen we de Klanten Accounts op van Exact Online (Dit doen we door in onze request een filter op te stellen)
- We converteren de verkregen json en slaan de data op die we kunnen gebruiken
- Als laatste vragen we de “Main contacts” van de klant op ook met behulp van een filter
- Deze hoofdcontact personen voegen we vervolgens toe aan het klantenobject

## Resultaat

# Conclusie

Je reflecteert over het resultaat van je stage en bachelorproef. Vragen die hier beantwoord moeten worden zijn:

- Wat hebben we kunnen realiseren?
- Voldoet het resultaat aan de verwachtingen?
- Wat hebben we hieruit geleerd?

## My Multi Word Header

# Appendices

## ASP.NET Core 2.0

## Entity Framework Core

## Razor Pages

## Bootstrap

# Bibliografie

## Algemeen

[1] Vraag en antwoord website voor algemene vragen rond codering.
https://stackoverflow.com
[2] Microsoft documentatie voor basis van ASP.NET Core 2.0 en code voorbeelden.
https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.0

## API's

[3] Rest API documentatie van Microsoft voor Visual Studio Team Services.
https://docs.microsoft.com/en-us/rest/api/vsts/
[4] Rest API documentatie van Exact Online.
https://start.exactonline.nl/docs/HlpRestAPIResources.aspx?SourceAction=10

## Mail

[5] Documentatie voor het gebruik van Nuget Package Mailkit/Mimekit.
https://github.com/jstedfast/MailKit/tree/master/Documentation

# Lijst van Figuren

## NuGet

Een term die belangrijk is.Hieronder verschijnen alle pagina's waar deze term te vinden is. Volgende term als voorbeeld.

## Stack

## Repository

## Javascript

## JQuery

## Framework