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

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

Aertgeerts Nathan

academiejaar 2017-2018

AP Hogeschool Antwerpen
Wetenschap &Techniek
Elektronica-ICT

ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

Table of Contents

i. Abstract

Een customer service portaal in ASP.NET Core 2.0 met Entity Framework Core. Een bachelorproef geschreven door Nathan Aertgeerts, student Elektronica-ICT aan Artesis Plantijn Hogeschool te Antwerpen. Door middel van voorgaande technologie stack creëren we een portaal waarop de klant zichzelf kan helpen. Het voorgaande onderzoek was bepalend voor de noodzakelijke features die een plaats hebben gekregen in de analyse. We hebben deze features opgelijst in de projectopdracht en vervolgens te werk gegaan in die volgorde.

Het ticketingsysteem met email integratie staat de klant toe om eenvoudig via het webportaal of mail een ticket aan te maken. Support medewerkers kunnen vervolgens op deze tickets antwoorden en klanten worden op de hoogte gehouden van wijzigingen met behulp van een mail. 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. Met behulp van de customer license feature zullen de support medewerkers bepaalde contractlevels kunnen koppelen aan bedrijven en klanten zodat gepersonaliseerde support mogelijkheden toepasbaar zijn.

Vervolgens zal dit uitgebreid worden met een knowledgebase en document management systeem waarbij geregistreerde klanten toegang hebben tot verschillende documenten en downloads. We willen de gebruikers informeren van nieuwigheden en veranderingen door notificaties te tonen wanneer nodig. Een optionele feature is het implementeren van een forum waarop de community en developers elkaar kunnen helpen.

ii. Dankwoord

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

Ik wil graag Intation bedanken voor de fijne samenwerking. In het bijzonder mijn stagebegeleider Baetens Cedric voor de professionele begeleiding en technische ondersteuning om dit project tot een succesvol einde te brengen.
Nico Van Hoorenbeke en Daan Roks voor de fijne samenwerking en alle kansen 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 fijne 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 consulting 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 geoutsourced, draait op wordpress en bevat een klantenzone. Dit is een eenvoudig contact formulier waarbij bestaande klanten vragen kunnen stellen over één van de bestaande producten. Wanneer het formulier is ingevuld komt dit in de Intation-mailbox terecht, daarna wordt dit verder behandelt vanuit de mailbox.

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

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

1.3 De opdracht

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

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

Technisch

In dit hoofdstuk beschrijven we de grondige analyse van het probleem alsook een gedetailleerde oplossing ervan. Specifieke termen zullen kort beschreven worden zodat de kern duidelijk te begrijpen is. Voor een uitgebreide beschrijving zal worden doorverwezen naar de appendices.

Dit hoofdstuk is onderverdeeld in volgende secties:

Analyse

In deze sectie wordt het onderzoek verduidelijkt.

Design

Het design licht toe hoe het project is onderverdeeld in features zodat we een inzicht verkrijgen van de structuur.

Implementatie

De implementatie is een uitgebreide beschrijving van elk functioneel blok in het project. We beschrijven wat, waarom en hoe deze funtionaliteit geïmplementeerd is. Dit is aangevuld met de problemen die zijn ondervonden en de resultaten die behaald zijn.

Analyse

Onderwerp

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

Om mijn onderzoek te starten ben ik na advies van Daan Roks, de productowner, opzoek gegaan naar reeds bestaande oplossingen of 'tools' op de markt. De bestaande helpdesk tools bieden zeer veel features aan en zijn in grote lijnen met elkaar te vergelijken. De grootste spelers op de markt zijn aanhangsels van andere CRM of teamsupport aanbieders zoals Salesforce of Atlassian. Waardoor een ideale implementatie van deze software vaak steunt op deze aanbieder. Aangezien Intation gebruik maakt 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. 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.

Features van bestaande producten

Het was echter wel noodzakelijk de features van de grote spelers op te lijsten om nadien te kunnen filteren op noodzaak van Intation. De grote 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 met een voorkeur naar een MIT-license. De vele open source mogelijkheden vereisten natuurlijk dat de database zelfstandig gehost wordt zodat zij zelf geen service-kosten hebben. Al snel was het duidelijk dat het grootste aantal open source mogelijkheden geschreven is op een basis van PHP. *Aangezien het customer service portaal moet aansluiten bij de bestaande infrastructuur is er geen voorkeur voor PHP.*

Ik ben vervolgens de mogelijkheden gaan bekijken die aansluiten bij de huidige website. Wordpress biedt een tal van plug-ins om digitale downloads te verkopen, helpdesk-systeem toe te voegen of een knowledgebase op te stellen. 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. Na zoveel onderzoek raak je soms de weg kwijt. Het was daarom noodzakelijk om samen te zitten met de productowner en de stagementor. Met de resultaten van mijn onderzoek gebundeld in een excel- en worddocument hebben we een eerste meeting gehouden. Tijdens deze meeting heb ik alle mogelijke oplossingen overlopen en eventuele 'uitblinkers' heb ik uitgebreid laten zien met een demo.

Op basis van de analyse en demo's hebben we tijdens de vergadering de richtlijnen van het project proberen te definiëren. We hebben de gewenste features opgelijst en gequoteerd op noodzakelijk of optioneel.

Extra

Na deze meeting heb ik het onderoek verder gezet. Gebasseerd op bovenstaande features kwamen nieuwe termen naar boven als SLA = servicelevel agreement, ITAM = IT-asset management, ITSM = IT service management. De zoekresultaten scheiden zich van de vorige doordat de reeds bestaande softwarepakketten totaal andere features en focus aanbieden dan de standaard help- of servicedesk oplossingen. De software oplossingen zijn veel meer business gericht en kunnen specifiekere taken aan. Het ticketing-systeem wordt hier echter wel meer naar achteren geschoven en dit was voor de productowner de belangrijkste feature.

De conclusie van de analyse:

Er zijn verschillende oplossingen mogelijk: Bestaande helpdesk-tools, servicedesk-tools, wordpress plug-ins en IT-software Management-tools die voor vele features oplossingen bieden of gecombineerd kunnen worden om een resultaat te bekomen. De eerste sprint werd gespreid over twee weken en was gefocust op onderzoek.

Aan de start van de eerste sprint dacht ik dat twee weken onderzoek overdreven was, maar ik zou nog veel meer tijd kunnen steken in verschillende tools te anaylseren. Het was zeker een leerrijk proces, voor zowel mezelf als Intation. We hebben features ontdekt, ondervonden en getest. We hebben bestaande tools op de markt gevonden die al dan niet geschikt zouden zijn, maar vooral een goede achtergrond opgebouwd om zelf een project te starten. We hebben gekozen om een asp.net core (2.0) met Entity framework te gebruiken omdat dit sterk aanleunt bij de bestaande stack van Intation

Het resultaat:

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

Design

Na de analyse was het noodzakelijk om het project op te delen in features zodat we een structuur creëren. De features bestaan uit funtionaliteit die de productowner gewenst had.

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 overzichtig te kunnen plannen en beheren.

Hiervoor werd een apart project aangemaakt waarin mijn code en sprint planning werd uitgewerkt. Als eerste zijn er volgende features aangemaakt:

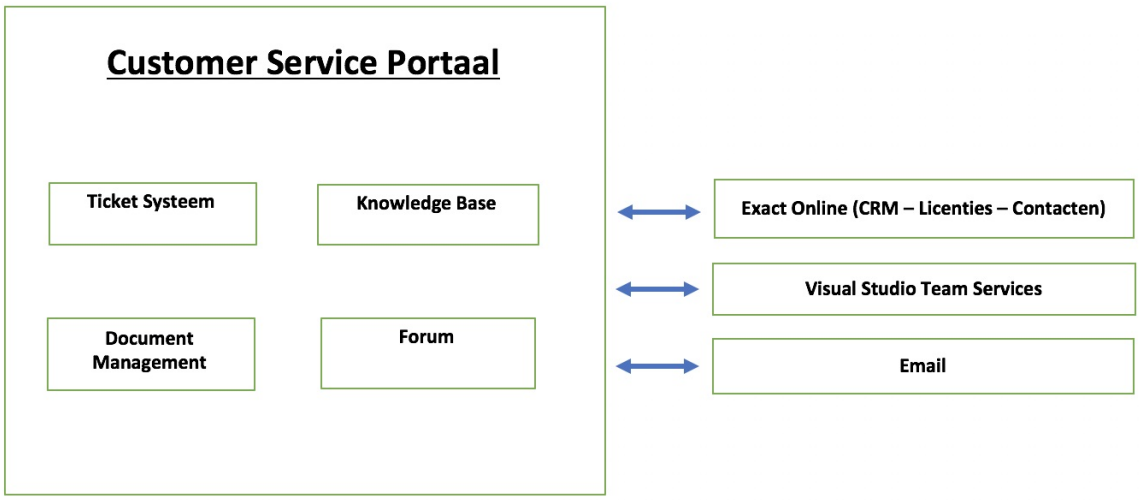
- Ticket Systeem
- Email Integratie met ticket systeem
- Integratie met VSTS
- Integratie met Exact Online
- Knowledge base met informatie
- Downloads en documenten voor geregistreerde gebruikers
- Notificaties voor updates, nieuws, etc

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 ontwikkelingstools in verschillende programmeertalen computerprogramma's te ontwikkelen of code te schrijven. Visual Studio en Visual Studio Team Services kunnen eenvoudig gekoppeld worden. Dankzij het ingebouwde versie controle systeem van VSTS hebben we altijd een back up van onze broncode bij de hand.

Design

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



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

Implementatie

Hier beschrijven we de volledige implementatie van het design blok

Customer Service Portaal

Basis

De basis voor ons 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 via een bestaand emailadres. Nadien kunnen we de gebruikers en rollen permissies opleggen. Om de opstart van onze applicatie te vereenvoudigen en vullen met data moeten we deze Initializeren bij de opstart van het project.

Implementatie

Wanneer je in Visual Studio een project aanmaakt heb je de mogelijkheid om gebruik te maken van een set basistools, hierdoor is het toevoegen van individuele useraccounts een klein werk.

Het is vanzelfsprekend dat een gebruiker zich moet verifiëren op het portaal met een bestaand emailadres. Hiervoor was het noodzakelijk om een mailingservice op te stellen. Ik heb eerst in de AppSettings.json de specifieke email informatie geïmplementeerd voor de SMTP server van mijn privé email adres (Gmail). Daarna heb ik een EmailSettings.cs klasse aangemaakt die deze gegevens ophaalt. Deze klasse vervolgens toegevoegd aan de startup.cs van het project voor dependecy injection. De noodzakelijke Email interface is reeds voorzien in het asp.net core project. Als volgende heb ik de EmailSender geconfigureerd. De noodzakelijk gegevens opvragen zoals het emailadres dat wordt opgevraagd, het onderwerp van de mail, de naam van de zender van de email. Deze gegevens zijn noodzakelijk om de emailconfiguratie tot een goed einde te brengen. Als laatste heb ik de EmailSender klasse geïnjecteerd in de AccountController om beroep te kunnen doen op de SendMailAsync methode ervan. Natuurlijk werkt deze implementatie nooit out of the box, debugging was dus vereist. Na onderzoek kwam een veel voorkomende fout dat Gmail inlogpogingen blokkeert van apps of apparaten die oudere securityinstellingen gebruiken. De oplossing hiervoor was deze instellingen aanpassen zodat minder beveiligde apps zou toestaan. Helaas kwamen er nog steeds geen mails binnen bij het registreren. Met debuggen in visual studio had ik snel de fout gevonden, de smtp service had ik verkeerd ingesteld op smtp.live.com en voor gmail accounts is dit smtp.gmail.com.

Hoe kunnen we gebruikers verschillende rollen toekennen? Dankzij de ingebouwde rolemanager van ASP.NET kunnen we eenvoudig rollen aanmaken en gebruikers hieraan toekennen. Het was even experimenteren en onderzoeken welke functionaliteiten er allemaal beschikbaar zijn. Ik heb vervolgens gekozen om 3 soorten rollen aan te maken: Admins, Agents en Customers. Vanzelfspreken kan een admin rol alle functionaliteit uitoefenen die beschikbaar is op het webportaal. Een Agent is beperkt in het verwijderen en aanpassen van zaken maar kan alles bekijken. Een customer is nog beperkter en kan enkel en alleen zijn eigen tickets bekijken. Ik heb nadien een dashboard aangemaakt voor agent en admin waarin zij een overzicht krijgen van belangrijke zaken en binnenkort instellingen kunnen aanpassen. 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 is. Hiervoor heb ik een DBInitializer klasse aangemaakt die controleert of er al data in de database zit.

- Autorisatie/permissies....
- Code voorbeelden
- ....

Waarom

- Waarom deze aanpak?

Resultaat

- Screenshots Applicatie
- Bespreking van het resultaat

Ticket Systeem

Wat is een ticket systeem

- Wat is een ticket systeem?

Gewenste functionaliteit

- Wat waren de user stories opgesteld voor deze feature

Implementatie

Om van start te gaan met ticket objecten maken we gebruik van basis CRUD-operaties: create, update, read en delete voor een ticket object. Aangezien we deze acties op verschillende plaatsen willen aanroepen zullen we deze samen bundelen in een ticket service. Om data te kunnen meegeven vanuit de razor pages moeten we invoervelden voorzien binnen een form zodat we deze met behulp van een button en databinding de data van de ingevoerde velden kunnen doorgeven aan onze backend. Bij het updaten van een ticket vragen we eerst het gekoppelde ID op en met behulp van dit ID kunnen we het juiste ticket object opvragen en eventueel bewerken. Dezelfde flow wordt gevolgd bij het lezen of verwijderen van een object uit de database. Om functionele redenen en op aanraden van Cedric hebben we de status en prioriteiten als Enums opgeslagen. Waarom? Omdat deze variabelen maar uit een kleine set van mogelijkheden kan bestaan, zo kunnen er al geen fouten via strings inkruipen. Als volgende heb ik een view gecreëerd waarbij we een lijst van tickets opvragen afhankelijk van hun status. Om tickets nog eenvoudiger te kunnen behandelen hebben we na overleg geopteerd om de bestaande producten van intation toe te voegen als enum. (InControl, InWave, InDocumentation)

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

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

- code
- ...

Resultaat

- Screenshots
- Besprekingen resultaat

Knowledge Base

- Een knowledge base is ....

Gewenste Features

- Artikels en Categorieën toevoegen en beheren

Implementatie

- ... CRUD
- ... Code

Waarom

- ...

Resultaat

- ...

Document Management Systeem

- Een document management systeem is ...

Gewenste funtionaliteit

- Documenten toevoegen
- Beperkingen instellen op documenten
- Downloads voor klanten (individueel en mappen)

Implementatie

- code ....

Waarom?

- ...
- ...

Resultaat

- ...
- Screenshots

Notificaties

- Meldingen van updates

Gewenste functionaliteit

- meldingen via email van updates

Implementatie

- hoe ..?

Waarom

- ...

Resultaat

- ....

Communicatie met Externe Services

Email Integratie

Wat?

- ontvangen van emails en emails versturen als antwoord
- Emails uitlezen en belangrijke data eruit halen

Gewenste funtionaliteit

- ontvangen van emails en emails versturen als antwoord
- Emails omzetten in tickets

Implementatie

De mail integratie bij het ticket systeem, dit betekent wanneer er een mailtje binnenkomt deze data kan omgevormd worden naar een nieuw ticket. Hiervoor hebben we een nieuw mailaccount aangemaakt via office 365 (support@intation.eu). Als eerste heb ik de mailing service die de confirmatiemails stuurt overgezet van mijn privé account naar het nieuwe mailadres. Nu moest ik onderzoeken welke mogelijkheden er zijn om inkomende mails op dit account te verwerken. Er zijn verschillende Nuget packages beschikbaar die deze functionaliteit ondersteunen. Ik heb gekozen voor Mailkit/Mimekit een Nuget Package met een MIT-licentie en een hoge rating aangezien deze beschikt over een goede documentatie en makkelijk uit te breiden is. Als eerste slaan we de nieuwe inkomende mails op in de huidige directory als .EML-bestand. Ik heb hiervoor een nieuw model aangemaakt in de database om deze mails te kunnen inlezen en de jusite data eruit te halen die we nodig hebben. Al de voorgaande functionaliteit heb ik gebundeld in een een MailReceiver Service. De Applicatie beschikt nu over een mailbox waarin de gebruiker alle inkomende mails op dit adres kan bekijken. Aangezien niet altijd alle mails automatisch een ticket moeten genereren heb ik ervoor gekozen om dit aan een manuele actie te koppelen en dus de agent of administrator beslist of de mail wordt geconverteerd naar een ticket waarbij de gegevens automatisch worden overgedragen naar het ticket. De administrator kan wel nog opteren om deze informatie te wijzigen en de oorspronkelijke mail al dan niet te verwijderen. Wanneer er een ticket wordt aangemaakt wordt er gekeken wie de oorspronkelijke zender van de mail is. Als deze persoon nog niet geregistreerd is zal hij een mail ontvangen om zijn account te verifiëren, nadien en in alle andere gevallen krijgen de aanvragers van een ticket een bevestigingsmail dat er een ticket is aangemaakt. Na wat finetunen kon ook deze feature op done worden gezet.

Waarom?

- ....

Resultaat

- ...

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. 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.

Gewenste funtionaliteit

- Bugs en features aanmaken en bekijken
- Koppelen aan tickets

Implementatie

Integratie met Visual Studio Team Services in het ticket systeem. VSTS beschikt over een uitgebreide en goed gedocumenteerde API-bibliotheek waar ik mezelf eerst moest in verdiepen. Vervolgens hebben we samen de gewenste functionaliteiten samengesteld gebaseerd op de mogelijkheden met behulp van de API’s. Vervolgens hebben we in VSTS een test project aangemaakt om hierin te kunnen experimenteren met verschillende API-calls. Als eerste moeten we API-token aanvragen om op een veilige manier GET en POST Request te kunnen sturen met onze VSTS account. Ten tweede heb is het antwoord van een API-call een JSON, we moeten dit omzetten in een object zodat we deze data kunnen uitlezen en toekennen. Met behulp van de NewSoft JSON library kunnen we de JSON deserializen in een object en deze data opslaan in de database. Vervolgens hebben we een POST Request geschreven om een bug of feature te creëren. Deze functionaliteit hebben we ook gebundeld in een service. Het is ook van belang om een link te kunnen leggen tussen een ticket en een bug of feature. Hiervoor hebben we een bug model aangemaakt en het ticket model aangepast zodat deze objecten van een bug kan bevatten. Wanneer een koppeling wordt gemaakt wordt de database het ticket ID opgeslagen bij de bijhorende bug zodat we deze eenvoudig kunnen opvragen en laten zien bij het juiste ticket. De API-calls heb ik eerst getest via postman en nadien met behulp van een C# voorbeeld van VSTS geschreven in C# in het asp.net project. Om de feature ook gebruiksnut te geven hebben we ervoor gezorgd dat de bugs worden weergegeven bij het ticket met hun onderwerp, ID, ‘Assigned To’ en door hier op te klikken eenvoudig genavigeerd kan worden naar de juiste VSTS-bug of feature. De retournerende informatie in het bug object was niet altijd even duidelijk waardoor ik bepaalde strings gesplit heb in C# met behulp van de split methode om relevante data duidelijker weer te geven op de razorpages. Als je op verschillend bugs klikt moet je ook het project meegeven wat verschillend kan zijn, dit heb ik opgelost door dit op te vragen wanneer een bug of feature wordt aangemaakt. Het kan ook zijn dat een ticket betrekking heeft tot een reeds bestaande bug of feature. Daarvoor heb ik de optie voorzien om deze manueel te koppelen door het ID van een bestaande bug mee te geven. Voorlopig was mijn eigen API-token manueel geconfigureerd in de C# code. Omdat elke Agent een verschillende API-token heeft moeten we dit variabel kunnen instellen. In het dashboard heb ik daarom een invoerveld voorzien waarbij we een token kunnen invoeren en koppelen via de include functionaliteit van ASP.NET Core aan onze ApplicationUser. We willen ook dat de gebruiker zijn API-key kan verwijderen wanneer gewenst en een nieuwe toe te voegen wanneer deze vervallen is. Ik heb dit efficiënt gemaakt door de in de UI links toe te voegen waardoor het voor de gebruiker eenvoudig is om te navigeren naar de instellingen.

Waarom?

- ...

Resultaat

- Screenshots
- ...

Exact Online Integratie

- Wat is exact Online?

Gewenste functionaliteit

- Refreshen van data

Implementatie

Een volgende Feature was de integratie met Exact Online, dit is een online CRM- en boekhoudingtool voor bedrijven en zelfstandigen om eenvoudig je gegevens, facturen en alle andere te beheren. Exact Online kan ook uitgebreid worden met verschillende modules. Er was eerst gevraagd om te kijken naar API-mogelijkheden aangezien deze documentatie ook beschikbaar is op de website van Exact Online. Helaas kan een gegenereerde token geen verschillende beperkingen opleggen dan het account waarop deze gecreëerd is. Intation beschikt maar over enkele accounts bij exact online waarop de leidinggevende toegang tot alles hebben. Aangezien dit wil zeggen dat ik via de API-calls dan toegang heb tot alle vertrouwelijke data en deze ook kan verwijderen hebben we beslist om geen token te creëren met volledige toegang voor testing/development purposes. Het aanmaken van eender welke extra account met al dan niet eventuele beperkingen resulteert in een maandelijkse extra fee voor deze account. Om deze extra kost te vermijden bij het ontwikkelen van een applicatie hebben we geopteerd om gebruik te maken van de exportfunctie van Exact Online naar een XML of CSV-file. De data die we hieruit verkrijgen is te vergelijken met de data die we van een API-call zouden terugkrijgen. Waardoor we onze functionaliteit verder kunnen uitwerken, maar gebruik maken van een statisch dataformaat dat we handmatig kunnen updaten. Het XML-bestand bevat veel klanteninformatie dat we moeten filteren op noodzaak in onze applicatie. Hiervoor hebben we het XML-bestand onderzocht, dit bestand bevat Accounts, deze Accounts zijn bedrijven of klanten en hebben soms een lijst van contact objecten. ASP.NET Core heeft een ingebouwde functionaliteit om XML-bestanden te parsen. Ik heb een object aangemaakt dat de data van deze file kan opslaan en vervolgens kunnen we een lijst weergeven van alle bedrijven. Om het XML bestand te uploaden zetten we dit eerste om in een ByteArray met de IFormFileExtension, deze extensie is vrij eenvoudig en zorgt ervoor dat we via databinding onze file kunnen uploaden in de razor pages en vervolgens omzetten in een ByteArray om op te slaan, echter kunnen we deze array niet zomaar uitlezen en zetten we deze om in een stream, deze stream valt perfect uit te lezen en data op te slaan in onze database. Het bestand wordt vervolgens geserialized naar een JSON object zodat dit overeen zou komen met de data van onze API-calls, deze gaan we dan deserializen om vervolgens onze data aan ons object toe te kennen en op te slaan in de database. Dit JSON object is ook eenvoudiger om mee te werken dan het XML-bestand. Om het bedrijf en zijn data te kunnen koppelen aan een gebruiker heb ik een extra tabel aangemaakt die het companyID en UserID opslaat. Dit valt te vergelijken met hoe de rollen worden opgeslagen in ASP.NET Core. Vervolgens heb ik ervoor gezorgd dat we deze data manueel kunnen koppelen door een button actie en de juiste informatie kunnen weergeven bij de tickets door het opvragen van de gebruikers in het bedrijf object.

Waarom.

- ....

Resultaat

- ....

Conclusie

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

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

Appendices

ASP.NET Core 2.0

Entity Framework Core

Razor Pages

Bootstrap

Bibliografie

Algemeen

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

API's

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

Mail

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

term1

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

gitbook

Verwijder uiteraard ook deze term in je glossary, maar aanschouw hier toch het resultaat.