AFSTUDEERVERSLAG

Office 365 integratie voor MKG

Rémi Dijkstra
470172@student.saxion.nl

Voorwoord

U leest het afstudeerverslag over de integratie van Office 365 voor MKGV van MKG. Dit project is uitgevoerd in de periode september 2023 – februari 2024 en is bedoeld voor MKG (Metaal Kennis Groep).

In de eerste fase van het project wordt een onderzoek uitgevoerd dat aandachtspunten identificeert en ontwerpkeuzes beïnvloedt. Aan de hand van dit onderzoek wordt een prototype gebouwd. In dit verslag worden het proces en de bijbehorende resultaten beschreven. Tot slot worden aanbevelingen gegeven en wordt gereflecteerd op het project en de geleverde resultaten.

Samenvatting

Voor het laatste deel van mijn opleiding heb ik een afstudeerstage bij MKG in Hengelo (OV) doorlopen. De stage heeft geduurd van 4 september 2023 tot en met 9 februari 2024. MKG ontwikkelt ERP (Enterprise Resource Planning) software voor de metaalindustrie, genaamd MKGV. Deze software ondersteunt bedrijven bij het organiseren van diverse bedrijfsprocessen, waaronder inkoop- en verkooporders, het maken van offertes, activiteitenplanningen. Daarnaast zijn er specifieke processen voor de metaalindustrie die makkelijker gemanaged kunnen worden met de ERP software, zoals onder andere productieorders, materiaalvoorraad, artikelen.

MKGV beschikt ook over een Documentbeheersysteem dat bestanden opslaat op een opgegeven locatie. Momenteel kunnen alleen locaties worden opgegeven die toegankelijk zijn voor het besturingssysteem, zoals lokale folders op een computer of een gedeelde harde schijf. MKG wil echter graag dat bestanden online in de cloud worden opgeslagen, zodat klanten er altijd en overal toegang toe hebben. Bovendien willen ze dat de beveiligingsmaatregelen die in MKGV zijn geïmplementeerd, worden behouden in de cloudomgeving.

Naast bestanden willen ze ook informatie, zoals contacten en kalenderitems, online opslaan en synchroniseren met Azure. Tot slot willen ze e-mails niet langer via de Outlook-app ophalen, maar direct via een API in MKGV laden.

Voor de ontwikkeling van deze functies is onderzoek nodig naar hoe deze informatie kan worden gesynchroniseerd en hoe dit veilig kan worden gedaan. Ook moet worden onderzocht hoe online opgeslagen bestanden gemakkelijk doorzoekbaar kunnen zijn.

Na het onderzoek is er een ontwerp opgesteld dat aangeeft hoe de nieuwe functionaliteit in MKGV zal worden geïmplementeerd en welke aanpassingen aan de database nodig zijn om de extra gegevens op te slaan.

Het ontwikkelen is begonnen, en er is eerst een Proof of Concept (POC) opgezet om te demonstreren dat de gewenste functionaliteit haalbaar is. Na afronding van POC A is de implementatiefase gestart, gebaseerd op de POC en de ontwerpen. Hierbij zijn onverwachte problemen opgetreden.

Het projectmanagement volgde de Scrum Agile methode, met sprints van 4 weken waarin hard werd gewerkt aan de nieuwe functionaliteit. Tools zoals Postman, Notepad++ en Progress Developer Studio werden gebruikt.

Aan het einde van de stage was alleen het uploaden van bestanden naar de cloud voltooid en opgeleverd. Dit terwijl er in de opdrachtomschrijving meer elementen stonden die voltooid hadden moeten worden. Het synchroniseren van andere data was gestart, maar er was niet genoeg tijd om het volledig te implementeren.

Inhoud

1. Inleiding

1.1. MKG

MKG ontwikkelt een ERP (Enterprise Resource Planning) pakket voor de metaalindustrie, genaamd MKGV. Dit pakket wordt door klanten gebruikt om bedrijfsprocessen te organiseren, variërend van inkoop- en verkooporders, het aanmaken van offertes tot activiteitenplanningen, enzovoort. Daarnaast zijn er processen die specifiek zijn voor de metaalindustrie, zoals productieorders, materiaalvoorraad, artikelen, enzovoort.

MKG richt zich uitsluitend op het ontwikkelen en onderhouden van hun ERP-pakket en biedt geen maatwerkoplossingen voor klanten. MKGV kan door de klant zelf worden gehost of door MKG zelf worden gehost.

Voor de ontwikkeling van MKGV wordt Progress OpenEdge gebruikt. Bovendien heeft MKG een eigen framework gebouwd dat ondersteunt bij het verwerken van data, het opbouwen van interfaces en het uitbrengen van updates voor MKGV. Dit alles is gebouwd met een 4GL (Fourth Generation Language) genaamd ABL (Advanced Business Language), ontwikkeld door Progress zelf. ABL is een volwassen taal die OOP-principes ondersteunt, en het stelt ook in staat om gebruik te maken van C#-bibliotheken om de functionaliteit verder uit te breiden.

1.2. Stakeholders Het project heeft de volgende stakeholders:

- Raymond Berning: Raymond Berning is de afstudeerbegeleider vanuit MKG.
- Jochen Dankelmann: Jochen is de product owner bij MKG.
- Etto Salomons: Etto Salomons is de afstudeerbegeleider vanuit het Saxion.
- MKG: Tijdens dit project wordt samengewerkt met MKG om nieuwe functionaliteiten toe te voegen aan hun software.
- Klanten van MKG: De klanten van MKG hebben veel belang bij dit project, omdat het een nieuwe manier toevoegt aan hoe MKGV kan worden gebruikt.

1.3. Startsituatie

MKG levert een ERP (Enterprise Resource Planning) pakket genaamd MKGV aan de metaalindustrie. Dit pakket kan lokaal bij een bedrijf worden gehost of online worden gehost door MKG zelf. MKGV heeft een DMS (documentmanagementsysteem) dat helpt bij het beheren van bestanden die nodig zijn in het ERP-systeem. Deze bestanden worden momenteel lokaal opgeslagen op computers in het bedrijf van de klant en worden dus niet centraal opgeslagen. Dit kan problemen veroorzaken wanneer bestanden nodig zijn die niet op hun lokale schijf staan. Bedrijven kunnen ervoor kiezen om hun bestanden zelf centraal op te slaan, maar MKG wil MKGV toegankelijker maken door de centrale opslag te integreren met behulp van SharePoint.

Daarnaast willen ze een vernieuwde koppeling met Microsoft Office, zodat er meer wordt gesynchroniseerd met Office 365, dus niet alleen de e-mail, maar ook de agendapunten en contactgegevens. Momenteel is er een koppeling tussen Outlook en MKGV die e-mails binnenhaalt in MKGV en ook e-mails kan versturen. Deze moet worden omgezet naar Office 365, zodat het een directere koppeling is en Outlook geen tussenstap meer is. Ook moet er een synchronisatie komen tussen ERP-activiteiten (kalenderitems) en Office 365, zodat wanneer klanten activiteiten maken in MKGV, deze ook in hun Office 365-kalender verschijnen. Ten slotte moet er nog een synchronisatie komen tussen ERP-contacten en Office 365-contacten.

Al deze synchronisaties moeten zo generiek mogelijk zijn, zodat ze gemakkelijk kunnen worden vervangen door een andere officesuite. Er is al enige kennis over Office 365-API's; MKGV heeft al

functionaliteiten voor het aanmaken, aanpassen en verwijderen van activiteiten en contacten. Ook is er een basisintegratie tussen Outlook en MKGV.

1.4. Probleemanalyse

Om de nieuwe integratie van Office 365 met MKGV tot stand te brengen, zijn er een aantal stappen die moeten worden uitgevoerd. Allereerst moet er onderzoek worden gedaan naar hoe een Office 365-account kan worden gekoppeld aan een ERP-account. Hierbij is het van belang om de mogelijkheden die Microsoft biedt te verkennen en de veiligste aanpak te bepalen. We gaan ervan uit dat alle gebruikers een Office 365-account hebben.

Ten tweede moet het mogelijk worden gemaakt om documenten te uploaden en downloaden van en naar SharePoint via MKGV. Hierbij gaat het met name om de essentiële bestanden die centraal worden opgeslagen, wat de hoogste prioriteit heeft voor MKG.

Daarna volgt de implementatie van de nieuwe e-mailkoppeling met Office 365. Er is al een interface in MKGV hiervoor aanwezig. Hierbij wordt rekening gehouden met de huidige werkwijze van MKG rondom e-mails in het ERP-systeem.

Als laatste stap komt de synchronisatie tussen activiteiten en contacten uit het ERP-systeem naar Office 365. Hierbij wordt prioriteit gegeven aan de data uit MKGV boven die van Office 365. Ook wordt rekening gehouden met mogelijke veranderingen in de database om het synchroniseren van de data te vergemakkelijken. De synchronisatie zal waar mogelijk real-time zijn, en er wordt de optie geboden om de data op bepaalde momenten te synchroniseren.

Dit gehele proces wordt ontwikkeld in OpenEdge ABL, een 4GL. OpenEdge ABL is een programmeertaal die in 1981 is uitgebracht voor niet-ontwikkelaars om zakelijke apps te maken. Het wordt veel gebruikt in de Verenigde Staten, met name bij financiële bedrijven. MKG gebruikt het omdat er binnen MKG veel expertise is en omdat het eenvoudiger is om met grote hoeveelheden data te werken.

Hieronder volgt een kort overzicht van de activiteiten die moeten worden uitgevoerd:

- Mogelijkheid bieden om een Office 365-account aan een ERP-account te koppelen.
- Het mogelijk maken om bestanden te uploaden en downloaden van/naar SharePoint.
- Een nieuwe koppeling maken tussen ERP en Office 365 voor e-mails.
- Synchronisatie tussen ERP-activiteiten (kalenderitems) en Office 365 tot stand brengen.
- Synchronisatie tussen ERP-contacten en Office 365 realiseren.
- Prioritering van real-time synchronisatie voor belangrijke data implementeren.

1.5. Doelstelling

Er moet een nieuwe integratie komen tussen MKGV en Office 365 met daarin een nieuwe koppeling tussen MKGV en office 365 voor e-mails en een synchronisatie voor data uit het ERP systeem zoals activiteiten, contacten en bestanden naar Office 365.

2. Vooronderzoek

2.1. Progress OpenEdge

Voor het onderzoek naar Progress OpenEdge heb ik vooral gebruikgemaakt van de cursussen die Progress zelf aanbiedt op hun Progress Education Community-website. ProgressO OpenEdge was voor mij nog onbekend daarom ben ik begonnen met de basiscursus "Introduction to Progress OpenEdge" en ben vervolgens doorgegaan met de cursus "Introduction to Progress OpenEdge Integration" om een idee te krijgen van hoe een normaal Progress OpenEdge-systeem eruitziet. Nadat ik die cursussen heb gevolgd, heb ik de cursussen "ABL Essentials" en "Introduction to Object-Oriented Programming" gedaan om kennis op te doen over ABL en hoe Progress OpenEdge OOP implementeert.

Om mijn kennis van ABL te vergroten, heb ik ook delen van de documentatie gelezen om meer te begrijpen over hoe ik .NET-bibliotheken of C#-functies kan gebruiken in ABL. Ook heb ik meer gelezen over hoe arrays en lijsten werken in ABL. Naast de cursussen en documentatie heb ik ook zelf wat code geschreven om te zien hoe Progress OpenEdge werkt. Deze code varieert van kleine tests om data uit databases te lezen tot grotere tests waarmee ik via een API-request gegevens ophaal en wegschrijf naar een bestand. Ook heb ik tests geschreven om te kijken hoe ik gebruik kan maken van .NET-bibliotheken in Progress OpenEdge. Als laatste heb ik kleine scripts gemaakt die delen van de functionaliteit uitvoeren die ik moet implementeren.

Wat ik heb geleerd over ABL is dat het vooral een 4GL-taal is als je bezig bent met data uit de database. Als je niet bezig bent met de database, is het meer zoals een normale programmeertaal. Eerdere versies van Progress waren meer een scripttaal waar je procedurebestanden had met procedures die je uitvoerde. In de laatste versies zijn ze meer overgegaan naar een objectgeoriënteerde taal met ondersteuning voor klassen, interfaces, enums, objectovererving, polymorfisme en delegatie. De structuur is echter anders dan ik gewend ben. Als je een functie in een klasse wilt aanroepen, doe je dat door een object van de klasse op te geven en door middel van een dubbele punt de functie aan te roepen (klasse:functie). Een regel code beëindig je door een punt te plaatsen. In plaats van accolades om een codeblok aan te geven, gebruik je voor if, while en for aan het begin de tekst do, en om het codeblok te beëindigen gebruik je de tekst end. Met deze opgedane kennis hoopte ik een goede basis te hebben om aan de slag te kunnen met de software van MKG

2.2. MKGV

Nadat ik een basis opgebouwd had in de taal waarin gewerkt wordt, moest ik kennis krijgen van de structuur van MKGV om daarin te kunnen werken. Voor het vooronderzoek naar MKGV ben ik begonnen met een paar vergaderingen met mijn begeleider waarbij ik vragen heb gesteld over hun framework, de code structuur, multi tenancy, multi threaded support en de OOP structuur die MKG aanhoudt. Ook heb ik uitleg gekregen over hoe het debuggen van de software werkt en hoe het huidige document beheer systeem werkt. Verder heb ik zelf onderzoek gedaan door documenten toe te voegen in MKGV en te debuggen welke code allemaal wordt gebruikt. Met kennis van zowel OPenEdge als de structuur van MKGV had ik een goed beeld van de omgeving en resources waarmee ik ging werken.

2.3. Microsoft authenticatie

Het vooronderzoek naar de Microsoft Identity Platform bestond voornamelijk uit het lezen van documentatie die door Microsoft is opgesteld. Daarnaast heb ik veel vragen van andere ontwikkelaars op Stack Overflow gelezen als aanvulling op de verouderde documentatie van

Microsoft. Ten slotte heb ik zelf requests uitgevoerd via Postman om te onderzoeken hoe de volledige flow van Microsoft verloopt en waar ik op moet letten.

2.4. Microsoft SharePoint

Mijn vooronderzoek naar SharePoint begon met een vergadering met een collega die al kennis heeft van SharePoint. In deze vergadering heb ik hem vragen gesteld over het uploaden van bestanden, het toevoegen van tags aan bestanden, zoeken op basis van tags, het delen van bestanden en aanvullende beveiligingsmogelijkheden die ik kan implementeren. Zelf heb ik ook onderzoek moeten doen naar het gebruik van tags in SharePoint en hoe deze kunnen worden ingezet voor zoekfunctionaliteiten.

MKG heeft mij toegang verleend tot een Microsoft-tenant die zij gebruiken voor ontwikkelingsdoeleinden, zodat ik kan experimenteren met SharePoint. Daarnaast heb ik extra onderzoek gedaan naar beveiligingsgroepen in SharePoint om bestanden af te schermen. Ten slotte heb ik onderzoek verricht naar de SharePoint Content Organizer om te beoordelen of deze kan helpen bij het organiseren en sorteren van documenten.

2.5. Microsoft Graph

Het vooronderzoek van Microsoft Graph was vooral gefocust op het bekijken wat er mogelijk was met Graph en hoe de Documentatie was. Ik ben begonnen met kijken naar de Graph Explorer die Microsoft aanbiedt, Graph Explorer laat alle API paden van de Graph API zien en linkt naar de correcte documentatie voor elk pad. Ik ben begonnen met onderzoeken hoe ik bestanden in SharePoint krijg via Graph, daarna heb ik onderzocht hoe ik het bestand kon delen via Graph. Als laatste heb ik onderzoek gedaan naar security groepen en hoe je die kan aanmaken en leden aan kan toevoegen via Graph.

2.6. conclusie

Uit mijn vooronderzoek blijkt dat de systemen die ik ga gebruiken over het algemeen gemakkelijk te gebruiken zijn. Progress OpenEdge lijkt misschien wat anders dan wat ik gewend ben, maar dankzij de nieuwe objectgeoriënteerde functies voelt het toch veel als programmeertalen die ik ken.

MKGV daarentegen blijft voor mij nog een beetje een raadsel. Het is omvangrijk, met bestanden die soms duizenden regels code bevatten. Het helpt ook niet dat er niet veel documentatie beschikbaar is voor het framework.

Voor de Microsoft-authenticatie heb ik voornamelijk onderzoek gedaan naar wat Microsoft aanraadt voor het inlogproces. Ik heb hier niet veel tijd aan besteed, omdat het vooral ging over het verkrijgen van toegangstokens. Wat betreft Microsoft Graph heb ik met name onderzoek gedaan naar de Graph Explorer die Microsoft aanbiedt. Dit was om meer kennis op te doen over hoe het systeem werkt en welke mogelijkheden het biedt.

3. Werkwijze

3.1. Scrum methode

Bij MKG wordt SCRUM gebruikt voor het projectmanagement, waarbij sprints van 4 weken worden gehanteerd. Voorafgaand aan elke sprint wordt bepaald wat er moet worden gedaan, en het werk wordt verdeeld onder de ontwikkelaars. Deze informatie wordt vastgelegd in TopDesk, waar ook kleinere wijzigingen worden geregistreerd die door ontwikkelaars kunnen worden opgepakt. Ik heb de basisstructuur van de scrumblokken gevolgd door werk voor 4 weken in te plannen. Er waren een aantal aanpassingen om het voor mijn afstudeeropdracht makkelijker te maken. Naast de scrum overleggen had ik wekelijks overleg met mijn begeleiders waar ik feedback kreeg, planningen maakte en problemen kon bespreken. Als afstudeerder viel ik ook een beetje buiten de boot omdat ik opdrachten uitvoerde onafhankelijke van mijn collega's en heb daardoor niet erg veel met het scrum systeem kunnen werken. Wat ik er van heb meegekregen vond ik niet onaangenaam.

Tussen twee sprints is er altijd een tussenweek waar ontwikkelaars kennis kunnen opdoen, proof of concepts kunnen maken of zich kunnen voorbereiden op de nieuwe sprint door de backlog door te nemen of technische ontwerpen te maken. Aan het einde van elke sprint worden retrospectieven gehouden om het Scrum-proces te verbeteren.

De sprints worden geleid door de twee scrum masters, Sander Klein Legtenberg en Sybrand Westhuis. Overtijd neemt Sander de positie van Sybrand over. Naast dit alles werken we ook met buddies, waar we om hulp kunnen vragen als iets onduidelijk is of als er problemen zijn.

3.2. Source control

Voor source control maakt MKG gebruik van SVN (Apache Subversion). SVN wordt gekozen omdat het direct is geïntegreerd met de Progress Developer Studio en geen extra software nodig heeft om te functioneren. SVN werkt op een vergelijkbare manier als Git, waarbij een hoofdbranch, genaamd de trunk, wordt gebruikt als startpunt en waar nieuwe branches van kunnen worden gemaakt. Bij MKG worden branches echter niet meer gebruikt; alle wijzigingen worden direct op de trunk geplaatst. Alleen voltooide functionaliteiten of fixes worden naar de trunk gepusht.

Wanneer er een release wordt gemaakt, wordt er een tag toegevoegd aan de trunk om bij te houden welke wijzigingen wel en niet zijn opgenomen in de volgende versie.

3.3. uitvoering

3.3.1. onderzoek

Allereerst wordt er een onderzoek uitgevoerd dat vastlegt wat de opties zijn voor het uitvoeren van het project. Dit onderzoek wordt geïntegreerd met de kennis die is opgedaan in het vooronderzoek om een duidelijk beeld te verkrijgen van hoe het project zal worden uitgevoerd. De bevindingen van dit onderzoek worden ook besproken met collega's om een definitief beeld te vormen van de mogelijkheden.

3.3.2. ontwerpen

De informatie uit de voorgaande fase wordt hier gebruikt om een ontwerp te creëren voor de implementatie. Dit ontwerp omvat een klassendiagram voor de generieke implementatie en de benodigde aanpassingen aan de database. Daarnaast wordt vastgelegd welke data nodig is om een koppeling met Office365 tot stand te brengen.

3.3.3. Prototyping/implementatie

Tijdens de prototyping- en implementatiefase wordt gewerkt aan een Proof of Concept (POC) van hoe de functionaliteit zou moeten werken. Deze POC is gebaseerd op het ontwerp van de vorige fase. Nadat de POC is gemaakt en goedgekeurd, wordt de functionaliteit geïmplementeerd in MKGV. Dit gebeurt op basis van de gemaakte POC en het ontwerp.

4. Requirements

De requirements zijn opgesteld bij de opdrachtomschrijving en tijdens het onderzoek zijn er geen zaken aan het licht gekomen die ervoor zorgden dat ze moesten worden aangepast.

Req	Omschrijving	FR/NFR	Prioriteit
RQ01	Gebruikers moeten data van Azure kunnen invullen om connectie te kunnen maken.	FR	M
RQ02	Gebruikers moeten bestanden kunnen uploaden naar SharePoint.	FR	M
RQ03	Gebruikers moeten doormiddel van categorieën autorisatie kunnen toevoegen	FR	M
RQ04	Gebruikers moeten gemakkelijk bestanden moeten terug kunnen vinden in SharePoint	NFR	M
RQ05	Gebruikers moeten SharePoint bestanden kunnen verwijderen, openen of printen via MKGV	FR	M
RQ06	Gebruikers moeten een SharePoint site en drive kunnen kiezen waar bestanden worden geplaatst	FR	M
RQ07	Data uit MKG moet leidend zijn	NFR	M
RQ08	Groepen en document categorieën moeten periodiek gesynchroniseerd worden	NFR	S
RQ09	Gebruikers e-mails moeten zonder de outlook app kunnen worden opgehaald vanuit MKGV	NFR	C
RQ10	E-mails moeten zonder de outlook app kunnen worden verstuurd vanuit MKGV	NFR	C
RQ11	Contacten uit MKGV moeten gesynchroniseerd worden met Office365	NFR	C
RQ12	Kalender items uit MKGV moeten gesynchroniseerd worden met Office365	NFR	C
RQ13	Kalender items en contacten moeten periodiek gesynchroniseerd worden	NFR	C
RQ14	Documenten die al in het systeem staan moeten ook naar SharePoint gesynchroniseerd worden	NFR	W

Legenda:

FR: Functionele Requirement

NFR: Non-Functional Requirement

Prioriteit: M (Must-have), S (Should-have), C (Could-have), W (Won't-have

5. Onderzoeksresultaten

Om een goede oplossing voor het probleem te kunnen ontwikkelen, is er een onderzoek uitgevoerd. In dit hoofdstuk worden de bevindingen van het onderzoek beschreven.

5.1. Onderzoeksvragen

Het doel van dit project is het synchroniseren van informatie uit MKGV naar Office365 of andere office suites. Dit zorgt ervoor dat de belangrijke informatie van klanten altijd beschikbaar is in een online omgeving. Hierdoor kunnen klanten gemakkelijker online samenwerken of informatie raadplegen als ze geen toegang hebben tot MKGV. Om dit project goed uit te voeren, zijn de volgende hoofdvraag en deelvragen opgesteld.

Hoofdvraag: Hoe kan data van MKGV worden gesynchroniseerd worden met Office365?

Deelvragen

1. Hoe kan een toegangstoken van Microsoft worden verkregen en welke mogelijkheden bestaan er?
2. Hoe kan bescherming vanuit MKGV worden overgebracht naar SharePoint?
3. Hoe kunnen bestanden worden getaged in SharePoint?
4. Welke apis zijn er voor SharePoint/Office?
5. Welke .net libraries zijn er voor Sharepoint/Office?

5.2. Hoe kan een toegangstoken van Microsoft worden verkregen en welke mogelijkheden bestaan er?

Het koppelen van een Office 365-account kan op drie verschillende manieren gebeuren. Voor alle drie de methoden moet je een app registreren in Azure, deze geregistreerde app regelt dan toegang die een gebruiker heeft tot de data in Azure. De eerste methode is door een gebruiker zijn Office 365-gegevens in te vullen in de ERP-software en deze te gebruiken om in te loggen. Hier zijn enkele risico's aan verbonden. Ten eerste moeten we inloggegevens opslaan in de database, zodat een gebruiker niet telkens opnieuw de gegevens hoeft in te vullen in de ERP-software. Het tweede risico is dat we een login uitvoeren voor de gebruiker bij Microsoft, waarbij we de inloggegevens verzenden via een POST-request. Als deze worden onderschept, zijn de gegevens van de gebruiker niet meer veilig. Tot slot geeft Microsoft zelf ook aan dat dit niet de beste manier is om in te loggen voor een gebruiker.

De tweede methode is het tonen van een pop-upscherm waarin je inlogt via Microsoft zelf. Hieruit komt een toegangstoken en een vernieuwingstoken die gebruikt kunnen worden voor verificatie. Het nadeel van deze methode is dat het vernieuwingstoken dat Microsoft aanmaakt, slechts 30 dagen geldig is. Als deze tokens worden onderschept, kunnen ze lange tijd worden gebruikt.

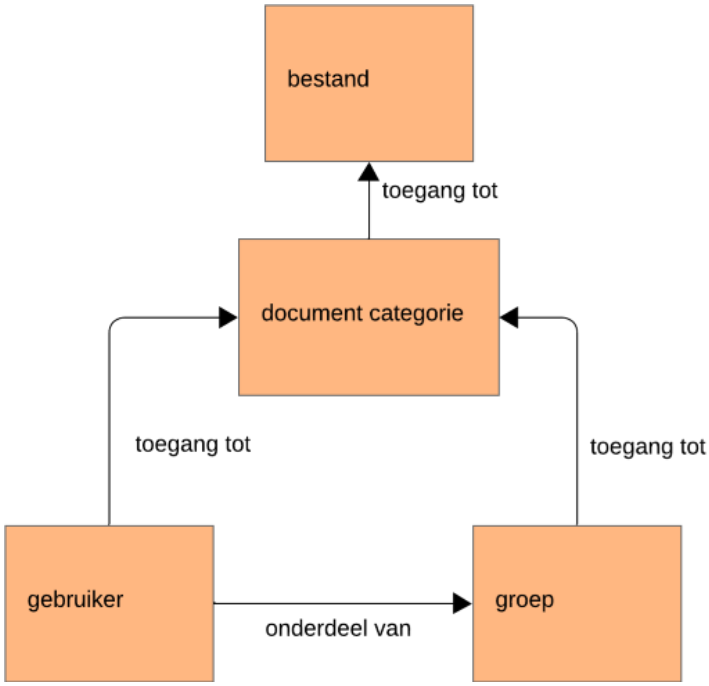
De laatste methode voor het inloggen werkt ook via een pop-upscherm waar je inlogt. Bij deze pop-up krijgen we geen vernieuwingstoken terug, maar wordt er gebruikgemaakt van een sessiecookie die bijhoudt of je ingelogd bent. Deze sessiecookie is één dag geldig en wordt verwijderd zodra de software wordt gesloten. Dit is ook de methode die Microsoft aanraadt om te gebruiken. Wij raden aan om de sessiecookiemethode te gebruiken, omdat de kans op het verkrijgen van gevoelige data dan kleiner is.

5.3. Hoe kan bescherming vanuit MKGV worden overgebracht naar SharePoint?

Op dit moment functioneert de beveiliging in MKGV als volgt: wanneer je een document toevoegt aan het documentmanagementsysteem, kun je een documentcategorie toewijzen. Deze documentcategorie wordt vervolgens gebruikt om te bepalen wie wel of geen toegang heeft tot het bestand. Aan documentcategorieën kunnen gebruikers of gebruikersgroepen worden toegevoegd die toegang hebben tot bestanden met die documentcategorie.

Om toegang te verlenen tot bestanden in SharePoint kunnen twee soorten groepen worden gebruikt. De eerste is een Office 365-groep, die zijn eigen e-mail, SharePoint-site, Teams-kamer en meer heeft. Dit is allemaal niet nodig voor de beveiliging van bestanden. De tweede soort groep is een beveiligingsgroep. Beveiligingsgroepen zijn ontworpen om gebruikers toegang te geven tot specifieke delen van SharePoint, Azure of Teams. Deze kunnen worden gebruikt om toegang tot bestanden te regelen.

We kunnen dezelfde opzet als in MKGV gebruiken om beveiligingsgroepen op te zetten. We kunnen gebruikersgroepen maken waar gebruikers deel van uitmaken en deze toevoegen aan een documentcategorie. Een documentcategorie kan dan toegang verlenen aan groepen of specifieke gebruikers. Om een bestand te delen, hoeven we alleen toegang tot het bestand te verlenen aan een documentcategorie. Op deze manier heeft iedereen die deel uitmaakt van een groep toegang tot het bestand. Hieronder zie je figuur 1, een diagram dat het bovenstaande uitlegt.



Figuur 1. Overzicht structuur toegang gebruiker tot bestand

5.4. Hoe kunnen bestanden worden getaged in SharePoint?

Deze vraag is belangrijk omdat het taggen van bestanden gaat helpen bij het makkelijker doorzoekbaar maken van de bestanden. Ook zorgt het ervoor dat je in één oogopslag kan zien wie wel en niet toegang heeft tot het bestand.

Het taggen van bestanden in SharePoint is mogelijk door gebruik te maken van aangepaste kolommen in SharePoint. Deze kolommen kunnen aangemaakt worden via het SharePoint Admin Center en vervolgens toegevoegd worden aan verschillende documentbibliotheken. Voor het toevoegen van tags kan gebruik gemaakt worden van 2 typen data: de managed metadata of een simpel tekstveld.

Managed metadata:

Dit datatypen bestaat uit vooraf bepaalde datatags die toegevoegd kunnen worden aan items. Deze tags kunnen van tevoren worden vastgezet door SharePoint-beheerders. Dit voorkomt dat gebruikers per ongeluk verkeerde of niet-bestaande tags toevoegen aan bestanden. Managed metadata heeft ook het voordeel dat het makkelijker te indexeren en doorzoeken is dan een tekstveld. Het enige nadeel aan managed metadata is dat het niet via een API of bibliotheek aan een SharePoint-bestand kan worden toegevoegd.

Tekstveld:

Het tekstveld-datatype is het makkelijkst te gebruiken; je hoeft het alleen maar toe te voegen als kolom en je kan beginnen met taggen. Het nadeel van het tekstveld is dat alle gebruikers de data kunnen wijzigen en dat je niet kan aangeven welk type data is toegestaan. Een ander nadeel is dat als je het doorzoekbaar wilt maken, je extra configuraties in SharePoint moet doen zodat het extra veld ook gebruikt kan worden voor zoekopdrachten. Het voordeel is dat je dit veld wel kan invullen via een API of bibliotheek.

Mijn advies is op dit moment om tekstvelden te gebruiken. Dit advies geef ik omdat er op dit moment geen ondersteuning is in enige SharePoint API of bibliotheek om te werken met Managed Metadata. Het feit dat je zomaar andere data in een tekstveld kan plaatsen is ook niet echt een groot probleem, omdat je eerst meerdere stappen moet nemen om de data aan te passen. Het kan ook zijn dat gebruikers misschien andere data willen toevoegen, en door een tekstveld te gebruiken is dat mogelijk.

5.5. Welke apis zijn er voor SharePoint/Office?

Voor dit project zijn er 3 API's waar gebruik van gemaakt kan worden. De eerste is de SharePoint API. Deze API biedt de mogelijkheid om alle basisfuncties van SharePoint uit te voeren. Ook biedt het de mogelijkheid om geavanceerdere taken uit te voeren, zoals het aanmaken van nieuwe sites, lijsten, documenten of het opnieuw laten indexeren van een site. Microsoft adviseert echter om niet langer gebruik te maken van de SharePoint API en over te stappen naar de v2-versie. De v2-versie is de Microsoft Graph API; deze mist echter nog enkele functionaliteiten van de SharePoint API.

De tweede is de Office 365 API. Deze API biedt de mogelijkheid om informatie uit verschillende Office-apps te lezen, schrijven, aanpassen of verwijderen. Deze API zou gebruikt kunnen worden voor de overige Outlook-functies die geïmplementeerd moeten worden in MKGV. Via de Office 365 API kun je e-mails ophalen, e-mails verzenden, contacten uitlezen, kalenderitems uitlezen en alles doen wat je met Outlook kunt doen. Ook hier adviseert Microsoft om deze API niet te gebruiken en over te stappen naar de v2-versie. De v2-versie is Microsoft Graph, waarin alle functionaliteiten van de Office 365 API zijn overgebracht naar de Microsoft Graph API.

Als laatste hebben we Microsoft Graph. Graph geeft de gebruiker gemakkelijk toegang tot alle data in Azure, SharePoint, Outlook, gebruikers en meer. De API heeft ook een explorer die het makkelijker maakt om requests op te bouwen. Het nadeel van Graph is dat de documentatie af en toe nogal onduidelijk is of niet bestaat. Ook zijn er enkele functionaliteiten van oudere API's die nog niet zijn overgebracht naar Graph.

Het advies is om gebruik te maken van Graph. Dit gebeurt omdat het veiliger en gemakkelijker is om slechts één API te gebruiken. Bovendien is het één API waar we tegen praten in plaats van meerdere. Het mist misschien een paar van de paden die aanwezig zijn in de SharePoint API, maar daar zouden we toch geen gebruik van maken.

5.6. Welke .net libraries zijn er voor Microsoft SharePoint/office?

Voror veiligheid en zekerheid is er voor gekozen om libraries te gruiken die worden geleverd door Microsoft. Van de libraries die Microsft levert zijn er twee die gebruikt kunnen worden voor het integreren van Office 365 in MKGV. De eerste is de Microsoft Identity Platform Library voor het inloggen bij Microsoft. Deze library maakt het gemakkelijker om in te loggen bij Microsoft door een link te leggen tussen de library en de identity servers van Microsoft. Deze library maakt veel gebruik van dependency injection en async methodes, functionaliteiten die niet worden ondersteund in Progress 11.

De tweede library is de Microsoft Graph SDK. Deze library helpt bij het opbouwen van Graph requests door het als methodes in code op te geven. Dit zorgt ervoor dat direct duidelijk is wat voor request er wordt uitgevoerd. Ook deze library maakt veel gebruik van dependency injection en async requests die niet worden ondersteund door Progress 11.

Beide libraries kunnen niet worden gebruikt, omdat er een conflict is met Progress 11 zoals hierboven vermeld. Het is beter om gebruik te maken van de Microsoft Graph en op een later moment terug te komen en te kijken of het niet beter is om een library te gebruiken.

5.7. Conclusie

Voor het uitvoeren van het project ga ik de volgende technieken gebruiken:

- Een Azure geregistreerde app
- Een normale accestoken dat 1 uur geldig is
- Microsoft Graph

De gemaakte technologische keuzes zijn gebaseerd op verschillende overwegingen. De Azure-geregistreerde app is als essentieel beschouwd, omdat deze altijd vereist is wanneer er gegevensuitwisseling met Azure plaatsvindt. Hoewel er alternatieve methoden zijn, raadt Microsoft deze specifieke benadering aan. De keuze voor het normale toegangstoken is ingegeven door het feit dat het, tussen de beschikbare opties om toegang te krijgen tot Azure, als de minst risicovolle wordt beschouwd. Daarnaast is Microsoft Graph API geselecteerd, omdat er geen codebibliotheken beschikbaar waren die in Progress 11 konden worden gebruikt. Hoewel er andere Microsoft API's beschikbaar zijn, was het doel om niet meerdere API's te implementeren voor deze functionaliteit, mede omdat Microsoft het gebruik van andere API's afraadt.

Het onderzoek heeft aangetoond dat de mogelijkheid bestaat, maar heeft ook tal van andere vragen opgeworpen, zoals: wat te doen als een bestand al bestaat, waar de upload naar SharePoint plaatsvindt - aan de front-end of back-end, waar de bestanden worden opgeslagen - in een nieuwe drive of een nieuwe site, hoe de toegangstoken veilig op te slaan zodat deze voor de hele tenant kan

worden gebruikt, en nog veel meer. Deze vraagstukken zullen beantwoord worden tijdens het ontwikkelproces en in overleg met collega's.

6. Technisch ontwerp

6.1. Frameworks

Het framework waar MKGV in is gebouwd, is Progress 11. Progress is een alles-in-één ontwikkelplatform waarin je alles kunt bouwen, van de back-end tot de front-end. De back-end werkt met zijn eigen 4GL-taal genaamd ABL (Advanced Business Language). Het 4GL-gedeelte is vooral gericht op hoe je data in de database kunt bekijken, aanmaken, aanpassen of verwijderen. De rest van de tijd werkt het als een normale programmeertaal. De front-end kan op verschillende manieren worden gebouwd; bij MKG maken ze gebruik van een UWP (Universal Windows Platform) app die ook is gebouwd in ABL. Voor de front-end maken ze ook gebruik van de DevExpress-componentbibliotheek om het gemakkelijker op te bouwen.

6.2. Database

Als database wordt er gebruik gemaakt van de Progress database. Deze database wordt in een multi-tenant architectuur gedraaid voor de cloudversie. De database kan ook lokaal bij klanten opgezet worden; in dat geval wordt de database in de single-tenant modus gezet. De tabelnamen worden regelmatig gebruikt in de code en om onnodig typewerk te beperken heeft MKG ervoor gekozen om de namen van tabellen af te korten naar 4 karakters, zoals "gebruikers" wordt afgekort tot "gebr" en "groepen" tot "grpn".

6.3. Database model

Hieronder staat een klein gedeelte van het databasemodel. De databasestructuur is heel groot en complex opgezet. Daarom staat hier alleen wat belangrijk is voor het uitvoeren van het uploaden van bestanden naar SharePoint. Omdat de structuur al zo complex is, is er een voorkeur om geen nieuwe velden aan te maken als het niet nodig is. Zie figuur 2 voor het databse model

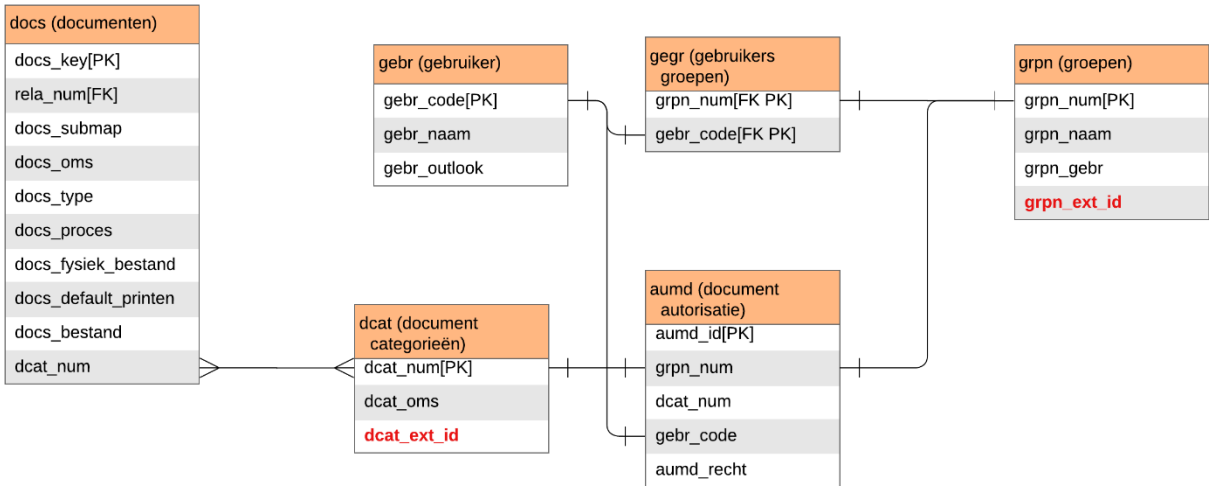
Voor het uitvoeren zijn 2 nieuwe velden nodig, deze velden zijn de "dcat_ext_id" en "grpn_ext_id" in de tabellen "dcat" en "grpn". Deze velden slaan een externe id op naar groepen in bijvoorbeeld Azure, Google of Amazon.

Voor bestanden is er gekozen om gebruik te maken van de "docs_submap". Dit is zo gedaan omdat dat veld ook wordt gebruikt als er alleen een link naar een bestand moet worden opgeslagen. De data die in de "docs_submap" wordt geplaatst ziet er zo uit: "?ExtDoc SharePointID". De "?extDoc" geeft aan dat het een bestand is dat op een externe opslag is opgeslagen; als het "?Local" is, dan is het een link naar een lokaal bestand.

Bij gebruikers is er ook niets toegevoegd en wordt er alleen gebruik gemaakt van de "gebr_outlook". Dit veld bevat een e-mailadres dat gebruikt kan worden om gebruikers te zoeken op de online omgevingen om ze de juiste rechten te geven.

De "gegr" tabel is een koppeltabel die gebruikers koppelt aan een groep. Hier is niets aangepast, maar het is wel belangrijk om gebruikers aan de juiste groepen te koppelen.

De "aumd" tabel is er om de autorisatie tussen gebruikersgroepen, documentcategorieën en gebruikers te regelen.

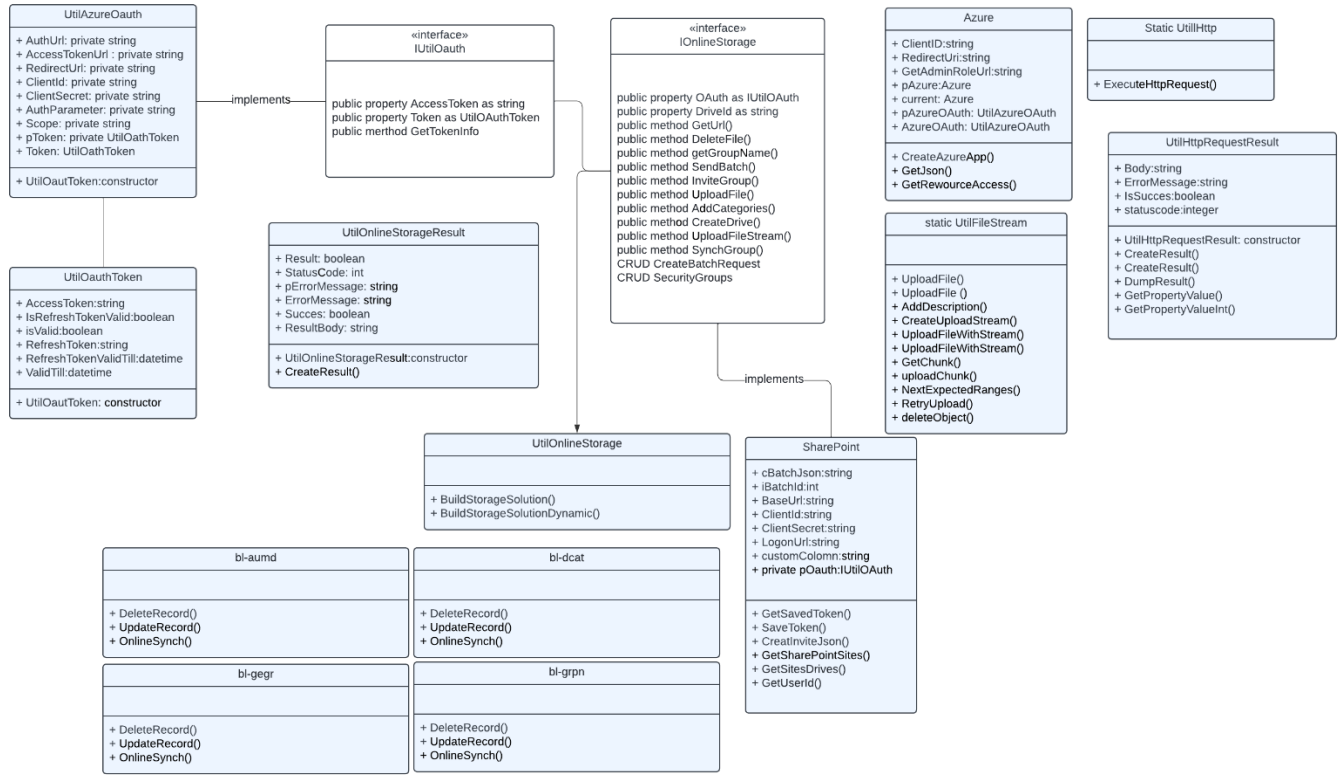


Rode tekst zijn toegevoegde velden aan de database

Figuur 2 Database model

6.4. Benodigdheden implementatie in de code

In het klassendiagram hieronder zie je de klassen, interfaces en methodes die nodig zullen zijn voor het implementeren van SharePoint in MKGV. De verschillende kleuren in het klassendiagram hebben geen specifieke betekenis, deze zijn gegeven door het programma waarin het gemaakt is.



Figuur 3 klassen diagram

Het begint met de interface `IUtilOauth`, die standaard methodes vastlegt voor het gebruiken van oAuth-tokens. De interface wordt gebruikt in `UtilAzureOauth`, die de implementatie voor Azure bevat. De token die we opvragen in `UtilAzureOauth` wordt opgeslagen voor gebruik in de `UtilOauthToken`-klasse. Hier staan een paar methodes om de data op te halen of op te slaan als applicatieproperty, zodat we niet elke keer weer een nieuwe access token hoeven op te vragen.

Als volgende hebben we de interface `IOnlineStorage`. Deze interface geeft een set standaardmethodes die geïmplementeerd moeten worden zodat het online opslaan van bestanden werkt. Deze interface wordt teruggegeven door de `UtilOnlineStorage`, die als factory class fungeert en dynamisch de juiste instantie van de IOnlinestorage interface teruggeeft. De factory class kan ook, afhankelijk van wat je vraagt, een andere class teruggeven. Op dit moment implementeert alleen de `SharePoint`-klasse de interface `IOnlineStorage`. Deze class heeft ook een paar methodes die specifiek zijn voor SharePoint, zoals het ophalen van SharePoint Sites of het ophalen van drives die zijn gelinkt aan een SharePoint Site. Ook is er een `Azure`-class die alleen een nieuwe applicatie kan registreren bij Azure.

De `UtilHttp`-klasse is een class die al aanwezig was; deze class zorgt ervoor dat je makkelijk HTTP-requests kunt uitvoeren in MKGV. In deze class moet ik een methode toevoegen die een `HttpWebResponse` teruggeeft in plaats van de custom class `UtilHttpRequestResult`. Deze methode is er zodat we bestanden kunnen downloaden van SharePoint als een ByteStream. De

`UtilHttpRequestResult` probeert de response altijd als een string op te slaan, en dat werkt niet voor bestanden die je downloadt.

De `UtilOnlineStorageResult` is een helper class die een statuscode en respons opslaat; deze class wordt gebruikt om data tussen methodes uit te wisselen als return type.

De `UtilFileStream`-klasse bevat alle code die nodig is om een bestand als filestream te uploaden naar SharePoint. Het uploaden van een file begint bij de `UploadFile`-methode. Er zijn 2 versies van deze methode; dit komt omdat eerst het idee was om een bestand te uploaden naar de back-end en daarna te uploaden naar SharePoint. Uiteindelijk is er toch maar besloten om de upload vanaf de front-end te doen. Toch moest het mogelijk blijven om een bestand vanaf de back-end te kunnen uploaden, dus is die versie van de methode gebleven. Deze class bevat alle methodes die nodig zijn voor het uploaden, zoals het aanmaken van een uploadstream, verkrijgen van een chunk van de file, uploaden van een chunk en het opnieuw proberen van de upload.

Ook zijn er classes die beginnen met `bl-`. Deze classes zijn standaard in het MKGV-framework. Ze worden gebruikt om data naar de database te checken, updaten, opvragen of aanpassen. In de `bl-dcat`, `bl-grpn`, `bl-gegr` en `bl-aumd`-classes heb ik de `UpdateRecord` en `DeleteRecord` aangepast om data in het opgegeven online documentbeheersysteem aan te maken, verwijderen of aanpassen gebaseerd op een update (als een nieuwe record wordt aangemaakt komt hij hier ook langs) of delete. In deze classes heb ik ook een `sync`-methode aangemaakt die over de hele tabel loopt en de data erin synchroniseert met het opgegeven online documentbeheersysteem.

7. Ontwerpkeuzes

In dit hoofdstuk lees je meer over de verschillende ontwerpkeuzes die zijn gemaakt tijdens de implementatie.

Token op server opslaan of client?

Om connectie te maken met Azure hebben we een toegangstoken nodig, deze token moet ergens opgeslagen worden, de vraag is of we die aan de client of server kant gaan opslaan.

Dit was een lastig dilemma omdat je een toegangstoken veilig wilt opslaan. We zouden hem aan de serverkant kunnen opslaan en gebruiken voor een hele tenant, aan de clientkant en per gebruiker. Aan de serverkant heb je het risico dat iemand hem daar weg kan halen en dan misbruikt, aan de clientkant heb je het probleem dat je voor elke gebruiker een nieuwe toegangstoken moet aanvragen en er dan een heleboel geldige tokens zijn die gebruikt kunnen worden. Uiteindelijk hebben we besloten om de toegangstoken toch maar aan de serverkant op te slaan zodat iedereen binnen een tenant dezelfde token gebruikt. Op deze manier zijn er minder tokens die gestolen kunnen worden.

Refresh token of normale?

Dit was een keuze die makkelijker te maken was omdat een vernieuwingstoken (refresh token) 24 uur geldig is voor single page apps en 90 dagen voor alle andere scenario's. Omdat we geen single page app maken, zou dat betekenen dat een vernieuwingstoken 90 dagen geldig is, iets dat mijn begeleider en de techniek afdeling niet zo veilig vinden. Een normale toegangstoken is geldig voor 1 uur en wordt dynamisch verlengd met 1 uur elke keer dat hij wordt gebruikt. Daarom hebben we ervoor gekozen om gebruik te maken van een normale toegangstoken en niet van de vernieuwingstoken.

Uploaden file via server of client?

Voor het uploaden van een bestand naar SharePoint vanuit MKGV was er de keuze om het aan de server- of clientkant te doen. Onafhankelijk van de upload locatie wordt het bestand in SharePoint opgeslagen en bereikbaar is voor iedereen met toegang. Voor het uploaden was er initieel gekozen voor de serverkant omdat er een agent draait op de server die ook bestanden kan aanmaken en moet kunnen opslaan op SharePoint. Een implementatie hiervoor was al gemaakt. Toen ik met de afdeling techniek ben gaan praten, waren ze niet enthousiast over het idee dat bestanden eerst naar de server moesten worden geüpload. Ze voorzagen problemen met te veel data die naar de server werd gestuurd en de server die vastloopt op de vele verzoeken. Dus uiteindelijk hebben we besloten een hybride oplossing te gebruiken: alle bestanden die door gebruikers moeten worden geüpload, worden aan de clientkant gedaan, en de agent heeft ook nog steeds de mogelijkheid om bestanden via de server te uploaden naar SharePoint.

Batch requests in plaats van single requests?

Dit is een ontwerpkeuze die meer uit noodzaak is gemaakt. De eerste implementatie van het systeem deed allemaal individuele verzoeken naar Microsoft Graph. Hier was een klein probleem mee: elke Put, Patch, Delete of Post naar Graph voor Azure groepen na de derde liet MKG vastlopen. Ik heb er een week aan besteed om het probleem te vinden en op te lossen; zelfs mijn begeleider begreep het niet. Uiteindelijk moest ik een andere manier van verzoeken uitvoeren zoeken, zoals meerdere gebruikers tegelijk aan een groep toevoegen. Na wat rondkijken op de Microsoft Graph-site zag ik dat ze ook batchverzoeken ondersteunden. Dit heb ik getest in MKGV en daar waren geen problemen mee, dus uit noodzaak heb ik de batchverzoeken geïmplementeerd voor het uitvoeren van bepaalde verzoeken naar Microsoft Graph.

Een batchverzoek is gelimiteerd tot 20 requests, wanneer hij dat aantal requests heeft bereikt wordt hij automatisch verstuurd. Bij een aantal lager dan 20 kunnen de verzamelde requests handmatig verstuurd worden in een batch. Er word geprobeerd zoveel mogelijk requests samen te voegen in een batch, dit is niet altijd mogelijk in MKGV in dit geval zal er 1 á 2 requests in een batch zitten.

Kiezen SharePoint Drive.

Voor de keuze van de SharePoint-drive waar alle bestanden worden opgeslagen, heb ik uitgebreide discussies gehad met de afdeling techniek. Verschillende mogelijkheden zijn besproken, zoals het opslaan van bestanden in de hoofd-drive van de standaard SharePoint-drive. Volgens de techniek was dit geen goede optie omdat klanten mogelijk de bestanden afgezonderd willen hebben binnen SharePoint. Na wat extra onderzoek kwam ik terug met twee nieuwe opties: het aanmaken van een nieuwe drive binnen de hoofd-SharePoint-site genaamd MKG, waar alle bestanden worden geüpload, of een map plaatsen binnen de hoofd-drive van de hoofd-SharePoint-site. Deze opties leken mij het beste omdat we geen nieuwe SharePoint-site konden aanmaken en er wat problemen waren met het ophalen van alle SharePoint-sites. Ook deze oplossingen werden niet als ideaal beschouwd, dus ging ik verder op zoek naar alternatieven. Na wat onderzoek heb ik het probleem met SharePoint-sites opgelost. Daarom kwam ik met de oplossing om een dropdown-menu in MKGV te plaatsen waar alle SharePoint-sites staan die niet van een persoon zijn en waar klanten uit kunnen kiezen. Ze kunnen ook een drive kiezen die verbonden is aan de site uit een andere dropdown of een nieuwe drive laten aanmaken die ze kunnen gebruiken. Dit werd beschouwd als een mooie oplossing en is ook geïmplementeerd.

Azure geregistreerde app via script of Graph Request?

Voor het aanmaken van een Azure-geregistreerde app zijn er drie manieren om dit te doen. De eerste is een handleiding maken die uitlegt hoe je dit kunt doen via Azure en welke rechten je aan de app moet toekennen. De tweede optie is het schrijven van een script dat gebruikmaakt van de Azure CLI om een nieuwe app aan te maken met alle benodigde gegevens. De derde optie is om via Microsoft Graph een nieuwe app te laten aanmaken. Hoewel de eerste optie niet de beste is, dient deze als een goede back-up voor het geval de gekozen optie niet werkt. De tweede optie kan functioneren, maar vereist dat we een script aan klanten leveren en extra benodigdheden voor hun installeren, zodat het script correct werkt. De derde optie wordt als de beste beschouwd omdat het eenvoudig vanuit MKGV kan worden uitgevoerd. Als er aanpassingen nodig zijn in de app, kan dit eenvoudig worden gedaan door de JSON die wordt meegegeven aan te passen. Ik heb de laatste twee opties voorgelegd aan alle ontwikkelaars tijdens een demo die ik heb gegeven. Iedereen was het erover eens dat het makkelijker en handiger was om te kiezen voor optie 3.

Upload stream conflicBehaviour rename, replace, fail?

Deze kwestie heeft betrekking op wat Graph moet doen als er al een bestand in de opgegeven folder bestaat met dezelfde naam. Microsoft biedt je de volgende 3 opties: hernoemen, vervangen en niet uitvoeren. Het niet uitvoeren van het uploaden is geen optie, omdat we niet kunnen aangeven dat de upload niet is gelukt vanwege het bestaande bestand. Vervangen van het bestand is ook geen eenvoudige oplossing, omdat het mogelijk twee verschillende bestanden zijn met dezelfde naam. In dat geval kunnen we niet simpelweg het oude bestand vervangen door een nieuwe. De enige overgebleven keuze is het hernoemen van het bestand. Als een bestand al bestaat, wordt het nieuwe bestand hernoemd door er een getal aan toe te voegen. Deze nieuwe naam wordt door MKGV ontvangen als respons en doorgegeven aan ons documentbeheersysteem. Dit zou eigenlijk geen probleem moeten zijn vanwege de gebruikte folderstructuur.

Opzet security groepen?

Voor de beveiligingsgroepen is gekozen om dezelfde structuur aan te houden als in MKGV. Dat betekent dat gebruikers kunnen worden toegevoegd aan groepen en groepen aan documentcategorieën. Ook kunnen gebruikers individueel worden toegevoegd aan documentcategorieën. Dit wordt gedaan om consistentie te behouden met MKGV.

Folder structuur op SharePoint?

Voor de folderstructuur heb ik meerdere gesprekken gehouden met de afdeling techniek om de opties te bespreken. Deze opties waren een structuur gebaseerd op documentcategorieën, gebruikers zelf een structuur laten opzetten, gebaseerd op groepen, of wat MKGV nu gebruikt: Jaar/Maand/Dag/Seconden. Hier hebben we ervoor gekozen om te gebruiken wat MKGV nu al gebruikt, omdat dat het makkelijker maakt voor bestaande gebruikers om te gebruiken. Als het onoverzichtelijk wordt, kan er gebruik worden gemaakt van de zoekfuncties die online Documentbeheersystemen hebben om snel en gemakkelijk een bestand terug te vinden.

Lege folders laten staan?

Hierbij was de vraag of, als we een bestand verwijderen en er zou een lege folder achterblijven, we die ook verwijderen of laten staan. Na het te hebben overlegd met collega's tijdens een demo die ik heb gegeven, hebben we besloten om lege folders ook te verwijderen. Op die manier blijft het online documentbeheersysteem opgeruimd.

Zelf opzetten extra SharePoint kolom.

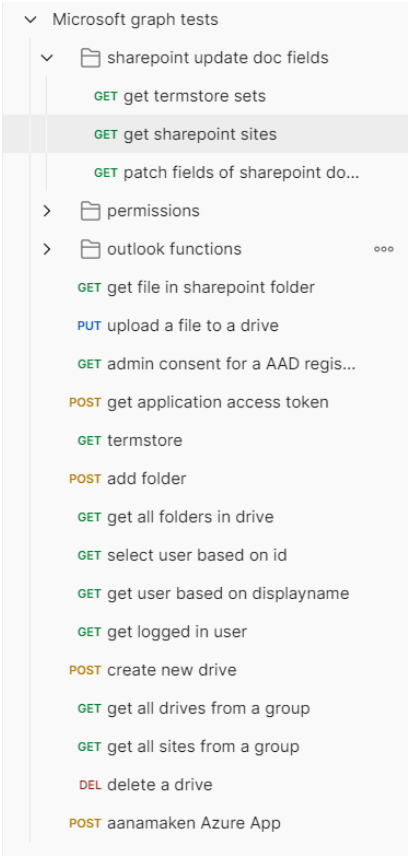
Dit is een designkeuze die niet anders kan; wij kunnen niet vanuit MKGV een nieuwe kolom aanmaken in SharePoint. Daarom is ervoor gekozen om een uitleg te schrijven die uitlegt hoe klanten een nieuwe kolom kunnen aanmaken in SharePoint en die kunnen linken aan MKGV.

Tekst veld in plaats van managed metadata.

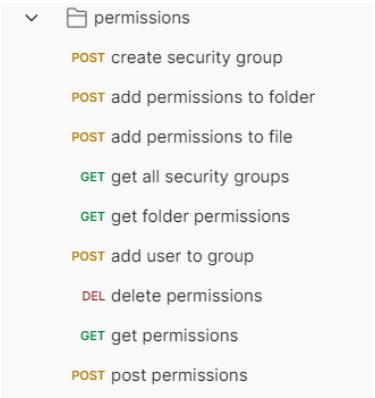
Dit is een designkeuze die is beïnvloed door de mogelijkheden van de API waarmee we werken. Voor een extra kolom in SharePoint kunnen we kiezen uit meerdere vormen van data die erin worden opgeslagen. Twee daarvan zijn Managed Metadata en tekst. Managed Metadata zou de betere optie hiervoor zijn omdat van tevoren kan worden vastgezet welke data erin kan worden gezet. Alleen kan Managed Metadata niet worden aangepast of ingevuld via Microsoft Graph of de SharePoint API. Dit betekent dat de enige goede optie een simpel tekstveld is waarin de extra data wordt geplaatst. Het tekstveld heeft wel het nadeel dat iedereen de informatie erin kan aanpassen als ze willen. Om dit te doen, moeten ze wel eerst naar het detailscherm van een bestand gaan.

8. Implementatie

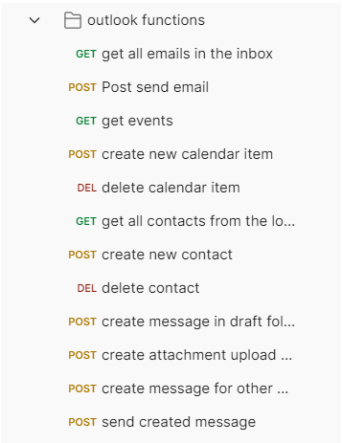
Voor de implementatie heeft er een vooronderzoek plaatsgevonden, waarbij al een overzicht is gemaakt van elke HTTP-request die nodig is voor het uitvoeren van de opdracht. Hieronder zie je de lijst in Postman. Alle requests die worden uitgevoerd, zijn gebaseerd op de requests in de Postman-collectie. Deze selectie van apirequests is gemaakt op basis van Microsoft Graph Explorer (https://developer.microsoft.com/en-us/graph/graph-explorer).



Figuur 4 SharePoint requests in postman



Figuur 5 SharePoint requests in postman

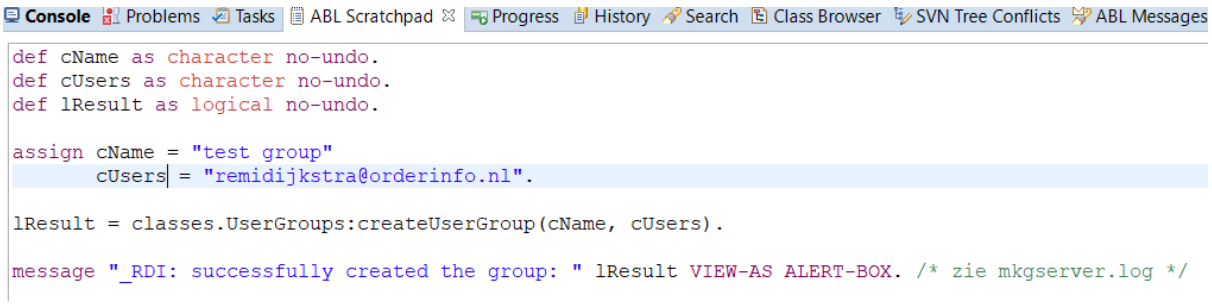


Figuur6 SharePoint requests in postman

8.1. POC SharePoint

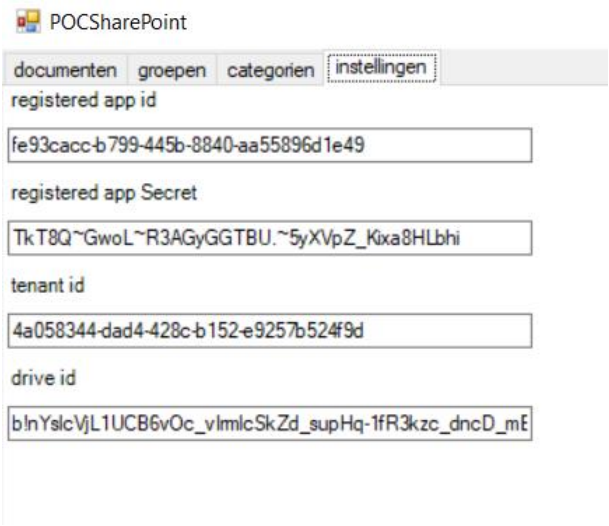
Het maken van de POC voor SharePoint gebeurde aan de client kant en verliep voorspoedig; er kwamen geen problemen naar voren. Een deel van de implementatie was al tijdens de vooronderzoeks- en onderzoeksfase gemaakt, namelijk het uploaden van bestanden naar SharePoint. Ik had toen al een script geschreven dat bestanden in chunks opdeelde en uploadde, het chunken is toegevoegd omdat er een limiet is aan de uploadgrootte. Zo kunnen grote bestanden automatisch in delen worden geupload. Dit script heb ik hergebruikt voor deze POC en de implementatie zelf.

Om de POC te maken, ben ik begonnen met het opzetten van de klassen zoals omschreven in het klassendiagram (zie figuur2). Nadat de structuur was opgezet, ben ik begonnen met het implementeren van de basisrequests naar Microsoft Graph. Deze basis heb ik getest door gebruik te maken van ABL scratchpad. Scratchpad stelt je in staat om je code als script uit te voeren en is een goede hulp als je snel dingen wilt prototypen of testen. Een voorbeeld kun je zien in figuur 6.



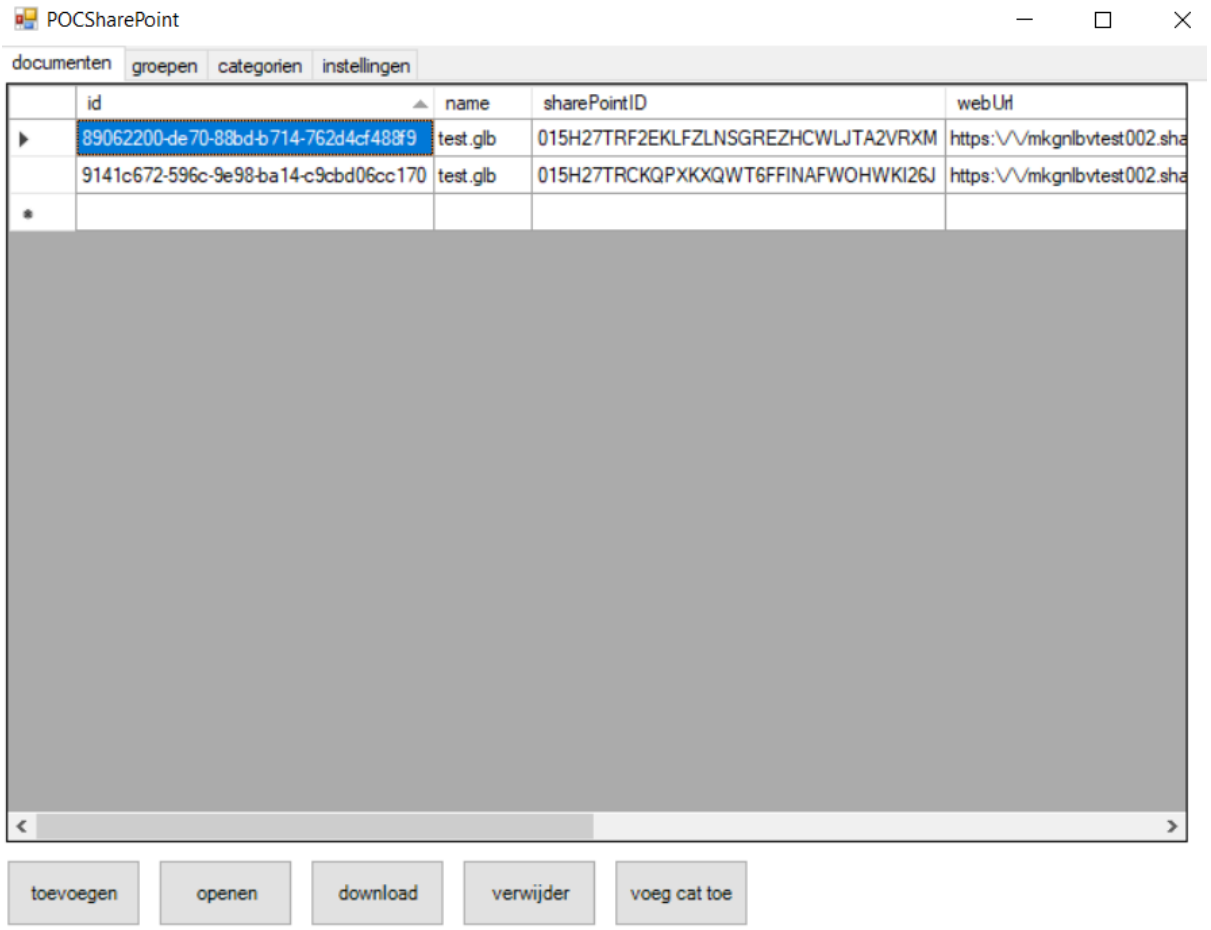
Figuur 7 code test in scratchpad

Met behulp van Scratchpad kon ik alle benodigde requests testen voordat ik een interface had gebouwd. Er is voor gekozen om deze tests niet te automatiseren, omdat het slechts om een POC gaat die maar één keer gebruikt gaat worden. Bovendien waren de testen bedoeld om te zien of de requests goed uitgevoerd worden en niet om de constantheid van de functionaliteit te testen. Nadat alle requests waren gebouwd en getest, ben ik begonnen met het ontwikkelen van een interface. Deze interface is opgezet in UWP (Universal Windows Platform) met behulp van de ingebouwde Windows Forms. Ik ben gestart met een instellingen-tab, zodat de gegevens voor Microsoft Graph niet langer hardcoded in de code hoefden te staan. Zie figuur 7.



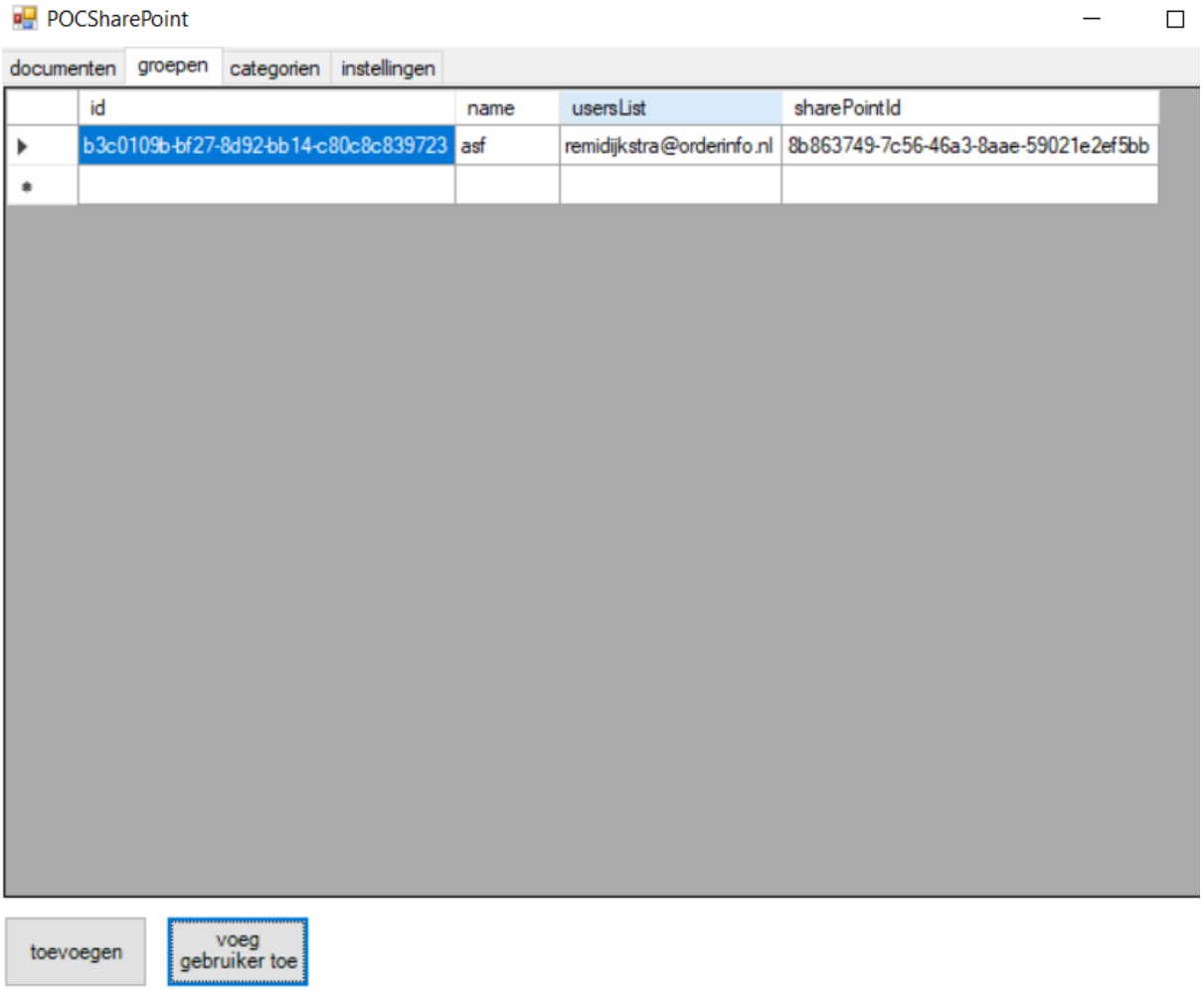
Figuur 8 POC instellingen scherm

Na de instellingen-tab heb ik een tab gemaakt waar informatie over bestanden kan worden weergegeven. Dit is bereikt door gebruik te maken van een DataGridView die de gegevens uit een tijdelijke tabel toont. Een tijdelijke tabel is een in-memory database tabel die je zelf kunt definiëren of waarvan je de definitie van een reeds bestaande tabel kunt gebruiken. Om de gegevens van de tijdelijke tabel in de DataGridView te krijgen, moet je een BindingSource gebruiken die een query van de tijdelijke tabel naar een datasource kan casten. Zo kan de data die laat zien waar het bestand is opgeslagen worden getoond. Hieronder zie je het resultaat van de BindingSource en tijdelijke tabel in de DataGridView. Zie figuur 8



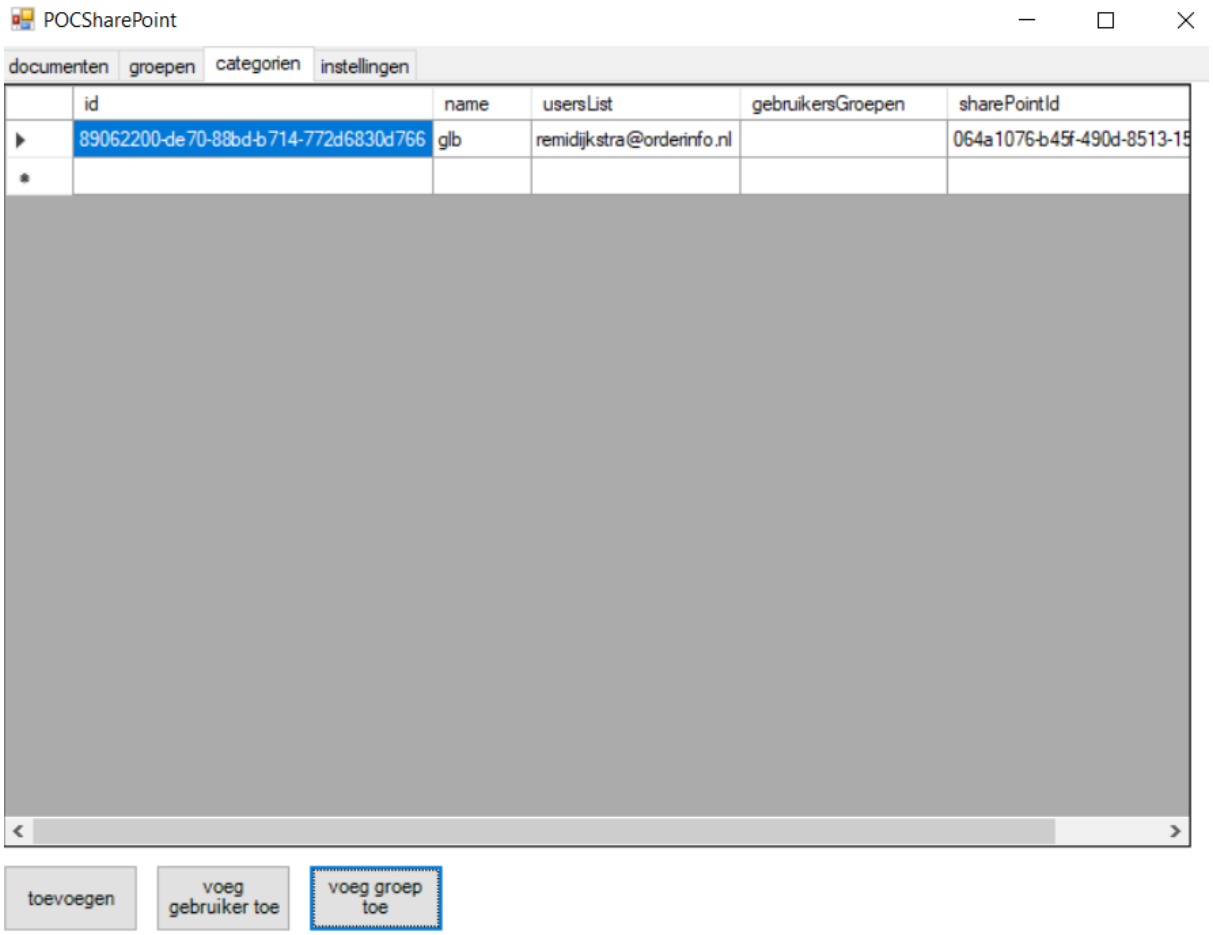
Figuur 9 POC documenten scherm

Onder de DataGridView zijn knoppen toegevoegd om acties uit te voeren, zoals het openen, verwijderen of downloaden van een geselecteerd bestand. Ook is er de optie om een nieuw bestand te uploaden. Als je op deze knop drukt, opent het een file explorer-scherm waar je een bestand kunt selecteren dat vervolgens naar SharePoint wordt geüpload. Bij deze upload hoort ook het delen van het bestand met een al bestaande documentcategorie en het toevoegen van tags aan de aangepaste kolom die ook vaststaat. Daarna ben ik verdergegaan met het maken van een scherm waar je groepen kunt aanmaken en bekijken. Ook hier wordt gebruik gemaakt van een DataGridView en tijdelijke tabel om gegevens op te slaan en weer te geven. Bij groepen kun je ze alleen maar aanmaken en gebruikers toevoegen, zie hieronder. Als je een gebruiker wilt toevoegen, moet je een e-mailadres opgeven dat bestaat in de Azure-omgeving waar de groep is aangemaakt. Als je dat niet doet, krijg je een foutmelding terug. Zie figuur 9.



Figuur 10 POC groepen scherm

Daarna heb ik een tab toegevoegd voor documentcategorieën die ook gegevens van een tijdelijke tabel via een DataGridView laat zien, zie hieronder. Ook hier zijn knoppen onder de DataGridView geplaatst. Via deze knoppen kun je nieuwe categorieën maken, gebruikers toevoegen of groepen toevoegen. Het toevoegen van gebruikers werkt hetzelfde als bij groepen. Voor het toevoegen van groepen is er een dropdownmenu dat alle mogelijke groepen laat zien. Zie figuur 10.



Figuur 11 POC categorieën scherm

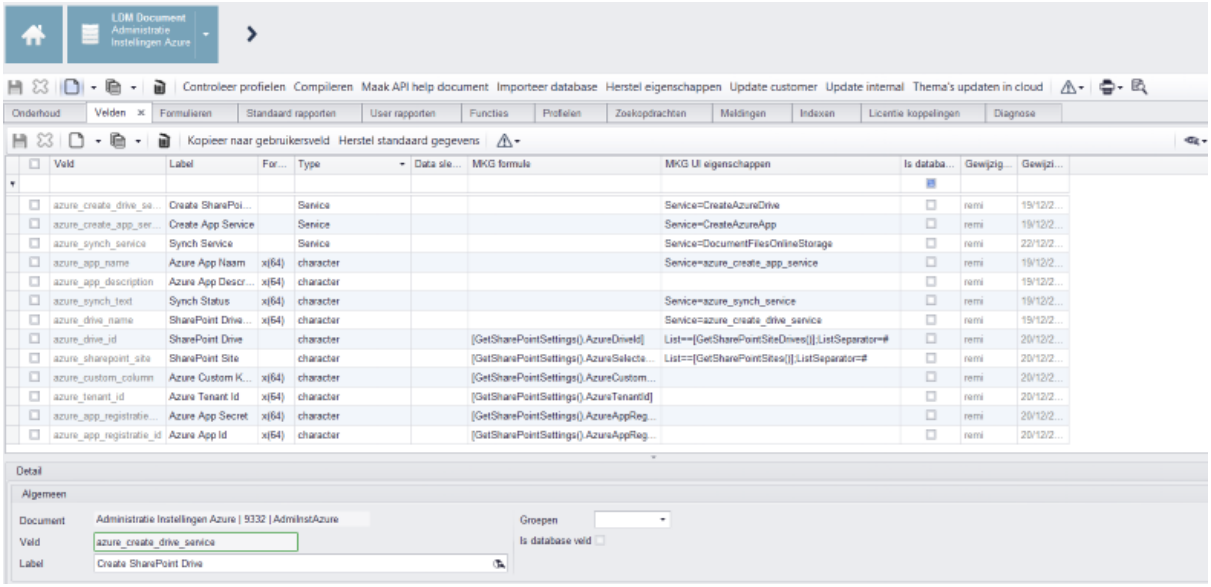
Als laatste heb ik een vorm van gegevenspersistentie toegevoegd. Temp tables hebben twee handige functies die hierbij kunnen helpen. Eén daarvan schrijft alle gegevens in de temp table weg naar een XML-bestand, de andere kan een XML-bestand uitlezen naar een temp table. Voordat de POC wordt afgesloten, schrijf ik alle gegevens weg naar drie XML-bestanden die worden opgeslagen in de tijdelijke map. Als de POC wordt opgestart, controleer ik of de bestanden bestaan. Als ze bestaan, lees ik de gegevens uit en zet ik ze in de tijdelijke tabel. En dat is alles wat ik heb gedaan voor de POC om aan te tonen dat het uploaden, downloaden, maken van groepen en toevoegen van gebruikers mogelijk waren.

8.2. Implementatie SharePoint

8.2.1. Instellingen pagina

Ook bij de implementatie van de nieuwe functionaliteit ben ik begonnen met het toevoegen van instellingen die door de gebruiker kunnen worden ingevuld. In MKGV wordt dit gedaan door een LDM (logisch datamodel) document aan te passen. LDM documenten worden in dit geval gebruikt om data en functies te definiëren om een interface mee te bouwen.

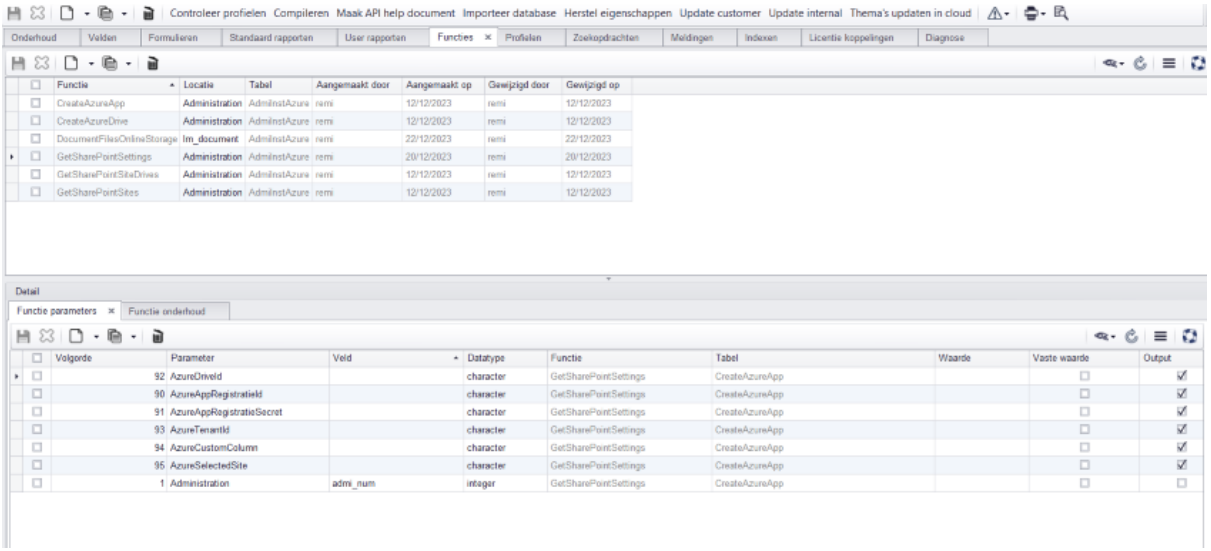
Document 9200 is een van deze LDMs. In het begin voegde ik alle velden toe die ik nodig had aan het document "9200 administratie instellingen". Dit werkte voor een tijdje, totdat ik meer data en methodes nodig had. Toen liep ik tegen het probleem aan dat het document "9200" te groot was en niet meer kon worden gecompileerd. Om dit probleem op te lossen, heeft mijn begeleider een nieuw LDM document voor me gereserveerd met het nummer "9332" (Azure-instellingen), zoals hieronder te zien is. Zie figuur 11.



Figuur 12 LDM document 9332 velden voor de interface

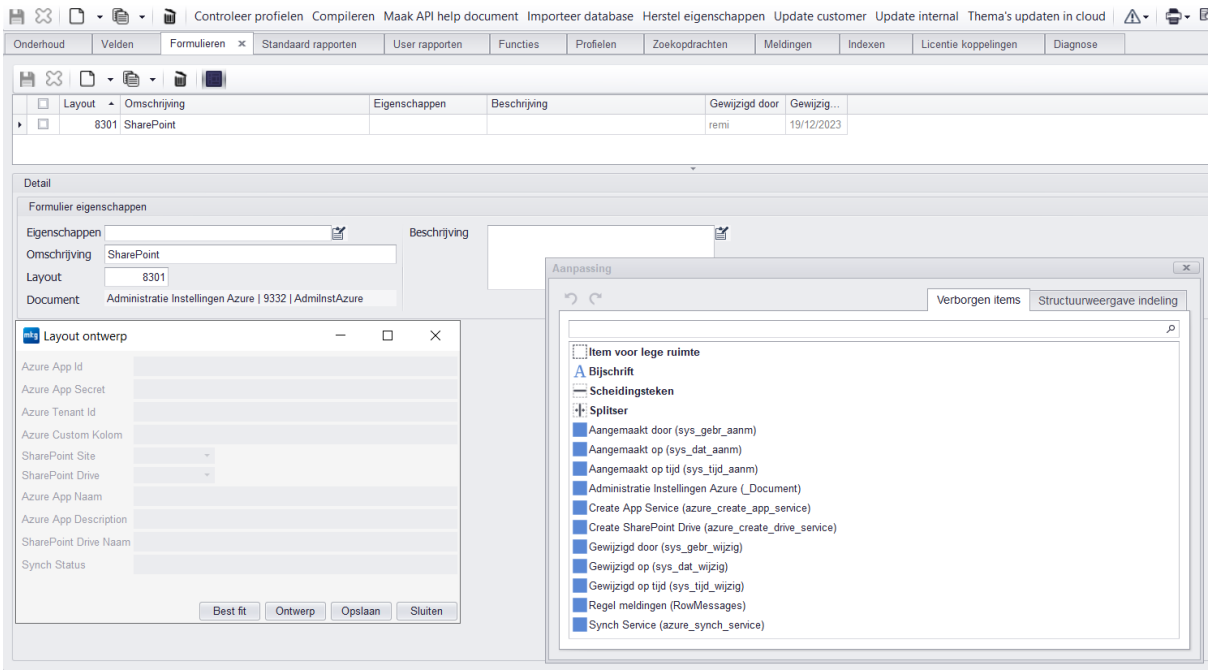
In dit document heb ik character velden opgezet die informatie waar gebruikers data kunnen invullen die nodig is om connectie te maken met de Azure geregistreerde app. Deze velden halen de opgeslagen informatie weer op uit de methode die staat in de MKG Formule veld, dit is een methode die is gedefinieerd in de functies tab. Ook heb ik 2 dropdown menu's gemaakt waar de gebruiker een keuze kan maken onder welke SharePoint site de documenten worden geplaatst, en dan in die SharePoint site in welke drive ze worden geplaatst. de SharePoint Sites waar je uit kan kiezen zijn alleen degene die vastzitten aan de Azure tennant waar bij je bent ingelogd, ook zijn hier de persoonlijke sites uit gefiltered via code omdat dat niet mogelijk is via Microsoft Graph. De drives zijn gebaseerd op de SharePoint site die is geselecteerd, als je de site aanpast update de lijst met drive direct.

Ook heb ik drie services geregistreerd deze services kunnen methodes die in de functie tab staan aanroepen. Deze 3 services zijn er om een nieuwe Drive aan te maken als die nog niet bestaat, een nieuwe Azure app aan te maken en om alle data uit MKGV te synchroniseren naar Azure. Deze services zijn vastgemaakt aan tekst velden omdat er eerst geen manier was services vanaf formulieren te laten uitvoeren, dus omdit mogelijk te maken heeft mijn begeleider deze functionaliteit toegevoegd. Zie figuur 12.



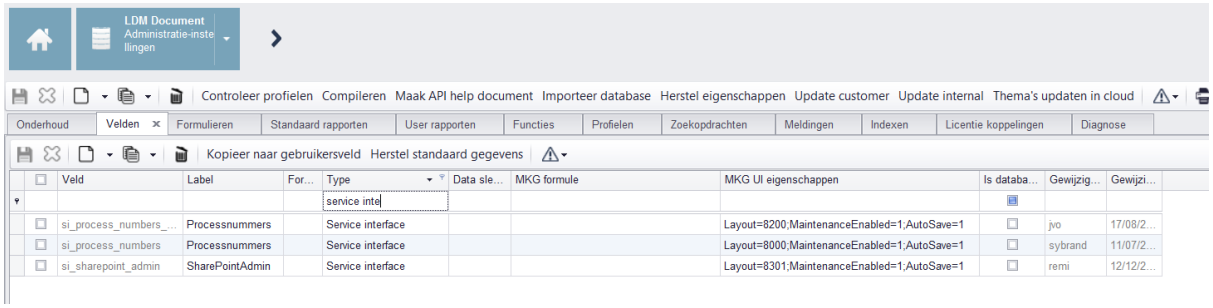
Figuur 13 LDM document 9322 functies die zijn geregistreerd

De functies tab heeft alle methodes die uitgevoerd kunnen worden vanuit het Document. Je registreert een nieuwe functie door de naam van de methode op te geven en het bestands naam waar hij in staat, onderin heb je een detail scherm waar de parameters kan opgeven en door kan geven of die uit velden komen, vaste waarden hebben of een output zijn. Nadat alle is opgezet en is gecompileerd kan ik een formulier opzetten. Dit wordt gedaan in het formulier tab.zie figuur 13



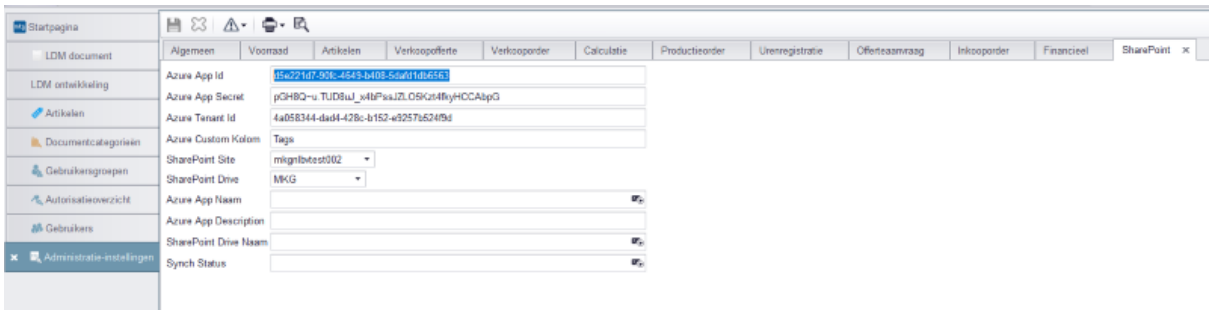
Figuur 14 LDM document 9332 formulier(interface)

In deze tab heb ik een nieuwe formulier aangemaakt voor de instellingen, door het formulier te selecteren en op de blauwe vierkant te drukken naast de prullenbak open je de ontwerper. De ontwerper geeft je alle velden die je hebt aangemaakt in het document als optie om op het formulier te plaatsen, ook zijn er standaard velden die je kan toevoegen. Nadat ik alle nodige velden had toegevoegd heb ik het formulier opgeslagen en het document opnieuw gecompileerd, nu heb ik wel een formulier maar kan ik hem nog steeds niet zien bij de administratie instellingen, dit kan opgelost worden door een service interface toetevoegen aan het document 9200 administratie instellingen zie hieronder. Zie figuur 14.



Figuur 15 LDM document 9200 link naar LDM document 9332

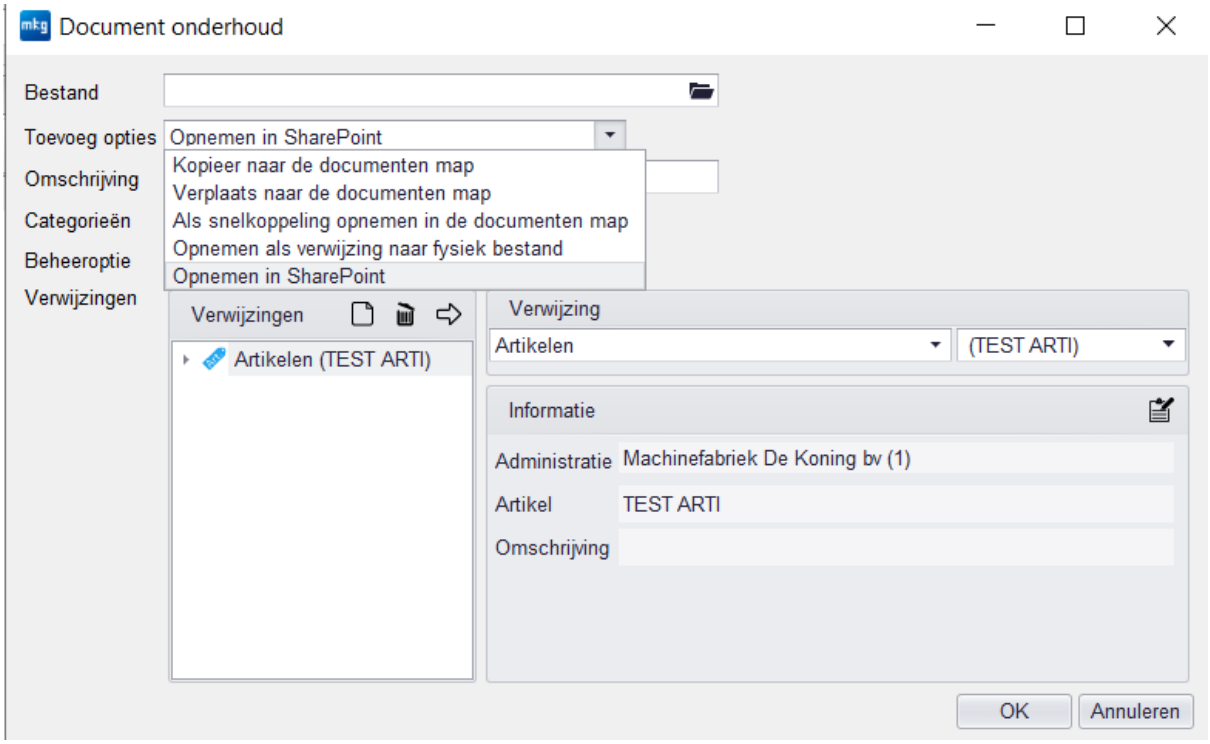
Een service interface legt een link tussen twee documenten, in dit geval heb ik een service interface gebruikt om het formulier die ik heb gemaakt in document 9332 Azure instellingen toe te voegen de administratie instellingen. Alleen de interface toevoegen is niet genoeg, je moet het formulier ook toevoegen aan een andere formulier want anders kan je hem nog steeds niet gebruiken bij de administratie instellingen. Zie hieronder het resultaat. Zie figuur 15.



Figuur 16 instellingen scherm in MKGV

8.2.2. Upload optie toevoegen

Nu heb ik wel een formulier waar ik data in kan zetten maar het wordt niet automatisch opgeslagen, het opslaan van de administratie instellingen gaat via de SaveSettings methode, deze methode is een grootte for loop met een switch case erin die checkt welk veld is aangepast en dan de data op de correcte plaats opslaat. Omdat mijn velden niet in document 9200 zijn gedefinieerd heb ik een nieuwe methode in de Administration.p file moeten maken die wordt aangeroepen door SaveSettings. Deze methode heet SaveSharePointSettings, in deze methode lees ik de velden uit document 9332 uit en loop ik over alle velden om daarna met een switch case de data op de goed plek te zetten. Ook check ik hier of ik genoeg informatie heb om een connectie met Microsoft Graph te maken, Als ik alle data heb die nodig is dan zet ik SharePointActive property op true, dit zorgt ervoor dat je de “upload to SharePoint” optie te zien krijgt bij het toevoegen van een nieuw bestand aan het document beheer systeem en dat je data kan synchen naar Azure. Hieronder zie je de nieuwe optie als je alle nodige informatie hebt ingevuld. Zie figuur 16.



Figuur 17nieuwe Sharepoint optie in document toevoegen scherm

8.2.3. Bestand uploaden

Nadat ik het nieuwe formulier heb toegevoegd en de nodige data had toegevoegd ben ik begonnen met het uploaden van een bestand. Hiervoor heb ik de code die ik heb geschreven voor de POC kunnen hergebruiken, er waren wel een paar verandering nodig in hoe requests werden uitgevoerd. In de POC had ik 5 methodes gemaakt die een ieder een type request uitvoert, in MKGV was er al code in de UtilHttp die requests kan uitvoeren maar dat is 1 methode met 8 overloads. Nadat de code was aangepast ben ik begonnen met het aanpassen van de AddFile methode in de UtilFiles class, in deze class heb ik een nieuwe methode toegevoegd die een bestand upload naar SharePoint, deze methode wordt alleen uitgevoerd als je de “Upload to SharePoint” optie hebt gekozen. De UploadFile methode doet een requests naar de server voor een access token die hij kan gebruiken voor de connectie met Microsoft Graph, in dezelfde request vraagt hij ook de ID van de drive op waar het bestand opgeslagen moet worden.

Nadat hij een response terug heeft gekregen van de server roept hij de UploadFile methode van UtilLayer class aan, de UtilLayer class is een class die kijkt of hij aan de client of de server kant is en dan gebaseerd daarop een andere interface implementatie gebruikt voor het uitvoeren van een methode. De UploadFile methode van de UtilLayer class vraagt dan weer aan de UtilOnlinStorage class welke optie actief is en krijgt dan een implementatie van de IOnlinStorage interface terug. Als er meerdere online opslage systemen actief zijn komt hij met een error terug omdat hij niet weet welke gebruikt moet worden.

Voor de SharePoint implementatie beginnen we met het aanmaken van een locatie waar het bestand wordt opgeslagen gebaseerd op de huidige tijd. Dit wordt dan gebruikt samen met de Drive ID, Access Token en file grootte om een Create Upload Stream request te doen naar MicroSoft Graph. Als de Response 200 is dan krijgen we een link terug die we kunnen gebruiken om de file chunks naar te uploaden. Als we een error terug krijgen dan geven we een error terug die zegt dat er geen connectie gemaakt kan worden met Azure.

De volgende stap is het opdelen van het bestand in chunks en ze uploaden. Het opdelen van het bestand begint met het berekenen hoeveel chunks er zullen zijn dit wordt gedaan door de bestands grootte te delen door de chunksize en omhoog af te ronden. Dit gebruiken we als we chunks te maken om te kijken of we een volledige chunk, een gedeeltelijke chunk of meerdere chunks moeten gaan sturen. Als we een chunk hebben gebruiken we de upload link die we hebben gekregen samen met de access token om hem op te sturen naar SharePoint. Dit doet MKG totdat alle chunks zijn verstuurd, als alle chunks gestuurd zijn geeft Microsoft Graph een 201 terug met de drive item ID.

Tijdens het uploaden houden we bij hoeveel bytes we hebben gestuurd en bij welke chunk we zijn gebleven, als we een chunk hebben overgeslagen dan stopt de loop met uitvoeren en proberen we het uploaden opnieuw vanaf de volgende range aan bytes die Microsoft Graph verwacht. Het opnieuw uploaden van een bestand proberen we 3 keer als het na de derde keer nog niet lukt geven we een error bericht terug.

Als het bestand succesvol is geüpload dan geeft de UploadFile methode de ID van de SharePoint item en het pad waar het is opgeslagen, het ID gebruiken we direct als er een omschrijving is opgegeven. De documentatie van Microsoft geeft wel aan dat je direct een omschrijving kan toevoegen als je een upload stream maakt maar dat werkt niet zoals ze zeggen, daarom voegen we de omschrijving pas toe wanneer het bestand is geüpload. Ook checken we dan of de gebruiker een eigen kolom heeft opgegeven waar de document categorieën aan kunnen worden toegevoegd, als er een kolom is opgegeven gebruiken we het ID weer om de categorieën in SharePoint toe te voegen.

Als het uploaden klaar is gebruiken we de docs_submap veld om de SharePoint ID in op te slaan, dit doen we door eerst “?EXT ” neer te zetten zo weten we direct dat het een externe bestand is. Ook zetten we in de docs_fysiek_bestand het pad waar het bestand is opgeslagen, voorafgaand aan het pad zetten we “SharePoint:” om aan te geven dat het op SharePoint staat.

Hieronder zie je in MKGV een bestand dat is geüpload en hetzelfde bestand op SharePoint staan.

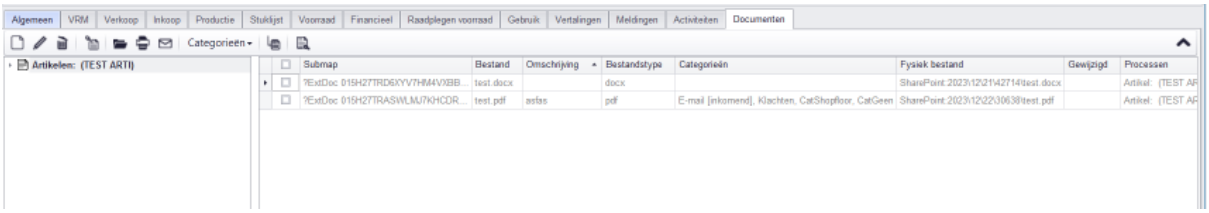


Figure 18 nieuwe document in documentbeheer systeem in MKGV

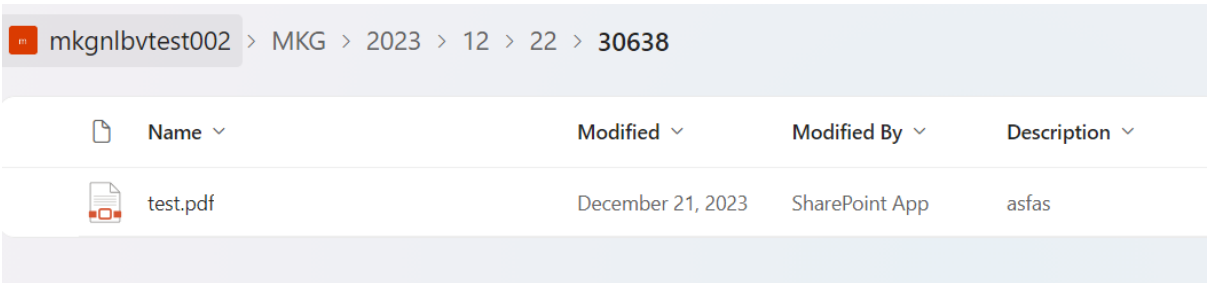


Figure 19 zelfde document in SharePoint

8.2.4. Openen, Printen, verwijderen bestanden

Nadat het uploaden geïmplementeerd is ben ik bezig gegaan met het implementeren van de overige functies die bij bestanden horen zoals het open, afdrukken en verwijderen. Als eerst ben ik begonnen met openen, het openen van bestanden was vrij simpel. Hoe het systeem werkte was dat het volledige pad van het bestand werd uitgevoerd door de OS waar MKGV op draait. Dit zorgt ervoor dat we niet hoeven na te denken over welk programma nodig is voor elk bestand, dit helpt ook met het openen van SharePoint bestanden omdat ik alleen een link hoef te geven aan de open methode. Om de link te verkrijgen heb ik een Get request naar Microsoft Graph uitgevoerd met de ID die bij het bestand hoort.

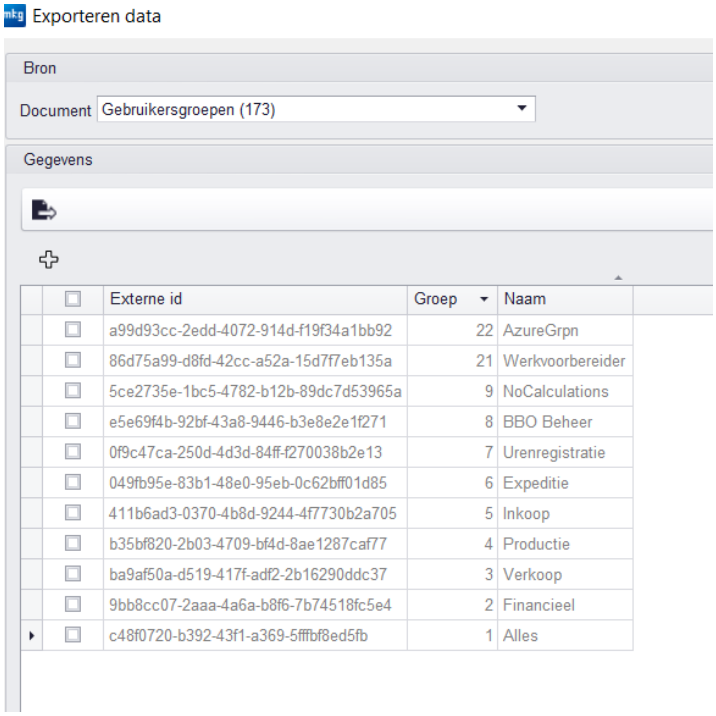
Het printen van een bestand was niet zo makkelijk omdat we een fysiek bestand nodig hebben op het systeem waar de client draait. Gelukkig heeft MKGV al een methode om tijdelijke bestanden te maken, dit heb ik gebruikt samen met een Get Request die een byte array van de SharePoint file terug geeft. Om dit voor elkaar te krijgen heb ik wel een nieuwe methode moeten toevoegen aan de UtilHttp class die een HttpWebResponse terug geeft, uit deze response kan ik een message stream krijgen ik kan gebruiken om de tijdelijke file die is gemaakt te vullen met de data uit SharePoint. Als laatste geef ik het pad van de tijdelijke file mee aan de al bestaande print methode die de rest afhandelt.

Het verwijderen van een SharePoint bestand was eerst simpel, het enige wat ik hoefde te doen was een Delete Request naar Microsoft Graph met de ID van het bestand dat verwijdert moet worden. Het probleem hiervan is dat er dan een heleboel lege folders achterblijven wat niet mooi is, om dit op te lossen had ik een recursive methode gemaakt die de parent van een item ophaalde en checkte of een item kinderen had, als de item geen kinderen had werd hij verwijderd en ging hij naar de parent. Dit zou hij doen totdat hij bij de root komt, als hij de bij de root aankwam stopte hij. Jammer genoeg werkte dit niet na 2 Delete requests te hebben uitgevoerd bleef de HttpWebResponse vastlopen en konden er geen Requests uitgevoerd worden. Ik heb samen met mijn begeleider gekeken en we weten allebei niet waarom hij blijft hangen. Uiteindelijk heb ik een oplossing gevonden, als ik een folder verwijder dan verwijdert hij ook meteen de kinderen. Dus door weer recursief door de boom te lopen kijkt hij of er meer dan 1 kind is, als er meer dan 1 kind is dan verwijderd hij het kind waar hij vandaan komt, als hij 1 kind vindt dan gaat hij een niveau omhoog om daar te kijken. Dit doet hij totdat hij bij de root is aangekomen.

8.2.5. Groepen

Nadat alles rondom bestanden klaar is ben ik begonnen met het maken van groepen voor de gebruikers en document categorieën. Ik ben begonnen met de gebruikers groepen, het eerste wat ik heb gedaan is uitvogelen hoe het aanmaken van groepen gaat. Het aanmaken, verwijderen en aanpassen van groepen gaat allemaal vi de bl-grpn.p bestand, dit bestand bevat alle code die te maken heeft met groepen. Na wat te hebben rondgekeken en gevraagd is mij verteld dat het aanmaken en aanpassen van een groep allemaal loopt via de UpdateRecord methode, het verwijderen loopt via de DeleteRecord methode. dus voor het aanmaken en updaten van een groep heb ik code toegevoegd aan de UpdateRecord methode die checkt of het een nieuwe groep is en dan de methode CreateGroup aanroept in de IOnlineStorage interface, deze methode geeft een ID terug die wordt toegevoegd aan de groep record voordat hij wordt opgeslagen. Als het een update is dan check ik of de nieuwe naam anders is dan de opgegeven naam in Azure, als de namen anders zijn dan update ik hem naar de nieuwe anders doe ik niks. Het verwijderen is simpel ik neem de ID die vast zit aan de groep record en die geef ik mee aan een Delete request naar Microsoft Graph. als er geen ID is dan doet hij niks. Hieronder zie je de een export van groepen in MKGV met een externe ID en dezelfde groepen in Azure.

Hieronder zie je een export van groepen met de externe id's en een groep in Azure.



Externe id	Groep	Naam
a99d93cc-2edd-4072-914d-f19f34a1bb92	22	AzureGrpn
86d75a99-d8fd-42cc-a52a-15d7f7eb135a	21	Werkvoorbereider
5ce2735e-1bc5-4782-b12b-89dc7d53965a	9	NoCalculations
e5e69f4b-92bf-43a8-9446-b3e8e2e1f271	8	BBO Beheer
0f9c47ca-250d-4d3d-84ff-f270038b2e13	7	Urenregistratie
049fb95e-83b1-48e0-95eb-0c62bff01d85	6	Expeditie
411b6ad3-0370-4b8d-9244-4f7730b2a705	5	Inkoop
b35bf820-2b03-4709-bf4d-8ae1287caf77	4	Productie
ba9af50a-d519-417f-adf2-2b16290ddc37	3	Verkoop
9bb8cc07-2aaa-4a6a-b8f6-7b74518fc5e4	2	Financieel
c48f0720-b392-43f1-a369-5fffbf8ed5fb	1	Alles

Figure 20 export van de bestaande groepen in MKGV met een externe id

Figure 21 een bestaande beveiligings groep in Azure die bestaat in MKGV

Nu ik gebruikers groepen heb kan ik gebruikers toevoegen, in MKGV wordt dit gedaan door een drop down waar je meerdere gebruikers kan selecteren die worden toegevoegd. deze informatie wordt opgeslagen in een koppeltabel genaamd gegr, alle code voor die tabel staat in een bestand genaamd bl-gegr. Ook hier heb ik code toegevoegd aan de UpdateRecord en DeleteRecord methodes. bij de update record heb ik toegevoegd dat gebruikers worden toegevoegd aan groepen via de methode AddUserToGroup in de IOnlineStorage interface, voor het toevoegen van een gebruiker aan een group heb je de gebruikers ID nodig, deze weten we niet in MKGV maar we hebben wel een e-mail dus gebruiken we opgegeven e-mail om de gebruikers Azure ID op te halen zodat we hem kunnen toevoegen aan de groep. Als er geen ID is gevonden dan doen we niks.

Hier heb ik een check moeten inbouwen voor het geval een record wordt aangemaakt maar ook een record met dezelfde data wordt verwijderd, in dit geval blijft de data hetzelfde en hoeft de gebruiker niet opnieuw toegevoegd te worden aan de groep. Deze check is er omdat als je gebruikers in een groep aanpast hij eerst alle gebruikers verwijderd in de groep voordat hij ze weer toevoegt. Het verwijderen van gebruikers gaat hetzelfde als het verwijderen van groepen, hiervoor is er ook een methode in de interface IOnlineStorage die heet RemoveUserFromGroup. Hieronder zie je een groep met toegevoegde gebruikers en dezelfde groep in Azure met dezelfde gebruikers.

Hieronder zie je de Azuregrpn-groep met toegevoegde gebruikers, en dezelfde groep in Azure met gebruikers. Niet alle gebruikers hebben een e-mailadres of een e-mailadres dat is gekoppeld aan dezelfde Azure-tenant.

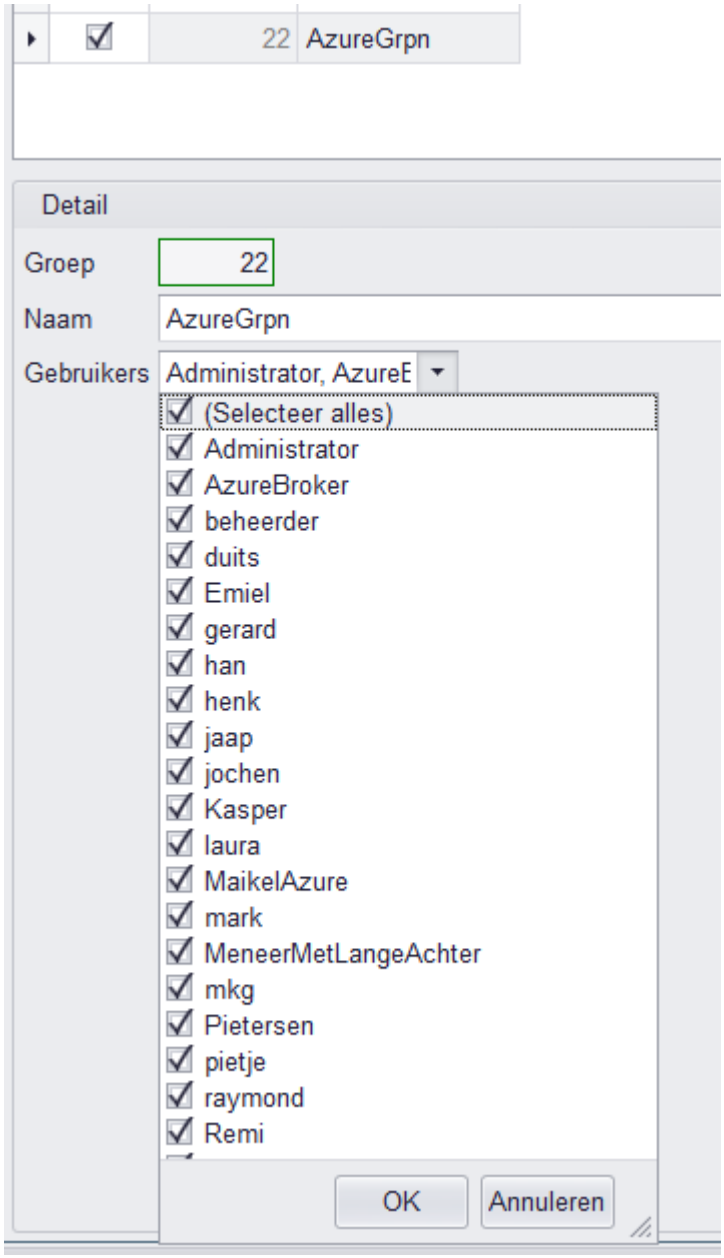


Figure 22 gebruikers die zijn toegevoegd in een MKGV groep

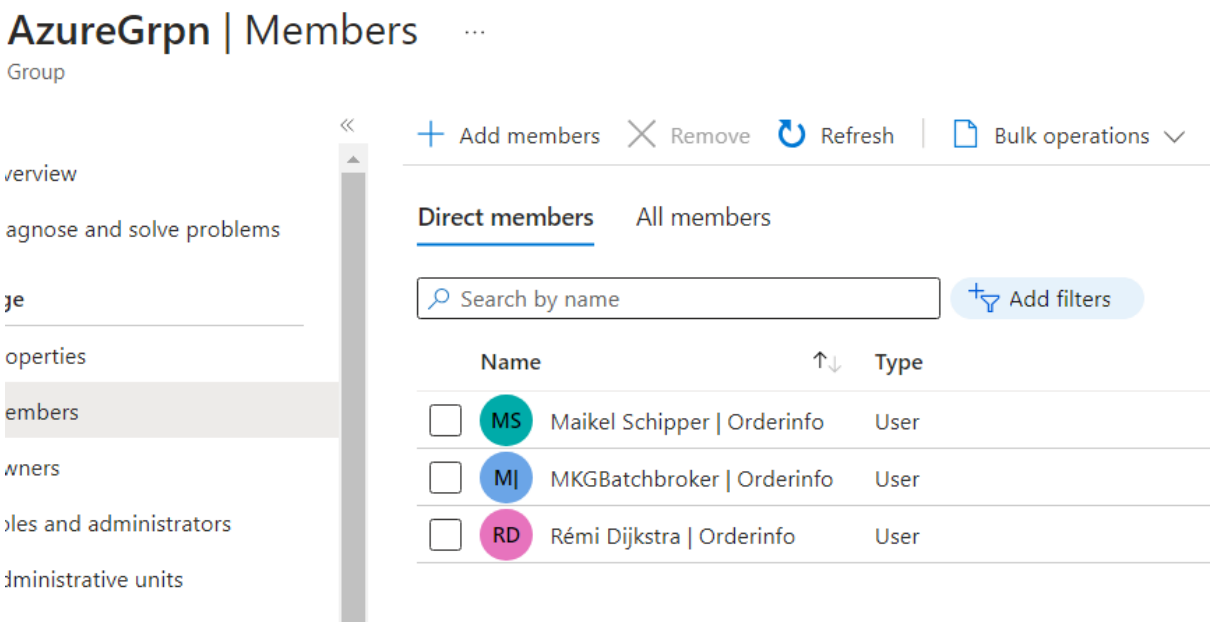


Figure 23 een MKGV groep met gebruikers in Azure

8.2.6. Document categorieën en toegang

Nadat de groepen gedaan zijn ben ik begonnen met de document categorieën. De implementatie hiervan is hetzelfde als die voor gebruikers groepen maar dan in het bl-dcat bestand, ook hier heb ik de UpdateRecord en DeleteRecord aangepast. Hieronder zie je een export van de document categorieën en de categorieën in Azure als security groepen.

Hieronder zie je een export van document categorieën met een externe ID.

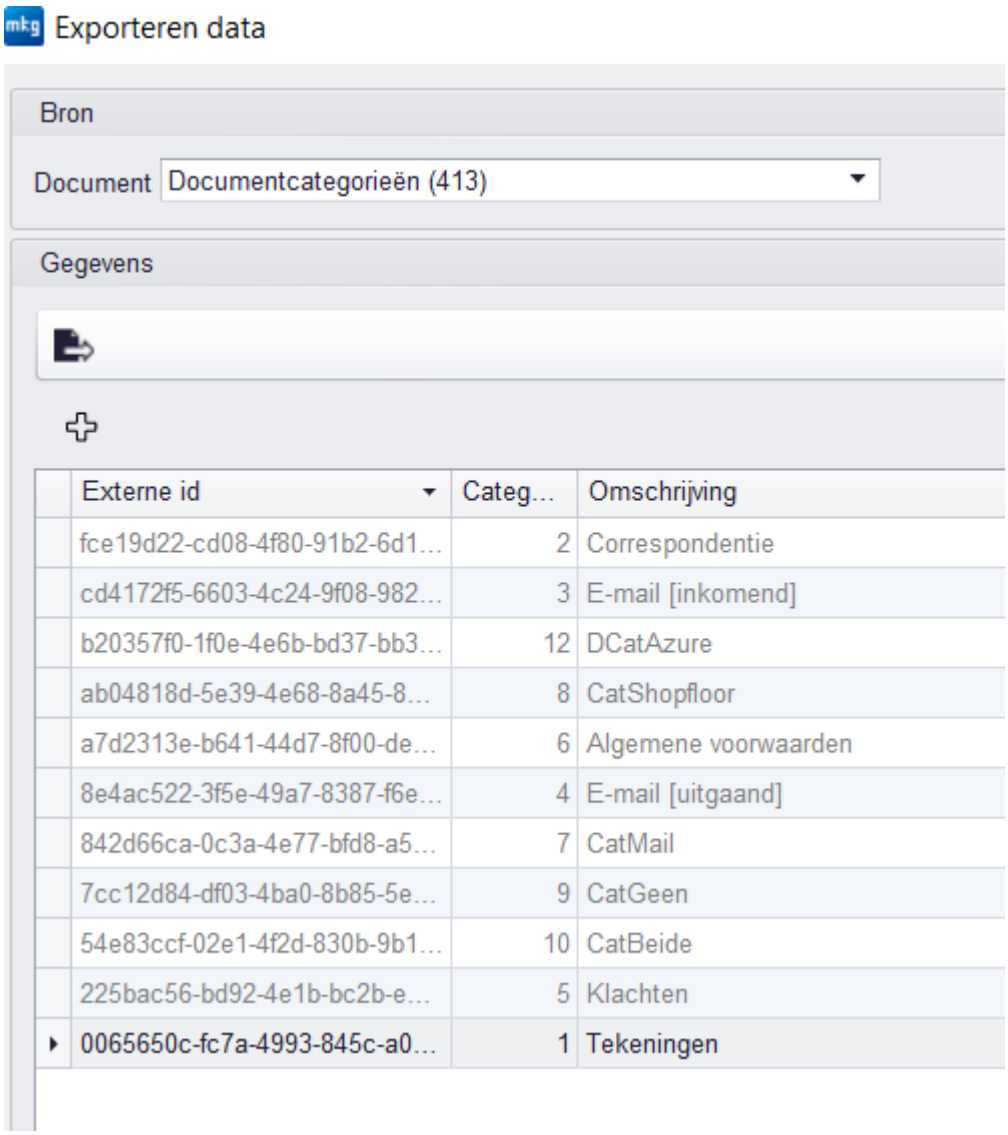


Figure 24 export van de bestaande categorieën in MKGV met een externe id

Toen de implementatie van document categorieën synchroniseren naar Azure klaar was ben ik begonnen met de groepen rechten geven aan bestanden. Dit heb ik gedaan door terug te gaan naar de UploadFile methode in de UtilFiles class, het delen van het bestand gebeurt voordat de categorieën worden toegevoegd aan de custom kolom. Dit wordt gedaan door een Post request te doen naar de invite link. Hieronder ze je de groepen die toegang hebben tot de voorgaande geüploade bestand.

Hieronder zie je de groepen waarmee het eerdere bestand is gedeeld.

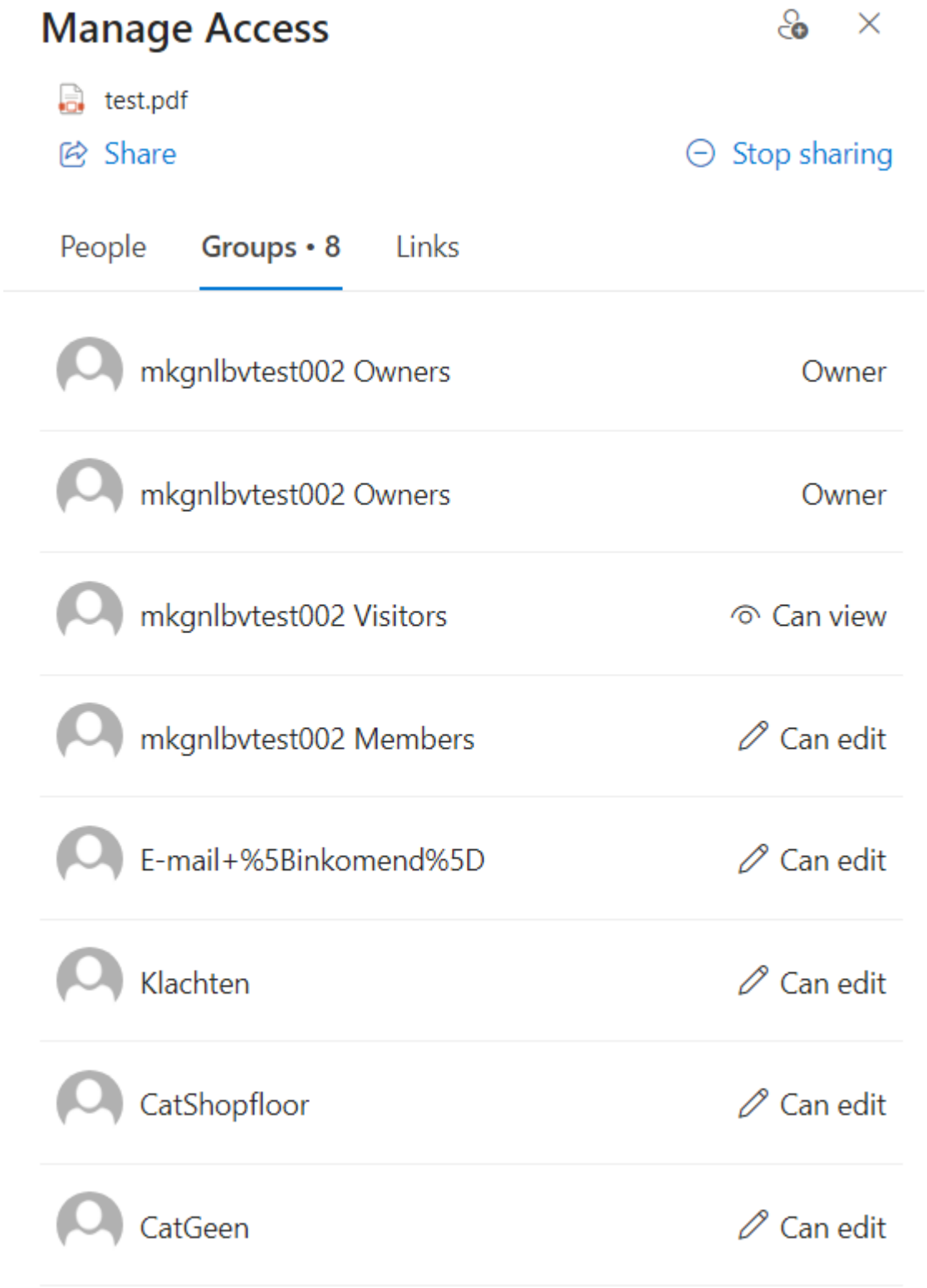


Figure 25 de categorieën die toegang hebben tot een bestand

8.2.7. Autorisatie groepen

Nadat ook het delen van bestanden was geïmplementeerd ben ik begonnen met het synchroniseren van de autorisatie uit MKGV. De autorisatie in MKGV gebeurt in de koppeltabel aumd, dat betekent dat ik code moet toevoegen in de bl-aumd bestand. Voor de autorisatie heb ik de UpdateRecord aangepast om groepen of individuele gebruikers toegang te geven de document categorie groep in Azure. Dit doe ik door te kijken of de link naar groep of gebruiker is gevuld. Ook heb ik de DeleteRecord methode aangepast om ze groepen of gebruikers te verwijderen. Hier heb ik niet een extra check te hoeven inbouwen omdat alle data die gelinkt is aan een document categorie niet wordt verwijderd bij elke verandering. Hieronder zie je de autorisatie overzicht in MKGV en de dcatAzure groep in Azure met de opgegeven rechten.

Hieronder zie je de autorisatie tabel in MKGV en de dcatAzure groep in Azure met dezelfde data als MKGV heeft

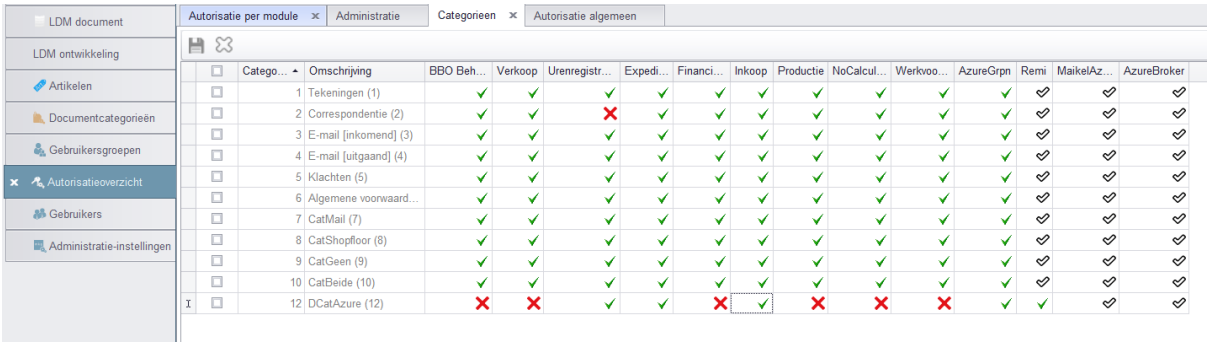


Figure 26 de autorisatie overzicht in MKGV

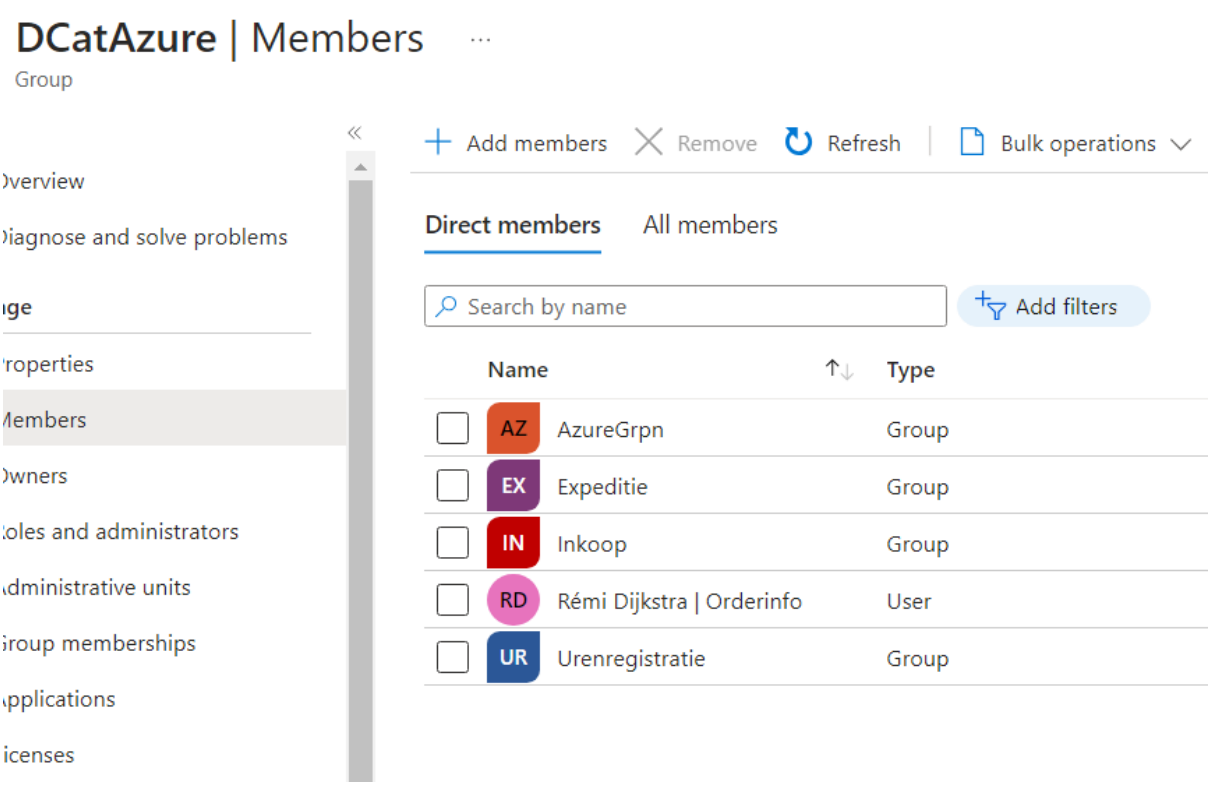


Figure 27 een document categorie in Azure met de correcte autorisatie

Het laatste dat is overgebleven was de optie om alles in een keer te synchroniseren, we synchroniseren niet bestanden alleen maar de groepen, document categorieën en de autorisatie.

Hiervoor heb ik in de bl-dcat, bl-grpn, bl-gegr en bl-aumd een synchronisatie methode toegevoegt, deze methodes lopen over alle records en kijken of hij een externe ID heeft. Als er geen externe id is maken we een nieuwe Azure groep aan, ook checken we of de namen nog steeds overeen komen als dat niet zo is updaten we de naam van de Azure groep zodat het in lijn is met MKGV. Ook voegen we nog een keer de gebruikers of groepn toe aan de Azure Groepen, als een gebruiker of groep als vast zit aan een Azure groep dan voegt Azure hem niet nog een keer toe.

8.2.8. Bugs.

Nadat ik klaar was met de groepen, document categorieën en het autorisatie ben ik er achter gekomen dat er problemen waren met de Post, Put, Patch en Delete requests. Het probleem was dat als we meer dan 3 van die requests richting Azure groepen deden dan liep de WebResponse vast en kwam hij met een timed out exception,. Daarna konden er geen nieuwe requests uitgevoerd worden. Na wat onderzoek heb ik een andere manier gevonden om het probleem op te lossen, de oplossing was door gebruik te maken van batch requests. De batch requests die we sturen naar Microsoft Graph lopen niet vast en worden gewoon uitgevoerd. Een batch request heeft een limiet van 20 requests die hui tegelijk kan uitvoeren dus hou ik bij hoeveel er zijn opgeslagen, als ik bij 20 aankom stuur ik de batch request naar Microsoft Graph, anders als methodes die gebruik maken van batch klaar zijn en niet 20 requests hebben toegevoegd word de batch alsnog verstuurd naar Microsoft Graph.

8.3. POC Outlook

Ik ben wel begonnen aan de POC Outlook maar heb er niet veel voortgang in kunnen boeken. Het enige wat er staat zijn de methodes om requests uit te voeren naar Microsoft Graph, deze methodes zijn getest door ze uit te voeren via ABL scratchpad met vaste data. Ook heb ik een klein beetje interface gedaan rondom het ophalen van e-mails. Jammer genoeg heb ik niet meer tijd gehad om er verder mee te gaan.

9. Conclusie

Alle vereisten met betrekking tot bestanden op SharePoint zijn behaald, met uitzondering van de periodieke synchronisatie. Helaas was er onvoldoende tijd beschikbaar om ook de integratie met Office 365 te voltooien. De basis hiervoor is aanwezig in de vorm van de HTTP-requests die moeten worden uitgevoerd, maar er was niet genoeg tijd beschikbaar om daar aan te beginnen. Dit komt omdat er zich onverwachte problemen voordeden tijdens de implementatie van de SharePoint integratie. Daarnaast moest ik eerst het framework van MKG nog leren en dat kostte meer tijd dan verwacht.

Ondanks dat niet alle taken voltooid zijn is er wel antwoord gegeven op alle deelvragen in de vele (voor)onderzoeken. De vraag betreffende de toegangstokens van Microsoft is beantwoord en toegepast. Ook is er opheldering over de overdracht van de MKGV veiligheid naar het nieuwe systeem. Bovendien is er nu informatie beschikbaar over apis, .net libraries en het taggen van bestanden in SharePoint

De implementatie is echter zo generiek mogelijk gemaakt, zodat MKG eenvoudig een ander online documentbeheersysteem kan implementeren. Alle code is getest tijdens de ontwikkeling en wordt opnieuw getest door het bouw- en testteam binnen MKG. Zij zullen ook een andere gebruikersinterface maken voor de benodigde instellingen om de nieuwe functionaliteit operationeel te maken.

Hoewel het opgeleverde niet volledig overeenkomt met de doelen die vooraf afgesproken zijn, is MKG positief over de uitkomst van mijn afstudeerproject. Het was een project waar al lang behoefte naar was. Een deel daarvan is nu functioneel en leverbaar; namelijk het uploaden en delen van bestanden via SharePoint. Dit was een belangrijk deel van de gewenste functionaliteit. Bij MKG zijn ze dus tevreden over het werk dat ik heb geleverd.

Nadat alles is getest en de nieuwe gebruikersinterface is gemaakt, wordt het uitgerold naar bèta-klanten die ermee gaan werken en het onder normale omstandigheden kunnen testen. Nadat het door de bèta-klanten is getest, wordt het uitgerold naar alle klanten.

De tabel hieronder toont de behaalde en niet-behaalde vereisten.

9.1. behaalde requirements

De volgende tabel toont de behaalde requirements.

Req	Omschrijving	FR/NFR	Prioriteit	Behaald
RQ01	Gebruikers moeten data van Azure kunnen invullen om connectie te kunnen maken.	FR	M	Y
RQ02	Gebruikers moeten bestanden kunnen uploaden naar SharePoint.	FR	M	Y
RQ03	Gebruikers moeten doormiddel van categorieën autorisatie kunnen toevoegen	FR	M	Y
RQ04	Gebruikers moeten gemakkelijk bestanden moeten terug kunnen vinden in SharePoint	NFR	M	Y
RQ05	Gebruikers moeten SharePoint bestanden kunnen verwijderen, openen of printen via MKGV	FR	M	Y
RQ06	Gebruikers moeten een SharePoint site en drive kunnen kiezen waar bestanden worden geplaatst	FR	M	Y
RQ07	Data uit MKG moet leidend zijn	NFR	M	Y
RQ08	Groepen en document categorieën moeten periodiek gesynchroniseerd worden	NFR	S	Y/N
RQ09	Gebruikers e-mails moeten niet meer via de outlook app opgehaald worden	NFR	C	N
RQ10	E-mails moeten niet meer via de outlook app verstuurd worden	NFR	C	N
RQ11	Contacten uit MKGV moeten gesynchroniseerd worden met Office365	NFR	C	N
RQ12	Kalender items uit MKGV moeten gesynchroniseerd worden met Office365	NFR	C	N
RQ13	Kalender items en contacten moeten periodiek gesynchroniseerd worden	NFR	C	N
RQ14	Documenten die al in het systeem staan moeten ook naar SharePoint gesynchroniseerd worden	NFR	W	N

10. Aanbevelingen

Om de Office 365 integratie af te maken moet het volgende nog gebeuren:

- periodiek de Synchronisatie methode laten draaien
- Als gebruiker inloggen bij Azure
- E-mails ophalen via Microsoft Graph
- E-mails versturen via Microsoft Graph
- Contacten uit MKGV synchroniseren naar Office 365 via Microsoft Graph
- Kalender items uit MKGV synchroniseren naar Office 365 via Microsoft Graph

Hier is nog geen code voor maar wel de http requests die je zou moeten doen naar Microsoft Graph, deze requests staan allemaal gedocumenteerd in een Postman workspace die gedeelt is met iedereen binnen MKG.

11. Reflectie

Naar mijn gevoel verliep het project wel goed ook al waren er dingen die ik beter had kunnen doen rondom planning en voorbereiding. Mijn eerste planning was wat onduidelijk en volgde ik niet altijd even goed. Dit zorgde er voor dat sommige geplande activiteiten wat later werden uitgevoerd dan ik gewild had. Dit kan opgelost worden door een volgende keer eerder te beginnen met plannen en de planning te delen met meerdere mensen. Ook zou het helpen als de gemaakte planning wordt gebruikt bij de voortgangsgesprekken met mijn begeleiders en andere collega's.

Het voorbereiden was redelijk goed gegaan in het begin. Ik had een goed overzicht van wat gedaan moest worden en wat ik nodig had. Ik zou de volgende keer wel meer tijd kunnen in plannen om nieuwe systemen zoals Progress en het framework van MKG te leren. Hierdoor wordt de ontwikkeling niet gehinderd door een gebrek aan kennis over de te gebruiken systemen.

Wel vind ik dat ik er niet te lang mee bleef zitten, als ik problemen had of extra richting nodig had. Ik ging dan op tijd hulp zoeken van iemand met meer kennis en ervaring over het probleem waar ik mee zat. Ook vind ik dat contact met collega's rondom verschillende keuzes goed ging, als er meerdere keuzes waren zette ik de keuzes goed op een rij. Vervolgens ging ik praten met mensen met meer kennis om te overleggen welke optie de beste was, hierbij gaf ik ook mijn eigen voorkeur aan.

Naar mijn mening heb ik aangetoond competent te zijn op de vlakken van analyseren, ontwerpen en realiseren. Met de vele onderzoeken die ik heb gedaan heb getoond dat ik analytisch kan denken en te werk kan gaan. Ik heb requirements opgesteld voor mijn project en een analyse uitgevoerd voor de rirsico's op gebied van veiligheid met betrekking tot de toegangstokens van Microsoft. Het klassendiagram en databasemodel zijn goede weergaven van mijn kunnen op het gebied van ontwerpen. Bij de ontwerpen is rekening gehouden met al bestaande structuren en systemen. Voor mijn competenties in het realiseren hoef je niet verder te kijken dan de nieuwe geïmplementeerde functionaliteiten van MKGV.

Al in al was deze een leerzame en leuke ervaring geweest voor mij.