

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



Table of Contents

Abstract	1.1
Dankwoord	1.2
Introductie	1.3
Structuur	1.4
Technologieën	2.1
Tools	2.1.1
Technologieën	2.1.2
Onderzoek	3.1
Resultaten	4.1
Basis	4.1.1
Ticket Systeem	4.1.2
Knowledge Base	4.1.3
Document Management Systeem	4.1.4
Notificaties en instellingen	4.1.5
Email Integratie	4.1.6
Visual Studio Team Services Integratie	4.1.7
Exact Online Integratie	4.1.8
Conclusie	5.1
Appendices	6.1
Bibliografie	6.2
Lijst Figuren	6.3
Glossary	6.4

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 om 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 een ticket aan te maken. Support medewerkers kunnen vervolgens op deze tickets antwoorden en klanten zijn op de hoogte van wijzigingen met behulp van notificaties. 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 kan Intation bedrijven en klanten gepersonaliseerde support mogelijkheden aanbieden. Tenslotte is er een knowledgebase en document management systeem voorzien waarbij geregistreerde klanten toegang hebben tot verschillende documenten en downloads. De gebruikers moeten worden geïnformeerd 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 die 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 om 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 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 en een knowledgebase systeem bevatten, zodat klanten zichzelf kunnen helpen en op de hoogte worden gehouden van belangrijke updates.

Deze structuur moet ervoor zorgen dat klanten efficiënt en snel kunnen worden geholpen, waardoor het supportteam zijn tijd optimaal kan benutten. 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 ontfermt 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. Git is een versiebeheersysteem en Agile is een manier van softwareontwikkeling.

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 applicaties 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, het wordt gebruikt om moderne cross-platform en cloud-based web applicaties te bouwen.

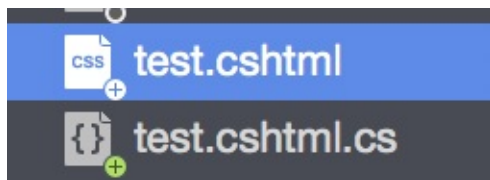
Entity Framework Core 2.0

Entity Framework is een open-source object relational mapper voor .NET applicaties, dat is 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. (1 = HTML, 2 = Klasse)



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 de 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 het 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.

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.*

Vervolgens ben ik 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.

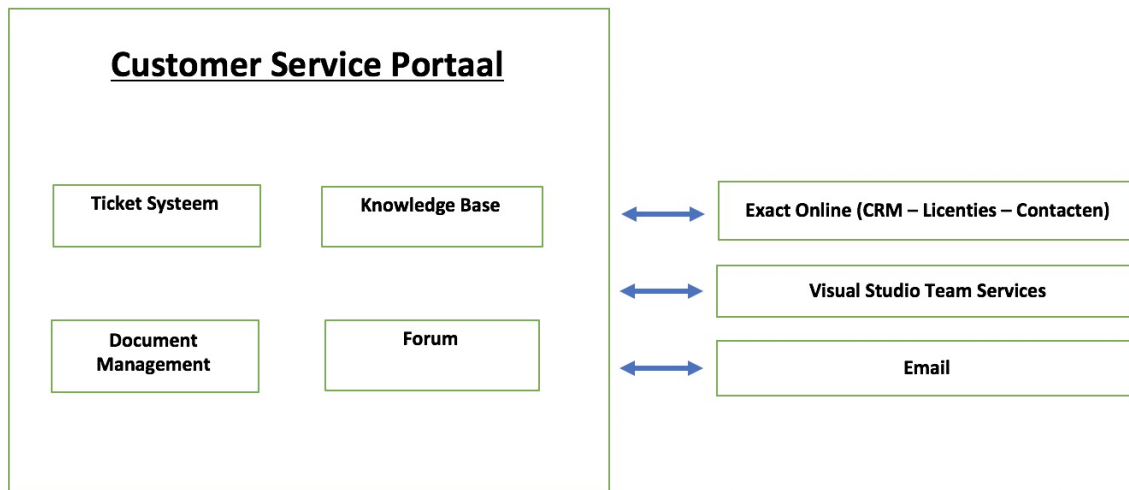
Op basis van de analyse en demo's hebben we tijdens de vergadering de richtlijnen van het project nauwkeurig te definiëren.

- 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:



Er is een duidelijke scheiding tussen features die alleenstaand moeten geïmplementeerd worden en features waarvoor we gebruik konden maken van communicatie met externe services.

De conclusie van de analyse

Er zijn verschillende oplossingen mogelijk:

- Bestaande helpdesk-tools
- servicedesk-tools
- wordpress plug-ins
- IT-software Management-tools

Elke mogelijkheid biedt min of meer voor de gevraagde eigenschappen een oplossing. Min of meer wilt zeggen dat sommige gecombineerd moeten worden om een resultaat te bekomen.

Het was een zeer 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. Er is 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 dat Intation wil bereiken is een Customer Service Portaal 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 en een knowledgebase systeem bevatten, zodat klanten zichzelf kunnen helpen en op de hoogte worden gehouden van belangrijke updates. Met als doel dat het supportteam zijn tijd optimaal kan benutten. Het is noodzakelijke voorgaande features te implementeren en te streven naar een tool die kan ingezet worden naarmate het klantenbestand van Intation groeit.

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 behulp van een bestaand emailadres. Nadien kunnen de gebruikers en rollen permissies worden opgelegd. Om de opstart van de applicatie te vereenvoudigen en de database te vullen met data moet deze automatisch gevuld en gecontroleerd worden bij de opstart van het project.

Start 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.

Registreren

In onderstaande code wordt een variabele aangemaakt van het type ApplicationUser, een klasse dat overerft van IdentityUser. IdentityUser is een standaard uit het Entity Framework dat gebruikt kan worden om gebruikers aan te maken en op te slaan in de database. De gegevens van de nieuwe gebruiker die registreert op het customer service portaal worden met behulp van een webformulier en databinding doorgegeven.

```
var user = new ApplicationUser { UserName = Input.Email, Email = Input.Email, Firstname = ... };
```

Vervolgens wordt de asynchrone methode CreateAsync aangeroepen om een gebruiker aan te maken via de userManager, dit voorziet de mogelijkheid om gebruikers te beheren in ASP.NET Core.

```
var result = await _userManager.CreateAsync(user, Input.Password);
```

Rollen

De ingebouwde rolemanager van ASP.NET zorgt ervoor dat er eenvoudig rollen kunnen aangemaakt worden. Er worden in de applicatie 3 rollen aangemaakt: Admin, Agent en Customer. Vanzelfsprekend kan een admin profiel 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. De restricties worden opgelegd aan een profiel of meerdere profielen met behulp van autorisatie. De toegang tot een controller kan worden geweigerd en de rol bepaald welke html code de gebruiker te zien krijgt.

```
var res = roleManager.CreateAsync(admin).Result;
```

De gebruiker wordt toegevoegd aan een bepaald profiel. Aangezien registratie volgens het portaal enkel voor klanten is, zullen deze allemaal worden toegekend aan het customer-profiel.

```
var AddToRole = _userManager.AddToRoleAsync(user, "Customer").Result;
```

</br>

Login

Tenslotte kan de gebruiker inloggen op het portaal met behulp van de `SignInManager`, deze standaard behandelt de autorisatie van de gebruikers. Er wordt gebruik gemaakt van de asynchrone methode `PasswordSignInAsync` om te controleren of de login gegevens correct zijn.

```
var result = await _signInManager.PasswordSignInAsync(Input.Email, Input.Password);
```

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 de functionaliteit van deze klasse op andere plaatsen zal worden aangesproken, wordt de code gebundeld in een service.

Vooraleer een methode uit de custom service `MailManager` aangeroepen wordt, genereert het systeem een code van het type `string` met behulp van de methode `GenerateEmailConfirmationTokenAsync`. Aan de hand van de code, `user-ID` en `url` van het portaal wordt er een `EmailConfirmatieLink` opgesteld. Deze unieke link wordt verstuurd naar het emailadres van de geregistreerde gebruiker.

```
var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
var callbackUrl = Url.EmailConfirmationLink(user.Id, code, Request.Scheme);
m_mailmanager.SendEmailConfirmationAsync(Input.Email, callbackUrl, Input.Email);
```

De `MailManager` maakt gebruik van de `SendEmailConfirmationAsync` methode. Deze methode leest de opgestelde template uit en vervangt de placeholders met de juiste gegevens. Door gebruik van `System.Net.Mail` wordt de mail via SMTP vanuit de ASP.NET Core applicatie verstuurd.

```
string body = File.ReadAllText(@"Templates\confirm_mail.html");
body = body.Replace("{url}", HtmlEncoder.Default.Encode(link));
body = body.Replace("{username}", username);

m_client.SendMailAsync(new MailMessage("support@intation.eu", to)
{
    Subject = "InSupport: New User Registration",
    IsBodyHtml = true,
    Body = body
});
```

Enkele problemen die ondervonden zijn:

Gmail blokkeert inlogpogingen van applicaties of apparaten die oudere securityinstellingen gebruiken. De oplossing hiervoor was deze instellingen aanpassen zodat Gmail hier wel toegang tot verleent.

De emailconfiguratie gegevens moeten correct zijn, de smtp service is afhankelijk van je email provider. Wanneer je mails wil versturen van een office 365 emailadres zullen deze instellingen anders zijn dan bij Gmail (smtp.gmail.com).

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 is er 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 worden gebruikers gecreëerd waarvan de emailverificatie reeds op `true` wordt gezet, tenslotte worden de gebruikers toegevoegd aan een bepaalde rol.

```
context.Database.EnsureCreated();
```

```
if (!context.Roles.Any(r => r.Name == "Admin"))  
{...}
```

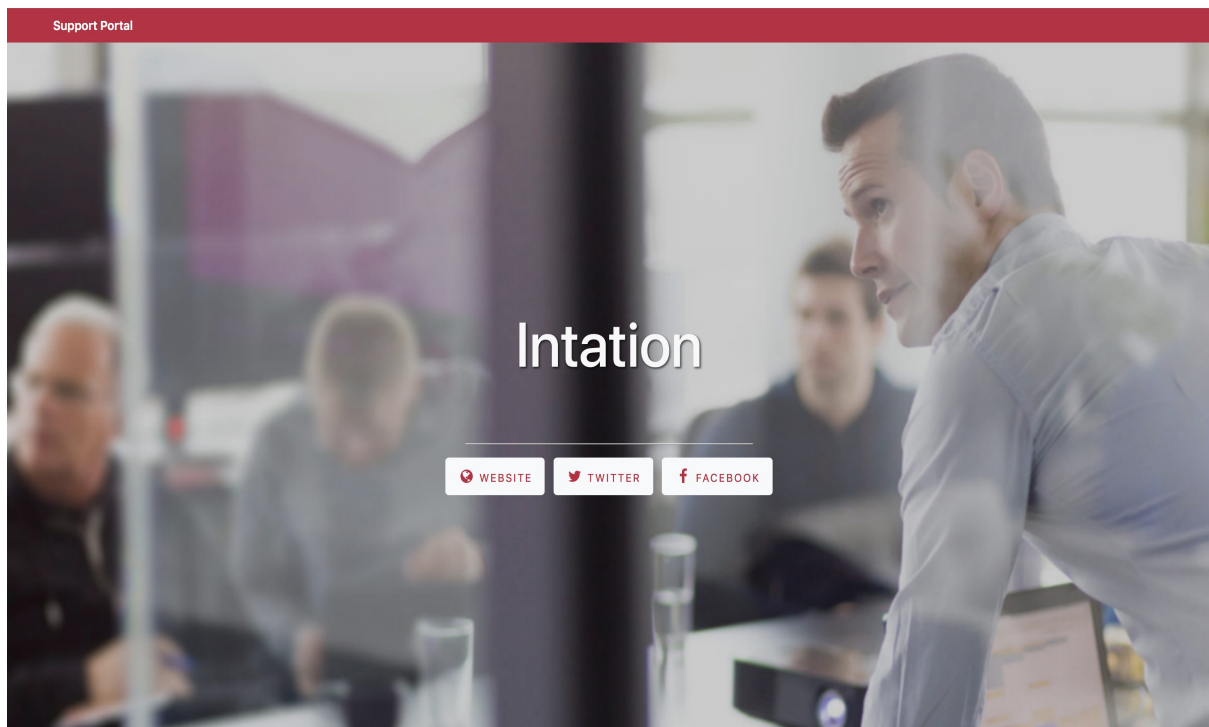
```
if (!context.Users.Any(u => u.UserName == "nathan.aertgeerts@intation.eu"))  
{...}
```

resultaat

Het resultaat is een webpagina waarop de gebruiker kan inloggen en registreren, de layout van de homepage is aangepast aan de Intation vormgeving en stijl. De basisgegevens van de gebruiker worden opgeslagen in de database en de gebruiker kan zijn emailadres bevestigen aan de hand van de bevestigingsmail die wordt verstuurd naar zijn emailadres.

InSupport Knowledge Documents Tickets ▾ Admin

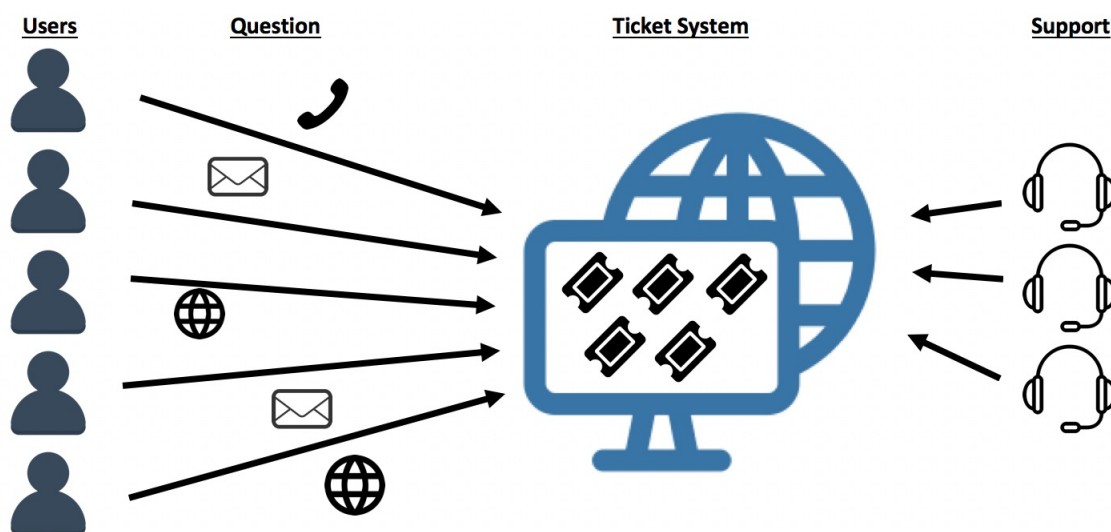
Hello nathan.aertgeerts@intation.eu! Log out



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'.



Create, read, update en delete

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

In onderstaande code wordt er een variable aangemaakt van het type ticket, met behulp van databinding kan er data worden doorgegeven uit een webformulier van de razor pages. De datum van het ticket wordt bepaald op het moment dat het ticket wordt aangemaakt, alsook de status bij een nieuw ticket zal steeds 'open' zijn. Vervolgens wordt het ticket meegegeven aan de methode AddTicket van de ticket service.

```

var ticket = new Models.Ticket()
{
    TicketSubject = Input.TicketSubject,
    TicketDetails = Input.TicketDetails,
    TicketRequestor = User.Identity.Name,
    TicketDate = DateTime.Now,
    PriorityType = Input.TicketPriority,
    StatusType = Models.Status.Open,
    ProductType = Input.TicketProduct,
};
  
```



```
m_ticketService.AddTicket(ticket);
```

De ticket service zal op zijn beurt het ticket aanmaken en opslaan in de database.

```
m_context.Ticket.Add(ticket)
m_context.SaveChanges();
```

Bij het updaten van een ticket wordt het gekoppelde ID meegegeven, met het ID kan het juiste ticket object worden opgevraagd en eventueel worden bewerkt. Om een object te zoeken in de database kan er gebruik worden gemaakt van verschillende methodes.

- FirstOrDefault()
- Where()
- Find()

FirstOrDefault en Where kunnen gebruikt worden op zowel een lijst, array als op een collectie. De methode Find kan enkel gebruikt worden op een lijst en zal opzoek gaan naar een specifieke voorwaarde. FirstOrDefault zal enkel het eerste element uit de lijst of collectie retourneren dat voldoet aan de voorwaarden. De methode Where zal alle items retourneren dat voldoen aan de voorwaarden.

Het object dat in de database gevonden wordt met het juiste ID zal toegekend worden aan de variabele UpdateTicket.

```
var UpdateTicket = m_context.Ticket.Find(id);
```

De gegevens van dit ticket worden geupdate.

```
UpdateTicket.PriorityType = ticketPriority;
UpdateTicket.StatusType = ticketStatus;
```

De aangepaste gegevens moeten worden opgeslagen in de database en het vernieuwde ticketobject wordt terug meegegeven.

```
m_context.SaveChanges();
return UpdateTicket;
```

Dezelfde flow wordt gevolgd bij het lezen of verwijderen van een object uit de database.

Enums

Om functionele redenen zijn het status- en prioriteitenveld als Enums opgeslagen. Omdat de variabelen maar uit een kleine set van mogelijkheden kan bestaan, zo kunnen er geen fouten insluipen via strings.

```
public enum Status
{
    Open = 1,
    Pending = 2,
    Closed = 3
}
```

Om tickets nog eenvoudiger te kunnen behandelen is er na overleg geopteerd om de bestaande producten van intation eveneens 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. De 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 zichzelf verteld veel over het probleem maar heeft weinig nut als er niet op geantwoord kan worden. Hiervoor moeten de tickets kunnen voorzien worden van antwoorden, dit is een one-to-many relatie. Er wordt hiervoor een nieuw Reply model gecreëerd en het ticket model wordt aangepast door te zeggen dat deze een lijst van objecten bevat van Reply.

```
public List<Reply> Replies { get; set; }
```

Om een antwoord toe te voegen wordt opnieuw het ticket object opgevraagd aan de hand van zijn ID. Er wordt gebruik gemaakt van de Include() functionaliteit in ASP.NET Core om de data van Reply mee op te vragen uit de database.

```
var getTicketById = m_context.Ticket.Include(x => x.Replies).FirstOrDefault(x => x.TicketId == id);
var dateNow = DateTime;

getTicketById.Replies.Add(new Reply()
{
    Content = ReplyContent,
    ReplyDate = dateNow,
    ReplyRequestor = user,
});

m_context.SaveChanges();
```

Vervolgens worden de antwoorden voorzien van de CRUD-operaties. Om antwoorden te verwijderen kan er ook gebruik worden gemaakt van de ingebouwde Include property in ASP.NET Core waardoor we geen rekening moeten houden met een cascade delete. Een customer kan enkel zijn eigen antwoorden verwijderen, in razor pages kan dit 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. Summernote wat gebruikt maakt van javascript laat ons toe om tekst als html door te sturen en afbeeldingen als bit64. Dit kan worden opgeslaan als een string en dankzij de razorpages functionaliteit kan je met HTML.raw() de string terug omvormen. Vervolgens wordt de HTML als afbeelding getoond bij het ticket. Tenslotte is dit ook toegepast op de antwoorden.

Om summernote te implementeren volstaat het om in de layout een stylesheet en javascript referentie te plaatsen

```
<link href="https://cdnjs.cloudflare.com/ajax/libs/summernote/0.8.9/summernote-bs4.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/summernote/0.8.9/summernote-bs4.js"></script>
```

Daarna plaats je volgend script in je razor page:

```
<script>
$(document).ready(function () {
    $('#summernote').summernote({
        height: 150
    })
    if ($('#summernote').summernote('maximumImageFileSize', '1048576')) {
```

```
        alert('File is too large.')
    };
});
</script>
```

De hoogte van de textarea wordt gedefinieerd met height, vervolgens limiteren we de uploadbare filesize. Het is belangrijk dat je in het input veld dat voorzien is voor de editor, het id meegeeft zoals je specificeert in je script.

```
id="summernote"
```

Resultaat

Tickets kunnen op het portaal worden aangemaakt door de gebruiker of support medewerker. Er kunnen antwoorden geplaatst worden op het ticket en deze kunnen rijkelijk gevuld worden met allerlei informatie.

InSupport Knowledge Documents Tickets ▾ Admin

Create Ticket

Requestor:

nathan.aertgeerts@intation.eu

Subject:

Subject*

Product:

InControl ▾

Priority:

Normal ▾

Category:

First Category ▾

Details:

Rich text editor toolbar: Bold, Italic, Underline, Link, Unlink, Apple System font, Color, Bulleted List, Numbered List, Indent, Outdent, Table, Link, Image, Video, Code, Help.

+ Create ticket

© 2018 - Intation

InSupport Knowledge Documents Tickets ▾ Admin

Hello nathan.aertgeerts

Ticket Updated: #INT-2
Your ticket has been up

Ticket View

Subject: test

Description:

test



test

nathanaertgeerts@gmail.com | 19 Apr 2018 - 08:50



test antwoord

nathan.aertgeerts@intation.eu | 03 May 2018 - 12:04

Antwoord:

Rich text editor toolbar: Bold, Italic, Underline, Link, Unlink, Apple System font, Color, Bulleted List, Numbered List, Indent, Outdent, Table, Link, Image, Video, Code, Help.

Reply

Cancel

Info Bugs Customer

Ticket ID: 2

nathanaertgeerts@gmail.com

InControl

4/19/2018 8:50:28 AM

2

0

Status:

Pending ▾

Priority:

High ▾

Category:

First Category ▾

Update

© 2018 - Intation

InSupport Knowledge Documents Tickets ▾ Admin







Hello nathan.aertgeerts@intation.eu! Log out

All Tickets

+ Create Ticket

Show 10 entries

Search:

Created	↑↓ Status	↑↓ Priority	↑↓ Category	↑↓ Product	↑↓ Subject	↑↓	↑↓	↑↓
1/1/0001 12:00:00 AM	● Open	Normal	First Category	InControl	First Ticket			
4/19/2018 8:50:28 AM	● Pending	High	First Category	InControl	test			
4/19/2018 9:07:23 AM	● Closed	Low	First Category	InControl	test			

Showing 1 to 3 of 3 entries

Previous

1

Next

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.

Artikels en folders

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

Mappen en artikels moeten aangemaakt kunnen worden door Agents en Administrators, hiervoor is een nieuwe razorpage gekoppeld aan de knowledgebase sectie op het portaal. Om artikels een rijke context te kunnen meegeven maken we gebruik van dezelfde summernote editor als bij de tickets. Hierdoor kan men afbeeldingen en allerlei eenvoudig opslaan als string in de database.

Knowledgebase doorzoeken

De knowledgebase kan eenvoudig doorzocht worden dankzij de methode Contains(). De Contains methode maakt het mogelijk om na te gaan of een string een bepaalde string bevat. De gebruiker kan dus op de webpagina een bepaalde zoekterm ingeven en de zoekterm wordt omgezet in kleine letters met de methode ToLower(). Vervolgens kijken we of de titel van een map of de content van een artikel de zoekterm bevat. Indien de zoekterm voorkomt in een titel of tekst zullen we enkel de gevonden folders of artikels tonen.

```
ViewData["CurrentFilter"] = searchString;
if (!String.IsNullOrEmpty(searchString))
{
    Maps = Maps.Where(x => x.Title.ToLower().Contains(searchString.ToLower())).ToList();

    Articles = Articles.Where(x => x.Content.ToLower().Contains(searchString.ToLower())).ToList();
}
```

Resultaat

Support medewerkers kunnen artikels en folders aanmaken. Artikels kunnen allerlei informatie bevatten en gebruikers kunnen eenvoudig zoeken tussen de folders of gebruik maken van de zoekfunctie.

InSupport Knowledge Documents Tickets ▾ Admin

Hello nathan.aertgeerts@intation.eu! Log out

Manage Article

Manage KB

Articles

Categories

Settings

Edit Article

Category:

FAQ

Edit Article

First Article in FAQ

Details:



The first article in FAQ

Edit

Cancel

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 er een integratie worden 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.

Beschikbare documenten

Eerst en vooral moeten de documenten die aangeboden worden als digitale download beschikbaar zijn op de server. Hiervoor is 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. Er kan wegens security redenen geen 'folder-picker' mogelijkheid voorzien worden vanuit het portaal. Dit zou wil zeggen dat de gebruikers de volledige mappen structuur kunnen onderzoeken van de server. Er wordt dus een suggestie gedaan 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.

Locaties van bestanden

Nu het path naar de juiste folder beschikbaar is, kan er voor elk document in deze map het path opgevraagd worden. Vervolgens wordt het filepath opgeslagen in de database.

```
string[] filesindirectory = Directory.GetDirectories(TargetDirectory.DirectoryPath);

FilePaths = filesindirectory.ToList();

foreach (var item in FilePaths)
{
    ...
}
```

Downloads

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. Het bestands-type en de naam voor de download worden gespecificeerd en het bestand wordt opgehaald met behulp van de File provider en het filepath dat is opgeslagen in de database.

```
public FileResult OnPostDownload()
{
    var fileName = Input.fileName;
    var DirectoryPath = Input.DirectoryP;
    var filePath = DirectoryPath + "/" + fileName;
    var fileExists = System.IO.File.Exists(filePath);
    var mimeType = System.Net.Mime.MediaTypeNames.Application.Octet;
    return PhysicalFile(filePath, mimeType, fileName);
}
```


Een volledige map stellen we beschikbaar als zip-bestand. Er wordt lokaal op de server een map aangemaakt met een zip extensie waarin de gevraagde documenten worden geplaatst.

```
public ActionResult OnPostDownload(string path)
{
    ZipFile.CreateFromDirectory(path, "support.zip");
    var mimeType = System.Net.Mime.MediaTypeNames.Application.Octet;
    return PhysicalFile(contentRootPath + "/support.zip", mimeType, "support.zip");
}
```

Restricties

Voorlopig zijn alle documenten zichtbaar en beschikbaar voor iedereen. Om dit te limiteren is er geopteerd om de documenten een level toe te kennen aan de hand van een Enum zodat deze nadien kunnen worden vergeleken met het contractlevel van een bedrijf. Dit geeft ons de mogelijkheid om restricties op te leggen en te specificeren 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 in de razorpages de documenten onzichtbaar kunnen gemaakt worden 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'.

```
SilverMaps = m_context.DMS.Where(x => x.DocumentLevel == DocumentLevel.None ||
                                     x.DocumentLevel == DocumentLevel.Bronze ||
                                     x.DocumentLevel == DocumentLevel.Silver).ToList();
```

```
if (CompanyDetails.ContractLevel == ContractLevel.Silver)
{
    Documents = SilverMaps;
}
```

Dit wil zeggen alvorens een gebruiker op het portaal toegang heeft tot een document hij gekoppeld moet zijn aan een bedrijf. De administrator kan gebruikers koppelen aan een bedrijf in het admin dashboard. Hiervoor is een one-to-many relatie opgesteld tussen een bedrijf en gebruikers (een bedrijf kan meerdere gebruikers bevatten). Er wordt gebruik gemaakt van een nieuwe database tabel die het user-ID en het company-ID opslaat. Deze werking is te vergelijken met gebruikers en rollen.

```
m_context.Add(new UserInCompany()
{
    UserId = UserId,
    CompanyId = CompanyId
});
```

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.

Document Settings

Show 10 ▾ entries

Search:

Available Maps ↑	Minimum Contract Level ↑
_Archive	Invisible ▾
Code Quality	Gold ▾
Development procedure	Gold ▾
Installers	Gold ▾
Manuals	Gold ▾
Results	Invisible ▾
Specifications	Invisible ▾
Testing	Invisible ▾

Showing 1 to 8 of 8 entries

Previous

1

Next

Save Changes

Notificaties

Notificaties zijn zeer belangrijk voor een gebruiker, zo kan hij of zij onmiddellijk op de hoogte worden gesteld wanneer er een antwoord is op een vraag, updates of nieuwigheden 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. Er is 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 wordt gecontroleerd alvorens een melding wordt verstuurd. Er wordt gecontroleerd op de boolean van de notificatie van de gebruiker voordat de mailmanager de mail zal opstellen.

```
foreach (var notifications in User.Notifications)
{
    SendMail = notifications.NewDocument;
}
if (SendMail == true)
{
    //mailmanager send mail
}
```

Resultaat

De notificaties zijn te vinden in het persoonlijke menu van de gebruiker en de instellingen kunnen eenvoudig worden aangepast.

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 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

Een algemeen emailadres is noodzakelijk zodat emails steeds in dezelfde inbox terechtkomen, hiervoor is een nieuw emailaccount aangemaakt via office 365 (support@intation.eu) binnen het Intation domein.

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 de functionaliteit voor het ontvangen van mails in een ASP.NET Core applicatie ondersteunen. Ik heb gekozen voor Mailkit/Mimekit een NuGet Package met een MIT-licentie en een hoge rating, omdat deze beschikt over een goede documentatie en makkelijk uit te breiden is.

Background tasks

Aangezien het binnenhalen en uitlezen van emails een taak is die enkele seconden nodig heeft kan de applicatie vertragen. Om dit te vermijden kan je gebruik maken van een backgroundtask die deze opdracht zal uitvoeren in de achtergrond. Een tweede voordeel aan deze aanpak is dat de mails automatisch worden opgevraagd.

Een background task kan worden geïmplementeerd dankzij de `IHostedService` interface, dit is een singleton klasse dat geactiveerd wordt wanneer de applicatie opstart en proper wordt afgesloten wanneer de applicatie sluit. Er zijn 3 manieren om een hosted service op te stellen:

1) Timed background tasks

Een getimed background task maakt gebruik van `System.Threading.Timer` klasse. De timer triggered de `Execute` methode en wordt gestopt wanneer `StopAsync` wordt aangeroepen.

2) Consuming a scoped service in a background task

De hosted service creëert een scope om de scoped background task op te lossen en zijn `Execute` methode aan te roepen.

3) Queued background tasks

De queued task worden gestart en gestopt volgens volgorde in de queue.

In dit project wordt er gebruik gemaakt van de timed background task omdat dit de mogelijkheid geeft telkens de mails op te halen na een bepaald aantal seconden. Onderstaande code verduidelijkt hoe de timed background task geïmplementeerd is.

```
protected abstract Task ExecuteAsync(CancellationToken cancellationToken);
```

```

public virtual Task StartAsync(CancellationToken cancellationToken)
{
    // Store the task we're executing
    _executingTask = ExecuteAsync(_stoppingCts.Token);

    // If the task is completed then return it,
    // this will bubble cancellation and failure to the caller
    if (_executingTask.IsCompleted)
    {
        return _executingTask;
    }

    // Otherwise it's running
    return Task.CompletedTask;
}

public virtual async Task StopAsync(CancellationToken cancellationToken)
{
    // Stop called without start
    if (_executingTask == null)
    {
        return;
    }

    try
    {
        // Signal cancellation to the executing method
        _stoppingCts.Cancel();
    }
    finally
    {
        // Wait until the task completes or the stop token triggers
        await Task.WhenAny(_executingTask, Task.Delay(Timeout.Infinite,
            cancellationToken));
    }
}

```

IMAP

Er wordt gebruik gemaakt van IMAP, Internet Message Access Protocol om de email berichten op te vragen over het internet. De emails worden niet gedownload op de computer, maar rechtstreeks gelezen van de emailserver. Er wordt een nieuw model aangemaakt in de database om deze mails te kunnen uitlezen en de juiste data op te slaan.

Met onderstaande code wordt er verbinding gemaakt met de mailserver en de emails uit de inbox worden uitgelezen. De host, port en Secure Socket Layer (veilige communicatie) moeten gedefinieerd worden. De nieuwe inkomende mails worden in een lijst gestoken zodat er over de lijst kan worden geïtereerd.

```

using (var client = new ImapClient())
{
    client.Connect("outlook.office365.com", 993, SecureSocketOptions.SslOnConnect);
    client.Authenticate("nathan.aertgeerts@intation.eu" + @"support@intation.eu", "*****");
    client.Inbox.Open(FolderAccess.ReadWrite);

    var uids = client.Inbox.Search(SearchQuery.All);
    foreach (var uid in uids)
    {
        ...
    }
}

```

De mail wordt vervolgens aangeduid als gelezen en verwijderd uit de inbox. Dit is essentieel om telkens nieuwe emails uit te lezen en niet de volledige inbox.

```
client.Inbox.AddFlags(uid, MessageFlags.Deleted, true);
client.Inbox.Expunge();
```

Intation heeft beslist dat niet voor elke inkomende mail een ticket moet worden aangemaakt, dit is om spam of andere praktijken te vermijden. Een ticket kan dus alleen worden aangemaakt op het portaal.

Eerst kijkt het systeem na of de gebruiker gekend is.

```
if (m_userManager.FindByEmailAsync(message.From.Mailboxes.FirstOrDefault().Address) != null)
{
    ...
}
```

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.

```
m_mailmanager.SendEmailRegisterFirst(...);
```

Als een gebruiker reeds gekend is wordt er gevraagd om een ticket aan te maken via het portaal.

```
m_mailmanager.SendEmailCreateFirst(...);
```

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. Om dit unieke ID uit het onderwerp van de mail te halen kijken we of het subject de voorafgaande code bevat, vervolgens bepalen we de positie van het unieke ID en halen we het ID uit de string.

```
if (message.Subject.Contains("#INT-"))
{
    int positionOfINT = message.Subject.IndexOf("#INT-") + 5;
}
```

Om het ID uit de string te halen maken we gebruik van Regex, regular expretion.

```
string subjectIsId = Regex.Match(Substring, @"\d+").Value;
```

Als de afzender, het subject en unieke ID gekend zijn zal de mail verder worden uitgelezen en wordt de context opgeslagen als antwoord op het ticket in de database.

```
m_ticketService.AddReply(...);
m_mailmanager.SendEmailTicketUpdate(...);
```

Email templates

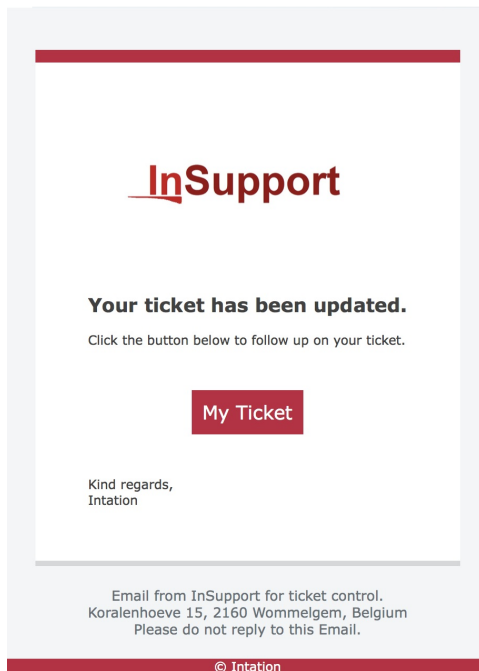
Om persoonlijke emails te sturen maakt de applicatie gebruik van een ontwerp dat zich baseert op de stijl en vormgeving van de Intation website. Emails bestaan uit html code en om consistentie te garanderen werken we met een tabel in html code zodat de mails die we willen versturen steeds een identieke vormgeving hebben. Vervolgens stijlen we de tabel aan de hand 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 in zijn mailbox krijgt.

Een voorbeeld van de templates kan u vinden in de appendices.

Resultaat

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

support@intation.eu
Ticket Updated: #INT-2
Aan: Nathan Aertgeerts



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.

Visual Studio Team Services beschikt over een uitgebreide en goed gedocumenteerde API-bibliotheek waar ik mezelf eerst in moest verdiepen. Vervolgens hebben we samen de gewenste functionaliteiten samengesteld gebaseerd op de mogelijkheden met behulp van de API's. Als eerste moet er een API-token worden aangevraagd om op een veilige manier GET en POST Request te kunnen sturen met een VSTS account. Ten tweede is het antwoord van een API-call een JSON, dit moet omgezet worden in een object zodat deze data kan worden uitgelezen en toegekend. Met behulp van de NewSoft JSON library kan de JSON worden gedeserialized in een object en de data worden opgeslagen in de database.

API calls

Elke support medewerker moet een eigen token aanmaken op VSTS om toegang te krijgen tot zijn projecten. De token moet dus instelbaar zijn per gebruiker, in het dashboard is daarom een invoerveld voorzien waarbij een token kan worden ingevoerd en gekoppeld via de include functionaliteit van ASP.NET Core aan onze ApplicationUser. De gebruiker kan zijn API-key ook verwijderen wanneer gewenst en een nieuwe toe voegen wanneer deze vervallen is. Dit is efficiënt gemaakt door in de user interface links toe te voegen waardoor het voor de gebruiker eenvoudig is om te navigeren naar de instellingen.

Om een bug of feature aan te maken wordt er gebruik gemaakt van een POST request. Een Bug of feature moet in VSTS een titel en beschrijving bevatten. Dit wordt geseerialized naar een JSON object en vervolgens meegestuurd in de POST request header. Het is ook van belang om een link te kunnen leggen tussen een ticket en een bug of feature. Hiervoor is een bug model aangemaakt en het ticket model aangepast zodat deze objecten van een bug kan bevatten. Wanneer een koppeling wordt gemaakt wordt in de database het ticket ID opgeslagen bij de bijhorende bug zodat deze eenvoudig kan worden opgevraagd en getoond worden bij het juiste ticket. De API-calls zijn eerst getest via postman en nadien met behulp van een C# voorbeeld van Visual Studio Team Services geschreven in C# in het asp.net project. De onderstaande code is een voorbeeld van een POST request om een feature of bug aan te maken.

```
var token = item.VSTToken;

string _personalAccessToken = token;
string _credentials = Convert.ToBase64String(System.Text.ASCIIEncoding.ASCII.GetBytes(
string.Format("{0}:{1}", "", _personalAccessToken)));

Object[] patchDocument = new Object[2];
patchDocument[0] = new { op = "add", path = "/fields/System.Title", value = Title };
patchDocument[1] = new { op = "add", path = "/fields/System.Description", value = Description };

using (var client = new HttpClient())
{
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(new
    System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
    client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", _credentials);

    var patchValue = new StringContent(JsonConvert.SerializeObject(patchDocument),
    Encoding.UTF8, "application/json-patch+json");
```



```

var method = new HttpMethod("PATCH");
var request = new HttpRequestMessage(method,
    "https://intation-development.visualstudio.com/" + Project + "/_apis/wit/workitems/$Feature?api-version=2
.2")
{ Content = patchValue };
var response = client.SendAsync(request).Result;

if (response.IsSuccessStatusCode)
{
    var result = response.Content.ReadAsStringAsync().Result;
    var rawResponse = JsonConvert.DeserializeObject<Bug>(result);
    AddBug(rawResponse.VstsBugId, TicketId);
}
}

```

Om de feature ook gebruiksnut te geven was het essentieel 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. Als je op verschillende 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.

```

<a href="https://intation-development.visualstudio.com/@Bug.ProjectName/_workitems?id=@Bug.VstsBugId">@Bug.Vst
sBugId</a>

```

Het kan zijn dat een ticket betrekking heeft tot een reeds bestaande bug of feature. Daarom is de optie voorzien om deze manueel te koppelen door het ID van een bestaande bug mee te geven. De ID van bugs en features zijn te vinden in VSTS.

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 behandeld 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.

REST API

Er wordt gecommuniceerd met exact online aan de hand van API-calls, De klantgegevens worden opgevraagd en hun details worden getoond. De klantobjecten van Exact worden gekoppeld aan accounts van het customer service portaal. Het is zeer belangrijk wanneer met dit replicateert om de redirect-url te veranderen. Dit moet je ook specificeren wanneer je een applicatie toevoegd in het Exact Online portaal.

Het opvragen van de klantgegevens gaat in verschillende stappen:

1) Autorisatie

Eerst moet je jezelf authenticeren met Exact Online, hiervoor gebruik je de redirect-url naar Exact Online. De redirect url moet je instellen bij het aanmaken van je token op het portaal van Exact Online. Vervolgens moet de webpagina in de applicatie worden omgeleid naar dezelfde url.

```
public IActionResult OnPostOauth()
{
    return Redirect("https://start.exactonline.be/api/oauth2/auth?client_id=
{****}&redirect_uri=http://192.168.5.20:5001/Admin/AdminExactOnline&response_type=code");
}
```

2) Accesstoken

De response die verkregen wordt na het invullen van correcte identificatie gegevens is een ACCESSTOKEN en Token Type (Bij een foute call vervalt de ACCESSTOKEN). De response is tevens een json object, om dit te kunnen uitlezen wordt dit gedeserialized met behulp van "Newtonsoft.Json", dit is een JSON framework voor .NET applicaties.

```
public void GetAPI(string Code)
{
    using (var client = new HttpClient())
    {
        var values = new Dictionary<string, string>
        {
            { "code", Code },
            { "redirect_uri", "http://192.168.5.20:5001/Admin/AdminExactOnline" },
            { "grant_type", "authorization_code" },
            { "client_id", "{****}" },
            { "client_secret", "*****" }
        };
        var content = new FormUrlEncodedContent(values);

        var method = new HttpMethod("POST");
        var request = new HttpRequestMessage(method, "https://start.exactonline.be/api/oauth2/token")
        { Content = content };
        var response = client.SendAsync(request).Result;

        if (response.IsSuccessStatusCode)
```

```

        {
            var result = response.Content.ReadAsStringAsync().Result;
            var rawResponse = JsonConvert.DeserializeObject<ExactToken>(result);
            var ACCESSTOKEN = rawResponse.access_token;
            var RefreshToken = rawResponse.refresh_token;
            var TokenType = rawResponse.token_type;

            GetDivision(ACCESSTOKEN, TokenType);
        }
    }
}

```

3) Divisie

Vervolgens moet je met behulp van de Accesstoken de 'Divisie' opvragen, de divisie is eigen aan Exact Online en bepaald je toegangsniveau.

```

var method = new HttpMethod("GET");
var request = new HttpRequestMessage(method, "https://start.exactonline.be/api/v1/current/Me?$select=CurrentDivision");

```

De response is een nieuwe token en je divisie, deze token kan in combinatie met je divisie gebruikt worden voor meerdere API calls.

4) API Calls

Klanten accounts

Vervolgens worden de klanten accounts van Exact Online opgevraagd, dit kan door in onze request een filter op te stellen. De response json wordt opnieuw gedeserialized en de klanten accounts worden toegekend aan het company object van het customer service portaal.

```

var method = new HttpMethod("GET");
var request = new HttpRequestMessage(method,
    "https://start.exactonline.be/api/v1/" + Division + "/crm/Accounts?$filter=Status eq'C'");

```

Main contacts

Als laatste worden de hoofdcontact personen van de klant opgevraagd ook met behulp van een nieuwe filter. De hoofdcontact personen wordt toegevoegd aan het klantenobject

```

var method = new HttpMethod("GET");
var request = new HttpRequestMessage(method,
    "https://start.exactonline.be/api/v1/" + Division + "/crm/Contacts?$filter=ID eq guid'" + MainContact + "'");

```

De klantgegevens kunnen manueel gekoppeld worden door een button actie zodat de juiste informatie kan worden weergegeven bij de tickets.

Problemen

Voor 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 token 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 hieruit verkregen wordt is te vergelijken met de data die je van een API-call zouden terugkrijgen, waardoor de functionaliteit verder kon worden uitgewerkt. Het verschil is dat er gebruik wordt gemaakt van een statisch dataformaat dat we handmatig moeten updaten.

Het XML-bestand bevat veel klanteninformatie dat we moeten filteren op noodzaak in onze applicatie. Hiervoor is 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 een lijst weergeven van alle bedrijven. Om het XML bestand te uploaden wordt dit eerst omgezet in een ByteArray met behulp van de `IFormFileExtension`, de extensie zorgt ervoor dat je via databinding de file kan uploaden in de razor pages en vervolgens omzetten in een ByteArray om op te slaan. De array kan echter niet zomaar uitgelezen worden, daarom wordt deze omgezet in een stream, de stream valt perfect uit te lezen en zodat de data kan worden opgeslagen in onze database. Het bestand wordt vervolgens geserializeerd naar een JSON object zodat dit overeen zou komen met de data van onze API-calls, dit wordt gedeserializeerd om vervolgens de data aan het object toe te kennen en op te slaan in de database.

```
var data = InputFile.ToByteArray();
Stream stream = new MemoryStream(data);

XmlDocument doc = new XmlDocument();
doc.Load(stream);

string json = JsonConvert.SerializeXmlNode(doc);
var rawResponse = JsonConvert.DeserializeObject<Rootobject>(json);
```

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.

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.

Resultaat

Company Details

Intation bv

Contact:

 Postbus 3999,4800 Breda, Nederland

Contract level:

Gold

Users:

Name	Email
Nathan Aertgeerts	nathanaertgeerts@hotmail.com
Nathan Aertgeerts	nathanaertgeerts@gmail.com
Nathan Aertgeerts	nathan.aertgeerts@student.ap.be
Nathan Aertgeerts	nathan.aertgeerts@intation.eu

Main Contact Person:

 CEO : Nico Van Hoorebeke

Update Company

Cancel

Ticket View

Subject: test

Description:

test



test


nathanaertgeerts@gmail.com | 19 Apr 2018 - 08:50



test antwoord

nathan.aertgeerts@intation.eu | 03 May 2018 - 12:04

Antwoord:



Reply

Cancel

Info Bugs Customer

Customer:

NathanAertgeerts

Company: Intation bv

Email:

Web:

Contractlevel: Gold

Main Contact Person: CEO : Nico Van Hoorebeke

Conclusie

Realisaties

In hoofdstuk 3 worden de resultaten gedetailleerd beschreven. Er is een customer service portaal in ASP.NET Core 2.0 met Entity Framework gerealiseerd.

De applicatie bevat volgende zaken:

- Ticketsysteem
- Email integratie
- Integratie met Visual Studio Team Services met behulp van REST-API
- Integratie met Exact Online met behulp van REST-API
- Knowledgebase
- Document management systeem

Ticketsysteem

Een gebruiker kan zichzelf registreren op het portaal en kan vervolgens een ticket aanmaken. De agents of administrators kunnen antwoorden op het ticket. Het antwoord komt overzichtelijk bij het ticket te staan en de klant krijgt een email in zijn mailbox. Vervolgens kan de klant rechtstreeks op de mail of via het portaal te conversatie verderzetten. De tickets kunnen met behulp van een ordelijke tabel éénvoudig gerangschikt of opgezocht worden. De applicatie werkt naar behoren en de grote hoeveelheid aan informatie wordt duidelijk weergegeven.

Email integratie

Emails worden automatisch verstuurd waardoor de gebruikers op de hoogte worden gehouden van updates. Ze kunnen dit op het portaal zelf instellen of ze al dan niet een email willen ontvangen over bepaalde zaken. De emails zijn afkomstig van het nieuwe emailadres "support@intation.eu". Mails die verstuurd worden naar dit adres worden automatisch in het systeem verwerkt, zodat tickets geupdate worden, gebruikers de juiste meldingen ontvangen en emails dat geen betrekking hebben tot het portaal worden gefilterd.

Integratie met Visual Studio Team Services met behulp van REST-API

Als agents hun eigen API-key aan maken in Visual Studio Team Services kunnen ze deze toevoegen aan het customer service portaal en zo zichzelf in directe verbinding brengen met de statussen van bugs en features binnen hun eigen projecten. Ze hebben ook de mogelijkheid om rechtstreeks uit het portaal bugs of features aan te maken in het juiste project. Hierdoor zal de efficiëntie positief worden beïnvloed.

Integratie met Exact Online met behulp van REST-API

De gegevens en data van klanten kan met behulp van API automatisch geupdate worden in het customer service portaal. De data van het bedrijf en de bijhorende contactpersonen worden gekoppeld en zijn zichtbaar in het portaal. Vervolgens kunnen we gebruikers van het customer service portaal synchroniseren met een bedrijf. Wanneer een bedrijf een contractlevel heeft meegekregen zal dit bepalend zijn voor de informatie dat de gekoppelde gebruikers van dit bedrijf kunnen zien.

Knowledgebase

Verschillende mappen of categorieën kunnen worden aangemaakt door de support medewerkers, zodat de knowledgebase artikels in de juiste folders terecht komen. Wanneer de knowledgebase groeit aan informatie kan de zoekfunctie worden gebruikt. De artikels kunnen gedetailleerd zijn doordat ze afbeeldingen, tabellen of code voorbeelden kunnen bevatten. Wanneer een nieuw artikel wordt toegevoegd zullen de gebruikers een mail ontvangen.

Document management systeem

Het Document management systeem zorgt ervoor dat support medewerkers documenten ter beschikking kunnen stellen en vervolgens restricties opleggen. Gebruikers kunnen documenten individueel downloaden of bepaalde folders als een gecomprimeerd bestand. De documenten zijn rechtstreeks beschikbaar uit Sharepoint vanop de server. Wanneer een nieuw document beschikbaar is worden alle gebruikers op wie dit van toepassing is op de hoogte gebracht.

Aanbevelingen

ASP.NET Core heeft een stijl leercurve, wanneer de kennis de bovenhand grijpt word je al snel geprikkeld om te experimenteren met de mogelijkheden. Het framework en de community bieden een goede ondersteuning en zijn ongetwijfeld een goede basis om eender welke web applicatie te bouwen.

Entity Framework Core in combinatie met ASP.NET is ongetwijfeld de beste aanpak om eenvoudig een database te beheren. De simpliciteit en mogelijkheden van Entity Framework bieden een beginnende en ervaren ontwikkelaar de mogelijkheid om veel sneller en productiever te programmeren.

Het gebruik van razor pages geeft de mogelijkheid om C# code te schrijven die achter de HTML pagina zit, waardoor de databinding zeer logisch is. Omgaan met user gegevens en input dat moet worden opgeslagen is in combinatie met Entity Framework een taak dat snel kan worden voltooid.

Toekomstig werk of uitbreidingen

De algemene stijl en user experience van het customer service portaal zou ongetwijfeld nog beter presteren aan de hand van verbeteringen door feedback van user testing. De interne performantie kan worden verbeterd door unit tests te implementeren. De twee bovenstaande factoren zullen bijdragen aan de code quality van het gehele project.

Om het ticket systeem uit te breiden kunnen we rekening gaan houden met de tijdsduur dat een ticket in behandeling is. Vervolgens hierop anticiperen en agents een melding sturen zodat er niets over het hoofd wordt gezien. Nadien kunnen we reports voorzien zodat de administrator een overzicht heeft van de efficientie. Een uitbreiding van Exact Online met een licentie module zal extra data opleveren, hierop moet dan opnieuw geanticipeerd worden zodat de support medewerkers continu de beste informatie hebben om de klant zo goed mogelijk te helpen. Om het document management systeem uit te breiden kunnen we gebruikers zelf een download map laten samenstellen. Support medewerkers een eenvoudigere oplossing bieden dan een statisch in te vullen bestandslocatie. De knowledgebase kunnen we optimaliseren door de zoekfunctie te optimaliseren.

Om de mogelijkheid te bieden aan gebruikers om elkaar te helpen kan een forum een meerwaarde bieden aan het customer service portaal. Hier kunnen dan zowel de gebruikers als de support medewerkers oplossingen aanbieden. Tot slot kan een eigen vorm van een uitgebreide licentie module een meerwaarde bieden om in de toekomst gepersonaliseerde support mogelijkheden aan te bieden.

Slot conclusie

We kunnen besluiten dat de opgelegde doelstellingen behaald zijn. De belangrijkste zaken voor Intation hebben hun plek gekregen in het project en zijn dankzij de consistente en positieve samenwerking allemaal geïmplementeerd.

Het was voor mij een nieuwe ervaring om met razor pages te werken. Alsook was de kennis van .NET in het begin nog niet te vergelijken met het niveau dat er bereikt is op het einde van de stage. Het aantal features was niet min, waardoor het overzicht soms te wensen overliet. Uiteindelijk hebben de features allemaal hun plaats gekregen en zijn deze met behulp van sprints en sprintmeetings volledig gedefinieerd. Doordat de samenwerking en communicatie zeer vlot is verlopen kon een bepaalde visie of aanpak snel worden bijgestuurd en dit had tot gevolg dat er weinig tot geen tijd verloren is gegaan.

Het geeft een enorme boost om terug te kijken op wat er verwezenlijkt is binnen de stageperiode, zowel op het vlak van kennis als persoonlijk. Ik ben tevreden met het eindresultaat, maar natuurlijk kijk ik er naar uit om in de toekomst zowel het project als mijn kennis van .NET te perfectioneren.

Appendices

In dit deel bevinden zich meer details over bepaalde zaken.

Code

De belangrijkste code is gegroepeerd per actie en hieronder terug te vinden.

- ChangeRoles ``

```
public void OnGet()
{
    Customers = m_userManager.GetUsersInRoleAsync("Customer").Result.ToList();
    Agents = m_userManager.GetUsersInRoleAsync("Agent").Result.ToList();
    Admins = m_userManager.GetUsersInRoleAsync("Admin").Result.ToList();
}
//Delete User
public IActionResult OnPost(string id)
{
    var user = m_userManager.FindByIdAsync(id).Result;
    //first update security stamp van de gebruiker die we verwijderen
    m_userManager.UpdateSecurityStampAsync(user);
    m_userService.DeleteUser(id);
}
```

```
// Clear the existing external cookie to ensure a clean login process
//HttpContext.SignOutAsync(IdentityConstants.ExternalScheme);

return RedirectToPage("../Admin/AdminChangeRoles");
}
```

* Companies

```
public void OnGet(string searchString) { Companies = m_exactOnlineService.GetCompanies();
```

```
    ViewData["CurrentFilter"] = searchString;
    if (!String.IsNullOrEmpty(searchString))
    {
        Companies = Companies.Where(x => x.Name.ToLower().Contains(searchString.ToLower())).ToList();

        if (Companies.Count == 0)
        {
            ModelState.AddModelError(string.Empty, "No matching results");
        }
    }
}
```

* CompanyDetails

```
[Authorize(Roles = "Admin")] public class AdminCompanyDetailsModel : PageModel { public class InputModel { public
Models.ContractLevel ContractLevel { get; set; }
```

```
}
```

```

[BindProperty]
public InputModel Input { get; set; }

private ExactOnlineService m_exactOnlineService;
private UserInCompanyService m_userInCompanyService;
private ApplicationDbContext m_context;
private UserManager<ApplicationUser> m_userManager;

public Companies Companies { get; set; }

public List<UserInCompany> UserInCompany {get; set;}
public List<ApplicationUser> UsersList { get; set; }

public AdminCompanyDetailsModel(ExactOnlineService exactOnlineService, UserInCompanyService userInCompanyService, ApplicationDbContext context, UserManager<ApplicationUser> userManager)
{
    m_exactOnlineService = exactOnlineService;
    m_userInCompanyService = userInCompanyService;
    m_context = context;
    m_userManager = userManager;
    this.UsersList = new List<ApplicationUser>();
}
public void OnGet(int id)
{
    Companies = m_exactOnlineService.GetCompanyById(id);

    UsersInCompany(id);
}

public List<ApplicationUser> UsersInCompany(int id)
{
    UserInCompany = m_userInCompanyService.GetUsersInCompany().FindAll(x => x.CompanyId == id).ToList();

    foreach (var item in UserInCompany)
    {
        var Gebruiker = m_userManager.FindByIdAsync(item.UserId).Result;
        UsersList.Add(Gebruiker);
    }

    return UsersList;
}

public IActionResult OnPostUpdateCompany(int id)
{
    var updateCompany = m_context.Company.FirstOrDefault(x => x.Id == id);

    updateCompany.ContractLevel = Input.ContractLevel;
    m_context.SaveChanges();

    return RedirectToPage("../Admin/AdminCompanies");
}
}

```

* CreateUser

```

public IActionResult OnPost() { var user = new ApplicationUser { UserName = Input.Email, Email = Input.Email ,
Firstname =Input.Firstname, Lastname = Input.Lastname}; var result = m_userManager.CreateAsync(user).Result; if
(result.Succeeded) { //add user to role var test2 = m_userManager.AddToRoleAsync(user, Input.Role).Result; //add
notification settings var userNotifications = m_userManager.Users.Include(x => x.Notifications).FirstOrDefault(x =>
x.UserName == user.UserName); userNotifications.Notifications.Add(new Models.Notifications() { TicketCreated =
true, TicketUpdate = true, NewArticle = true, NewDocument = true }); m_context.SaveChanges();

```

```

        //new account has been created
        //stuur een Email confirmatie mail ....
        var code = m_userManager.GenerateEmailConfirmationTokenAsync(user).Result;
        var callbackUrl = Url.EmailConfirmationNewTicketLink(user.Id, code, Request.Scheme);

        m_mailmanager.SendEmailConfirmationAsync(Input.Email, callbackUrl, Input.Email);

        // zorg ervoor dat de gebruiker hier ook zijn password kan setten
    }
    else
    {
        return RedirectToPage("../Error");
    }

    return RedirectToPage("../Admin/AdminChangeRoles");
}

```

* DMS

```

public IActionResult OnPostDeleteDirectory(int id)
{
    m_documentManager.DeleteTargetDirectory(id);
    return RedirectToPage();
}
public IActionResult OnPostAdd()
{
    //Save Path van directory voor documenten

    m_context.TargetDirectory.Add(new TargetDirectory
    {
        DirectoryPath = Input.NewPath
    });
    m_context.SaveChanges();
    return RedirectToPage("/Admin/AdminDMS");
}

```

* Documents

```

public void OnGet(string id)
{
    DirectoryPath = id;

    string[] filesindirectory = Directory.GetFiles(DirectoryPath);

    FilePaths = filesindirectory.ToList();

}

public FileResult OnPostDownload()
{
    var fileName = Input.fileName;
    var DirectoryPath = Input.DirectoryP;
    var filePath = DirectoryPath + "/" + fileName;
    var fileExists = System.IO.File.Exists(filePath);
    var mimeType = System.Net.Mime.MediaTypeNames.Application.Octet;
    return PhysicalFile(filePath, mimeType, fileName);
}

```

* DocumentSettings

```

public void OnGet(int Id)
{
    TargetDirectory = m_documentManager.GetTargetDirectoryByID(Id);

    string[] filesindirectory = Directory.GetDirectories(TargetDirectory.DirectoryPath);

    FilePaths = filesindirectory.ToList();
    Docs = m_context.DMS.ToList();
    //for elke filepath kijken of deze al bestaat, anders toevoegen aan DB
    foreach (var item in FilePaths)
    {
        var newPath = Docs.Find(x => x.FilePath == item);
        if (newPath == null)
        {
            TargetDirectory.FilePaths.Add(new Models.DMS
            {
                FilePath = item,
                DocumentLevel = Models.DocumentLevel.Invisible
            });
            //add new => document always invisible
            m_context.SaveChanges();
        }
        else
        {
            //path already in DataBase => content will still be available
        }
    }

    Docs = TargetDirectory.FilePaths.ToList();
}

public IActionResult OnPostUpdate()
{
    //Update de Document levels van de bestaande paths
    foreach (var item in Docs)
    {
        var Document = m_context.DMS.FirstOrDefault(x => x.Id == item.Id);
        OldLevel = Document.DocumentLevel;

        var newLevel = m_context.DMS.FirstOrDefault(x => x.Id == item.Id);
        newLevel.DocumentLevel = item.DocumentLevel;

        m_context.DMS.Update(newLevel);
        m_context.SaveChanges();

        //controleren op nieuwe document wijzigingen??
        if (newLevel.DocumentLevel != OldLevel)
        {
            if (newLevel.DocumentLevel == DocumentLevel.None)
            {
                LevelNone = LevelNone + 1;
            }
            if (newLevel.DocumentLevel == DocumentLevel.Bronze)
            {
                LevelBronze = LevelBronze + 1;
            }
            if (newLevel.DocumentLevel == DocumentLevel.Silver)
            {
                LevelSilver = LevelSilver + 1;
            }
            if (newLevel.DocumentLevel == DocumentLevel.Gold)
            {
                LevelGold = LevelGold + 1;
            }
        }
    }
}

```

```

        }
    }
}

//stuur mail naar contacten uit bedrijf waarop verandering document van toepassing is.
CheckfornewItems(LevelNone, LevelBronze, LevelSilver, LevelGold);

return RedirectToPage();
}

public void CheckfornewItems(int none, int bronze, int silver, int gold)
{
    //Als document levels geupdate worden moeten we mails sturen naar users in companies die hun meldingen
aan hebben staan
    var link = Url.Documents(Request.Scheme);
    //alle gebruikers die gekoppeld zijn aan een bedrijf
    UserInCompanyList = m_userInCompanyService.GetUsersInCompany();
    foreach (var GebruikersInBedrijf in UserInCompanyList)
    {
        //user object oprvagen voor elke user in bedrijf
        var UserObject = m_userManager.FindByIdAsync(GebruikersInBedrijf.UserId).Result;
        var CompanyObject = m_context.Company.FirstOrDefault(x => x.Id == GebruikersInBedrijf.CompanyId);

        if (CompanyObject.ContractLevel == ContractLevel.None)
        {
            if (none != 0)
            {
                m_mailmanager.SendEmailNewDocument(UserObject.Email, link);
            }
        }
        if (CompanyObject.ContractLevel == ContractLevel.Bronze)
        {
            if (none != 0 || bronze != 0)
            {
                m_mailmanager.SendEmailNewDocument(UserObject.Email, link);
            }
        }
        if (CompanyObject.ContractLevel == ContractLevel.Silver)
        {
            if (none != 0 || bronze != 0 || silver != 0)
            {
                m_mailmanager.SendEmailNewDocument(UserObject.Email, link);
            }
        }
        if (CompanyObject.ContractLevel == ContractLevel.Gold)
        {
            if (none != 0 || bronze != 0 || silver != 0 || gold != 0)
            {
                m_mailmanager.SendEmailNewDocument(UserObject.Email, link);
            }
        }
    }
}
}

```

* ExactOnline

public void OnGet(string code) {

```

    Code = code;

    if (Code != null)
    {
        GetAPI(Code);
    }
}

```

```

    }
    public IActionResult OnPostOAuth()
    {
        return Redirect("https://start.exactonline.be/api/oauth2/auth?client_id={*****}&redirect_uri=http://192.168.5.20:5001/Admin/AdminExactOnline&response_type=code");
    }
    public void GetAPI(string Code)
    {
        using (var client = new HttpClient())
        {
            var values = new Dictionary<string, string>
            {
                { "code", Code },
                { "redirect_uri", "http://192.168.5.20:5001/Admin/AdminExactOnline" },
                { "grant_type", "authorization_code" },
                { "client_id", "{208b2aab-272c-4123-a35d-fc872079bceb}" },
                { "client_secret", "FcjRzuk0QvEf" }
            };
            var content = new FormUrlEncodedContent(values);

            var method = new HttpMethod("POST");
            var request = new HttpRequestMessage(method, "https://start.exactonline.be/api/oauth2/token") { Content = content };
            var response = client.SendAsync(request).Result;

            //if the response is successful, set the result to the workitem object
            if (response.IsSuccessStatusCode)
            {
                var result = response.Content.ReadAsStringAsync().Result;

                var rawResponse = JsonConvert.DeserializeObject<ExactToken>(result);

                var ACCESSTOKEN = rawResponse.access_token;
                var RefreshToken = rawResponse.refresh_token;
                var TokenType = rawResponse.token_type;

                GetDivision(ACCESSTOKEN, TokenType);
            }
        }
    }
    public void GetDivision(string Token, string Type)
    {
        using (var client = new HttpClient())
        {
            //set our headers
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
            client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Token);

            var method = new HttpMethod("GET");
            var request = new HttpRequestMessage(method, "https://start.exactonline.be/api/v1/current/Me?$select=CurrentDivision");
            var response = client.SendAsync(request).Result;

            //if the response is successful, set the result to the workitem object
            if (response.IsSuccessStatusCode)
            {
                var result = response.Content.ReadAsStringAsync().Result;

                var rawResponse = JsonConvert.DeserializeObject<ExactDivision>(result);

                foreach (var item in rawResponse.d.results)
                {
                    var division = item.CurrentDivision;
                    GetAccountGUID(Token, division);
                }
            }
        }
    }

```

```

    }
}
}
public void GetAccountGUID(string Token, int Division)
{
    using (var client = new HttpClient())
    {
        //set our headers
        client.DefaultRequestHeaders.Accept.Clear();
        client.DefaultRequestHeaders.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHeaderValue
("application/json"));
        client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Token);

        var method = new HttpMethod("GET");
        var request = new HttpRequestMessage(method, "https://start.exactonline.be/api/v1/" + Division + "/"
crm/Accounts?$filter=Status eq'C'");

        //Working API calls
        //Juiste !!!
        //crm/Accounts?$filter=Status eq'C'

        ///api/v1/{division}/crm/Contacts?$filter=ID eq guid'00000000-0000-0000-0000-000000000000'&$select=
Account,AccountIsCustomer = lijst nog te groot, geeft klanten contact personen
        ///crm/Contacts?$filter=AccountIsCustomer eq true = contacten is customer geeft alle contacten van
bedrijf terug
        ///crm/Accounts?$select=Type = geeft type A, accounts terug
        ///crm/Accounts?$select=AccountManager
        //////////////////////////////////////
        ///crm/Contacts?$filter=IsMainContact eq true and AccountIsCustomer eq true === te weinig data???
        //////////////////////////////////////

        var response = client.SendAsync(request).Result;

        //if the response is successfull, set the result to the workitem object
        if (response.IsSuccessStatusCode)
        {
            //als we tot hier geraakt zijn, gaan we ervan uit dat we alle contacten en companies hebben kun
nen binnenhalen,
            //dus hier verwijderen we de contacten lijst om nadien een clean add te doen voor contacten per
bedrijf.

            if (m_context.Contacts.ToList() != null)
            {
                var OLDContactList = m_context.Contacts.ToList();
                foreach (var contact in OLDContactList)
                {
                    m_context.Contacts.Remove(contact);
                    m_context.SaveChanges();
                }
            }

            var result = response.Content.ReadAsStringAsync().Result;
            var rawResponse = JsonConvert.DeserializeObject<ExactCompany>(result);

            var list = m_context.Company.ToList();
            foreach (var Company in rawResponse.d.results)
            {
                var result2 = list.FirstOrDefault(x => x.Name == Company.Name);
                if (result2 == null)
                {
                    m_exactOnlinService.AddCompany(Company.Name, Company.AddressLine1, Company.Email, Compa
ny.Code, Company.Website, Company.Phone, Company.Postcode, Company.CountryName, Company.City);
                }
                else
            }

```



```

        {
            var updateCompany = m_context.Company.FirstOrDefault(x => x.Name == Company.Name);

            updateCompany.Phone = Company.Phone;
            updateCompany.Address = Company.AddressLine1;
            updateCompany.Email = Company.Email;
            updateCompany.Code = Company.Code;
            updateCompany.Postalcode = Company.Postcode;
            updateCompany.Country = Company.CountryName;
            updateCompany.City = Company.City;

            m_context.SaveChanges();

        }
        GetMainContact(Token, Division, Company.MainContact, Company.Name);
    }
}

public void GetMainContact(string Token, int Division, string MainContact, string CompanyName)
{
    using (var client = new HttpClient())
    {
        //set our headers
        client.DefaultRequestHeaders.Accept.Clear();
        client.DefaultRequestHeaders.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
        client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Token);

        var method = new HttpMethod("GET");
        var request = new HttpRequestMessage(method, "https://start.exactonline.be/api/v1/" + Division + "/" +
            "crm/Contacts?$filter=ID eq guid'" + MainContact + "'");
        var response = client.SendAsync(request).Result;

        //if the response is successfull, set the result to the workitem object
        if (response.IsSuccessStatusCode)
        {
            var result = response.Content.ReadAsStringAsync().Result;

            var rawResponse = JsonConvert.DeserializeObject<ExactMainContact>(result);

            foreach (var item in rawResponse.d.results)
            {
                m_exactOnlinService.AddContact(CompanyName, item.FullName, item.FirstName, item.LastName, item.Email, item.BusinessEmail, item.Phone, item.BusinessPhone, item.Mobile, item.BusinessMobile, item.JobTitleDescription);
            }
        }
    }
}

```

* Settings

```

public void OnGet()
{
    ApiKeys = m_userService.GetApiKeys(User.Identity.Name);
}

public IActionResult OnPostSetVSTSToken()
{
    m_userService.AddAPIKey(models.Source.VSTS, User.Identity.Name, Input.Key, DateTime.Now);
    // userService ophalen en de functie Add VSTS token aanroepen
    return RedirectToPage("../Admin/AdminSettings");
}

```

```
public IActionResult OnPostDeleteToken(int id)
{
    m_userService.DeleteAPIKey(Input.id);
    return RedirectToPage("../Admin/AdminSettings");
}
```

* Tickets

//onGet ticket by ID binnenhalen public IActionResult OnGet(int id) { CategoryList = m_ticketService.GetCategories();
ApiKey = m_userService.GetApiKeys(User.Identity.Name);

```
var APIkey = m_userManager.Users.Include(x => x.ApiKeys).FirstOrDefault(x => x.UserName == User.Identity.Name).ApiKeys.ToList();
if (APIkey.Count() == 0)
{
    //hier kan je nog error handling voorzien
}
else
{
    foreach (var item in APIkey)
    {
        if (item.Source == Source.VSTS)
        {
            var token = item.VSTToken;
            m_VSTSService.GetAllBugs(token);
        }
        else
        {
            //hier kan je nog error handling voorzien
        }
    }
}

Ticket = m_ticketService.GetTicketById(id);
Input = new InputModel()
{
    TicketId = id
};

if (Ticket == null)
{ return RedirectToPage("../Tickets/MyTickets"); }

UserDetails = m_userService.GetUserByMail(Ticket.TicketRequestor);
if (m_userInCompanyService.GetUsersInCompany().FirstOrDefault(x => x.UserId == UserDetails.Id) == null)
{
    Page();
}
else
{
    CompanyDetails = GetCompanyDetails(UserDetails.Id);
}

return Page();
}

//Delete ticket
public IActionResult OnPostDelete(int id)
{
    Ticket = m_ticketService.DeleteTicket(id);
```

```

        return RedirectToPage("../Tickets/MyTickets");
    }

    //Update ticket
    public IActionResult OnPostUpdate(int id)
    {
        Ticket = m_ticketService.UpdateTicket(id, Input.TicketPriority, Input.TicketStatus);
        m_ticketService.AddCategoryToTicket(Ticket.TicketId, Input.Category);

        SendMailTicketUpdate(id);

        return RedirectToPage();
    }

    //Add reply
    public IActionResult OnPostReply(string ReplyContent)
    {
        SendMailTicketUpdate(Input.TicketId);

        m_ticketService.AddReply(User.Identity.Name, DateTime.Now, Input.ReplyContent, Input.TicketId);

        return RedirectToPage();
    }

    //Delete Reply
    public IActionResult OnPostDeleteReply(int id)
    {
        Reply = m_ticketService.DeleteReply(Input.ReplyId);

        return RedirectToPage();
    }

    //Add bug
    public IActionResult OnPostLink(string WorkItemId)
    {
        m_VSTSService.AddBug(Input.WorkItemId, Input.TicketId);

        return RedirectToPage();
    }

    public void SendMailTicketUpdate(int Id)
    {
        var callbackUrl = Url.TicketLink(Request.Scheme);
        m_mailmanager.SendEmailTicketUpdate(Input.TicketRequestor, callbackUrl, Id);
    }

    //public void GetUsersInCompany(string userId)
    //{
    //    var CompanyFromUser = m_userInCompanyService.GetUsersInCompany().FirstOrDefault(x => x.UserId == user
    Id);
    //}
    public Companies GetCompanyDetails(string userId)
    {
        var CompanyFromUser = m_userInCompanyService.GetUsersInCompany().FirstOrDefault(x => x.UserId == userId
    );
        var company = m_context.Company.Include(x => x.Contacten).FirstOrDefault(x => x.Id == CompanyFromUser.C
    ompanyId);
        return company;
    }
}

```

* CreateBug

```

public IActionResult OnPostCreateBug()
{
    var description = Input.Description + " " + Url.TicketLinkId(Request.Scheme);
    m_VSTService.CreateBug(User.Identity.Name, Input.Title, description, Input.TicketId, Input.Project);

    return RedirectToPage("../Tickets/MyTickets");
}

//Service
//Create Bug
public void CreateBug(string name, string Title, string Description, int TicketId, string Project)
{
    var user = m_userManager.Users.Include(x => x.ApiKeys).FirstOrDefault(x => x.UserName == name).ApiKeys.
    ToList();
    foreach (var item in user)
    {
        if (item.Source == Source.VSTS)
        {
            var token = item.VSTStoken;

            string _personalAccessToken = token;
            string _credentials = Convert.ToBase64String(System.Text.ASCIIEncoding.ASCII.GetBytes(string.Fo
rmat("{0}:{1}", "", _personalAccessToken)));

            Object[] patchDocument = new Object[2];

            patchDocument[0] = new { op = "add", path = "/fields/System.Title", value = Title };
            patchDocument[1] = new { op = "add", path = "/fields/Microsoft.VSTS.TCM.ReproSteps", value = De
scription };

            //use the httpclient
            using (var client = new HttpClient())
            {
                //set our headers
                client.DefaultRequestHeaders.Accept.Clear();
                client.DefaultRequestHeaders.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHea
derValue("application/json"));
                client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", _ creden
tials);

                //serialize the fields array into a json string
                var patchValue = new StringContent(JsonConvert.SerializeObject(patchDocument), Encoding.UTF
8, "application/json-patch+json");

                var method = new HttpMethod("PATCH");
                var request = new HttpRequestMessage(method, "https://intation-development.visualstudio.com
/" + Project + "/_apis/wit/workitems/$Bug?api-version=2.2") { Content = patchValue };
                var response = client.SendAsync(request).Result;

                //if the response is successfull, set the result to the workitem object
                if (response.IsSuccessStatusCode)
                {
                    var result = response.Content.ReadAsStringAsync().Result;

                    var rawResponse = JsonConvert.DeserializeObject<Bug>(result);

                    AddBug(rawResponse.VstsBugId, TicketId);
                }
            }
        }
    }
}
}
}
}

```

* CreateFeature

```

public IActionResult OnPostCreateFeature()
{
    var description = Input.Description + " " + Url.TicketLinkID(Request.Scheme);
    m_VSTSService.CreateFeature(User.Identity.Name, Input.Title, description, Input.TicketId, Input.Project
);

    return RedirectToPage("../Tickets/MyTickets");
}

// Service
public void CreateFeature(string name, string Title, string Description, int TicketId, string Project)
{
    var user = m_userManager.Users.Include(x => x.ApiKeys).FirstOrDefault(x => x.UserName == name).ApiKeys.
ToList();
    foreach (var item in user)
    {
        if (item.Source == Source.VSTS)
        {
            var token = item.VSTSToken;

            string _personalAccessToken = token;
            string _credentials = Convert.ToBase64String(System.Text.ASCIIEncoding.ASCII.GetBytes(string.Fo
rmat("{0}:{1}", "", _personalAccessToken)));

            Object[] patchDocument = new Object[2];

            patchDocument[0] = new { op = "add", path = "/fields/System.Title", value = Title };
            patchDocument[1] = new { op = "add", path = "/fields/System.Description", value = Description }
;

            //use the httpclient
            using (var client = new HttpClient())
            {
                //set our headers
                client.DefaultRequestHeaders.Accept.Clear();
                client.DefaultRequestHeaders.Accept.Add(new System.Net.Http.Headers.MediaTypeWithQualityHea
derValue("application/json"));
                client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", _ creden
tials);

                //serialize the fields array into a json string
                var patchValue = new StringContent(JsonConvert.SerializeObject(patchDocument), Encoding.UTF
8, "application/json-patch+json");

                var method = new HttpMethod("PATCH");
                var request = new HttpRequestMessage(method, "https://intation-development.visualstudio.com
/" + Project + "/_apis/wit/workitems/$Feature?api-version=2.2") { Content = patchValue };
                var response = client.SendAsync(request).Result;

                //if the response is successfull, set the result to the workitem object
                if (response.IsSuccessStatusCode)
                {
                    var result = response.Content.ReadAsStringAsync().Result;

                    var rawResponse = JsonConvert.DeserializeObject<Bug>(result);

                    AddBug(rawResponse.VstsBugId, TicketId);
                }
            }
        }
    }
}

```

```
* CreateTicket
```

```
public IActionResult OnPost() {
```

```
    var ticket = new Models.Ticket()
    {
        TicketSubject = Input.TicketSubject,
        TicketDetails = Input.TicketDetails,
        TicketRequestor = User.Identity.Name,
        TicketDate = DateTime.Now,
        PriorityType = Input.TicketPriority,
        StatusType = Models.Status.Open,
        ProductType = Input.TicketProduct,
    };

    m_ticketService.AddTicket(ticket);
    m_ticketService.AddCategoryToTicket(ticket.TicketId, Input.Category);

    var user = new ApplicationUser { UserName = Input.TicketRequestor, Email = Input.TicketRequestor };
    var result = m_userManager.CreateAsync(user).Result;
    if (result.Succeeded)
    {
        //add user to role
        var test2 = m_userManager.AddToRoleAsync(user, "Customer").Result;
        //add notification settings
        var userNotifications = m_userManager.Users.Include(x => x.Notifications).FirstOrDefault(x => x.UserName == user.UserName);
        userNotifications.Notifications.Add(new Models.Notifications()
        {
            TicketCreated = true,
            TicketUpdate = true,
            NewArticle = true,
            NewDocument = true
        });
        m_context.SaveChanges();
        //new account has been created
        //stuur een Email confirmatie mail ....
        var code = m_userManager.GenerateEmailConfirmationTokenAsync(user).Result;
        var callbackUrl = Url.EmailConfirmationNewTicketLink(user.Id, code, Request.Scheme);

        m_mailmanager.SendEmailConfirmationAsync(Input.TicketRequestor, callbackUrl, Input.TicketRequestor)
    ;

        // zorg ervoor dat de gebruiker hier ook zijn password kan setten
    }
    else
    {
        //account already in DB
        //stuur mail dat er een ticket is aangemaakt
        SendMailTicketCreate(ticket.TicketId);
    }

    //na post van ticket, redirect naar my ticket lijst
    return RedirectToPage("../Tickets/MyTickets");
}

public void SendMailTicketCreate(int Id)
{
    var callbackUrl = Url.TicketLink(Request.Scheme);
    m_mailmanager.SendEmailTicketCreate(Input.TicketRequestor, callbackUrl, Id);
}
```

* Upload

```
public IActionResult OnPost() { if (InputFile == null) { return Page(); } if (!InputFile.FileName.EndsWith(".xml")) { return Page(); } else { var data = InputFile.ToArray(); Stream stream = new MemoryStream(data);
```

```
        XmlDocument doc = new XmlDocument();
        doc.Load(stream);

        string json = JsonConvert.SerializeXmlNode(doc);

        var rawResponse = JsonConvert.DeserializeObject<Rootobject>(json);

        var test = rawResponse.eExact.Accounts.Account.ToList();

        var list = m_context.Company.ToList();

        foreach (var Company in test)
        {

            var result = list.FirstOrDefault(x => x.Name == Company.Name);
            if (result == null)
            {
                m_exactOnlineService.AddCompany(Company.Name, Company.Address.AddressLine1, Company.Email,
Company.code, Company.HomePage, Company.Phone, Company.Address.PostalCode, Company.Address.Country, Company.Address.City);
            }
            else
            {
                var updateCompany = m_context.Company.FirstOrDefault(x => x.Name == Company.Name);

                updateCompany.Phone = Company.Phone;
                updateCompany.Address = Company.Address.AddressLine1;
                updateCompany.Email = Company.Email;
                updateCompany.Code = Company.code;
                updateCompany.Postalcode = Company.Address.PostalCode;
                m_context.SaveChanges();
            }

        }

        return RedirectToPage("../Admin/AdminCompanies");
    }
}
```

* ManageArticle

```
public void OnGet(int id) { Article = m_knowledgeService.GetArticleById(id); AllCategories = m_knowledgeService.GetMaps(); foreach (var item in AllCategories) { if (item.Articles == null) { //no articles } else { foreach (var Article in item.Articles) { if (Article.Id == id) { Category = m_knowledgeService.GetMapById(item.Id); } } }
```

```
    }

    }

    public IActionResult OnPostUpdateArticle()
    {
        m_knowledgeService.UpdateArticle(Input.Id, Input.Title, Input.Content, Input.CategoryID);
        return RedirectToPage("../Knowledge/ManageArticles");
    }

    public IActionResult OnPostDelete()
```

```
{
    m_knowledgeService.DeleteArticle(Input.Id);
    return RedirectToPage("../Knowledge/ManageKnowledge");
}
```

* ManageArticles

```
public IActionResult OnPostCreateArticle() { m_knowledgeService.CreateArticle(Input.Title, Input.Content, Input.Id);
var link = Url.Knowledgebase(Request.Scheme);
```

```
    var Customers = m_userManager.GetUsersInRoleAsync("Customer").Result.ToList();

    foreach (var Customer in Customers)
    {
        m_mailManager.SendEmailNewArticle(Customer.Email, link);
    }

    return RedirectToPage("../Knowledge/ManageArticles");
}
```

* ManageKnowledge

```
[Authorize(Roles = "Admin, Agent")]
public class ManageKnowledgeModel : PageModel
{
    public class InputModel
    {
        public string Title { get; set; }
    }
    [BindProperty]
    public InputModel Input { get; set; }

    public List<Models.KnowledgeMap> Maps { get; set; }

    private KnowledgeService m_knowledgeService;

    public ManageKnowledgeModel(KnowledgeService knowledgeService)
    {
        m_knowledgeService = knowledgeService;
    }
    public void OnGet()
    {
        Maps = m_knowledgeService.GetMapsAndArticles();
    }

    public IActionResult OnPostMap()
    {
        m_knowledgeService.CreateMap(Input.Title);
        Maps = m_knowledgeService.GetMapsAndArticles();

        return RedirectToPage("../Knowledge/ManageKnowledge");
    }
}
```

* NotifcationSettings


```
public IActionResult OnPost() { UserNotifications = m_userManager.Users.Include(x =>
x.Notifications).FirstOrDefault(x => x.UserName == User.Identity.Name);
```

```

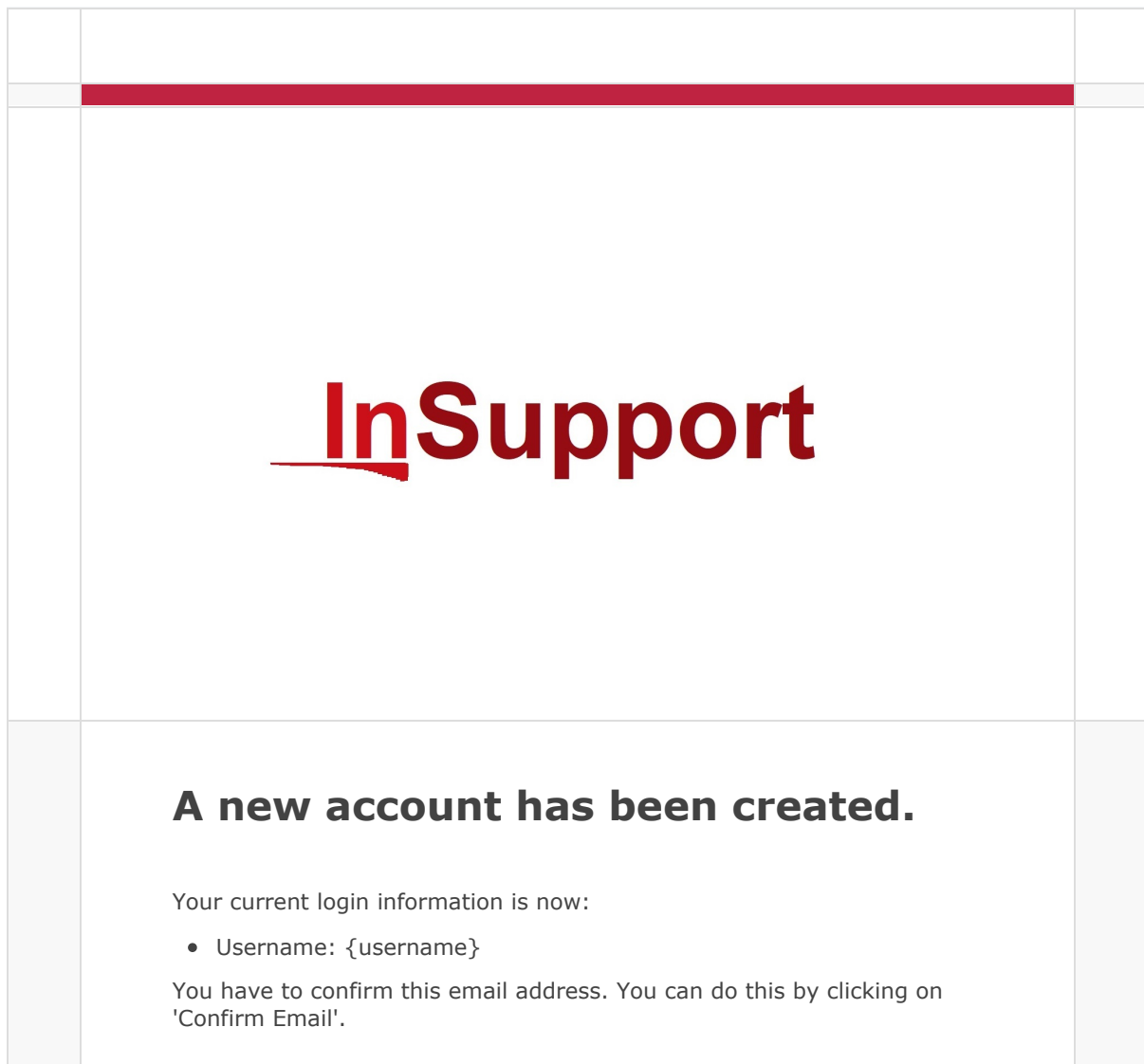
//delete settings
m_context.Remove(UserNotifications.Notifications.First());

//set new settings
UserNotifications.Notifications.Add(new Models.Notifications()
{
    NewArticle = Input.NewArticle,
    NewDocument = Input.NewDocument,
    TicketCreated = Input.TicketCreated,
    TicketUpdate = Input.TicketUpdate,
});
m_context.SaveChanges();

return RedirectToPage();
}

```

Email Template



		Confirm Email	
	Kind regards, Intation		
		One-time email from Incontrol for password confirmation. Koralenhoeve 15, 2160 Wommelgem, Belgium Please do not reply to this Email.	
		© Intation	

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>

Glossary

Framework

Een essentiële structuur of basis van software.

NuGet

Een door Microsoft ondersteund mechanisme voor het delen van code.

NuGet Package

Een NuGet package is een bestand met code gedeeld door een ontwikkelaar.

NuGet Tools

Tools in Visual Studio om eenvoudig NuGet packages te downloaden en toevoegen aan je project, vervolgens kan je gebruik maken van de functionaliteit.

Stack

Een conceptuele structuur van frameworks die de voorkeur hanteren binnen een organisatie of software ontwikkeling.

Repository

Een centrale plaats of opslag waar code en hun vorige versies worden bijgehouden.

Javascript

Een object georiënteerde programmeertaal gebruikt om interactieve effecten te maken in webpagina's.

JQuery

JQuery is een Javascript-framework om eenvoudig dynamische en interactieve webpagina's te bouwen.

API

Application Programming Interface een verzameling van definities op basis waarvan een computerprogramma kan communiceren met een ander programma of service

CRM

Customer Relationship Management is een engelse benaming voor klantenrelatiebeheer, vaak wordt met deze term verwezen naar software dat het eenvoudig maakt om al de factoren van CRM te beheren.

ORM

Object Relational Mapping een programmeer techniek om eenvoudig objecten te beheren in een database

EF

Entity Framework is een open source object relation mapper voor het .NET framework.