SharePoint Framework

&

**Microsoft Graph**

Programutvecklare inom SharePoint/.NET

Jonas Öhman

EC-Utbildning

Innehållsförteckning

Beteckningar 1

Sammanfattning 2

Förord 3

Inledning 4

Bakgrund 4

Syfte och frågeställning 4

Avgränsningar 4

Huvuddel 4

Analys av resultat 6

Avslutande diskussion 6

Referenser 7

# Beteckningar

SharePoint Framework (SPFx): Ett ramverk som tillåter utvecklare att anpassa SharePoint online gränssnittet samt bygga applikationer som är framtidssäkra och responsiva med kända tekniker som använder öppen källkod.

API: Ett sätt att dela data på ett kontrollerat och standardiserat sätt, t.ex. så har SL ett API som tillåter utvecklare av trafikappar att hämta viss information om tågtrafiken.

Microsoft Graph: Ett API som Microsoft använder för delning av data inom deras egna programvaror vilket gör det mycket framtidssäkert och som nu är öppet för utvecklare att integrera i sina lösningar.

REST: En formaterad http adress som gör anropet mot i detta fall ett API, svaret från detta anrop fås i form av JSON format.

JSON: Enkelt språk för hur data som hämtas via t.ex. REST anrop formateras för att lättare kunna behandlas av utvecklaren.

# Sammanfattning

Genom test av Microsofts egna Notisfunktioner i SharePoint insåg jag dess begränsning och valde därför att bygga en Extension i SharePoint Framework och genom att använda API anrop mot Microsoft Graph kunde jag skapa en egen Sidfot på min SharePoint, därefter sorterade jag den JSON sträng jag sedan mottog och kunde i detta fall visa Avsändare, Ämne samt innehåll i det senaste mottagna mailet i min Outlook, detta går självklart enkelt att byta ut mot annan information som kan vara av värde just för dig genom att anpassa REST anropet gärna via det enkla verktyget Microsoft Graph Explorer samt även genom enklare anpassningar av koden i själva SharePoint Framework Applikationen.

# Förord

I det dagliga arbete krävs det ofta snabb interaktion med datakällor på olika platser och i olika program, genom att kunna skapa applikationer som är lätta att anpassa och användandet av Microsofts egna API kan denna SharePoint Framework Extension förenkla arbetet och göra att du själv kan styra vad du vill ha för information och var det presenteras genom enkla medel.

# 

# Inledning

Ett arbete gjort för att visa på enkelheten och kraften i att skapa applikationer i SharePoint framework med insamling av data från Microsoft Graph.

Genom att skapa applikationer i SharePoint framework och därigenom importen av inbyggda funktioner för att hantera Microsofts egna API kallat Microsoft Graph, kan man skapa applikationer som har åtkomst till många olika datakällor inom Office 365 och utnyttja dessa källor för skapandet av kraftfulla applikationer för att effektivisera och underlätta det dagliga arbetet genom att slippa byta hemsida och program för att se denna viktiga information.

## Bakgrund

Under vår kurs i SharePoint Framework så såg jag potentialen i ramverket och hur effektiv utveckling kan ske genom att många funktioner finns inbyggda vilket även gör det enkelt för juniorutvecklare att skapa kraftfulla applikationer.

när jag sedan även upptäckte Microsofts API och hur stor tillgång till olika datakällor inom Office 365 man har tillgång till insåg jag att nyttan av att kombinera dessa två skulle vara stor genom enkelheten att implementera detta API i SharePoint framework applikationer och extensions.

## Syfte och frågeställning

Syftet med examensarbetet är att visa på enkelheten i utveckling med SharePoint Framework samt bredden i att använda Microsofts egna API kallat Microsoft Graph. Kan detta användas som en grund för att enkelt vidareutveckla applikationen för just ditt behov av data?

## Avgränsningar

Jag har lyckats lösa det jag från början hade som plan att skapa dock så har det under utvecklingens gång växt fram ytterligare funktioner och möjligheter jag velat implementera men detta får ske på egen hand utanför detta projekt, det jag hade velat skapa ytterligare är funktionen att kunna markera ett E-post meddelande som läst samt även funktion att kunna ta bort mailet från mailkorgen.

# Huvuddel

Del 1:

I första skedet undersökte jag hur Microsofts egna notiser gällande inkommande mail fungerade vid arbete i en SharePoint miljö och insåg ganska snart att den inte fyllde de önskemål jag hade på hur snabbt man vill ha information om inkommande mail, detta medförde att jag började kolla på möjligheten att lyfta in Outlooks funktionalitet direkt in i SharePoint dels för att slippa byta program för att hantera sina mail och dels för att underlätta det dagliga arbetet.

Det första jag gjorde var att använda Microsoft Graph Explorer för att se att de data jag ville ha åtkomst till går att hämta via REST anrop vilket självklart gick, det jag sedan behövde göra var bestämma i vilken form denna data skulle presenteras och i och med att jag tyckte om enkelheten i SharePoint framework valde jag att bygga en extension och inte en web part, detta för att nyttan blir större om man har tillgång till informationen över hela sin SharePoint site och inte bara som en inlagd applikation på en enda sida i miljön.

Min extension som jag valde att bygga i form av en anpassad sidfot skulle därmed följa med användaren även vid navigering på olika sidor i SharePoint siten och hela tiden visa det senast inkomna e-post meddelandet i användarens inkorg.

Del 2:

När jag började bygga min extension följde jag den officiella guide som Microsoft tillhandahåller för hur man genererar ett nytt projekt, detta skapar en anpassad struktur och den grund man behöver för projektet och de val man gör under genereringen anpassar projektet för det ändamål man har.

För att få tillgång till funktioner i SharePoint framework som gör kommunikation till Graph API: et möjligt måste man importera dessa genom en så kallad Namespace.

import { AadHttpClient, HttpClientResponse } from "@microsoft/sp-http";

Ovanstående Namespace är de jag använde I projektet och är det som möjliggör de REST anrop jag behöver göra mot Graph API: et,

Observera att det finns ett specifik Namespace för Graph kallad ”MSGraphClient” som jag valde att inte använda då denna fortfarande är i förstadie och kan komma att ändras med nya releaser av SharePoint framework.

Nästa steg var att se till att skapa REST anropet för att erhålla den data jag senare vill visa i min extension, detta gjordes genom att jag besökte Microsoft Graphs hemsida där ett verktyg kallat Graph Explorer finns varpå man kan skapa en REST adress som hämtar just den information jag vill ha och formaterar denna enligt JSON.

Den adress jag slutligen använde blev den enligt nedan då jag ur denna länk får med mig mycket alternativ data vilket gör den mer framtidssäker om man vill bygga vidare med nya funktioner och anpassningar.

"https://graph.microsoft.com/v1.0/me/mailFolders/Inbox/messages/delta"

Nästa steg var att definiera i package-solution.json vilka resurser som SharePoint ska ge tillåtelse till att hämta information från, detta görs via nedanstående kod som säger åt SharePoint online att skapa en förfrågan i Administratörscentret för att ge administratören en fråga om att öppna upp för ett scope i API: et, i detta fall behövde jag bara öppna upp för att läsa E-post hos den inloggade användaren.

"webApiPermissionRequests": [

{

"resource": "Microsoft Graph",

"scope": "Mail.Read"

}

],

När man sedan publicerar denna extension så kommer applikationen starta en förfrågan i SharePoints administratörcenter under fliken API-Hantering där administratören sedan kan acceptera eller avslå applikationens förfrågan.

DEL3:

Vid skapandet av det visuella så läste jag till en början en del guider som Microsoft tillhandahåller samt vissa blogginlägg angående SharePoint framework extensions och fick därigenom den kunskap som jag behövde för att skriva koden som skapar upp och verifierar den anpassade sidfoten, det som gjorde det lite svårare var att mycket av den dokumentation som finns tillgänglig inte längre är aktuell efter att en ny version av SharePoint framework släppts och därigenom gjort vissa Namespaces inaktuella vilket gör att man får anpassa koden en del för de nya Namespaces som tillkommit och som är aktuella i dagsläget.

För att förbättra det visuella delarna i min extension så använder jag mig av en CSS fil som sköter en del av hur elementen presenteras och stylas, i tillägg används en liten del av Office-UI-fabric för att skapa snygga ikoner på min sidfot.

# Analys av resultat

Applikationens skapande var ganska enkelt överlag och det som jag hade problem med under utvecklingens gång var i stort två saker, det ena var att jag lätt kunde få fram all data från mitt restanrop dock så när jag ville gå ner i detalj och välja ut mer specifikt vilka delar av informationen jag ville använda kunde jag till en början inte få ut denna, detta berodde på ett fel i koden där jag var tvungen att likt en array välja att specifikt plocka ut det första objektet, för det andra så hade jag problem med stylingen där jag inte kunde få funktionen toggleClass i JQuery att fungera till en början varpå min CustomFooter gick att minimera men inte att expandera efter minimeringen.

Detta kunde efter lite felsökning härledas till att ett klickevent på sidan kördes två gånger varpå toggleClass självklart utgick till utgångsläget hela tiden, detta kunde jag lösa genom att innan min toggleClass använda funktionen Unbind varpå min funktion med toggleClass fungerar igen och skiftar mellan två olika CSS klasser som stylar om min customFooter.

Själva skrivandet av kod var även det lite utmanande innan man fick till de Namespaces som används då dessa har bytts ut när SharePoint Framework uppdaterades till en ny version, dessa nya Namespaces fungerar lite annorlunda och koden behövde därmed anpassas efter detta vilket krävde lite utforskande av bloggar samt testande innan jag fick det att fungera, den lättaste delen var användandet av Microsoft Graph där implementationen i koden gick smidigt tack vare den dokumentation som Microsoft tillhandahåller med steg för steg instruktioner, jag ser dock att man bör vara varsam med rättigheterna då Microsoft Graph är ett API som kan ge användaren mycket tillgång till data inom ens Office 365 och är man inte vaksam som utvecklare kan man ge användaren tillgång till att ta bort data från sina resurser.

Som tur är kan man som TenantAdmin se exakt vilken tillgång som olika SharePoint Framework applikationer har och även ta bort dessa rättigheter om de getts ut felaktigt.

# Avslutande diskussion

Det jag satte ut att göra har jag lyckats med i stort, jag ville bygga en extension i SharePoint Framework som skulle visa det senast inkomna mailet i den inloggade användarens Outlook med hjälp av Microsoft Graph API, detta då jag ser hur kraftfullt detta API är och hur lätt det är att anpassa de REST anrop man gör mot detta API för att få fram de specifika data man vill ha oavsett om det är ens egna mailkorg, ens grupper i Teams eller tillgång till listor i SharePoint.

Det jag lärt mig under detta projekt är framförallt att använda kända tekniker och funktioner där det finns gott om referensmaterial annars kan det som juniorutvecklare vara svårt att komma i mål utan att det tar väldigt lång tid, mycket tid gick åt till att läsa bloggar och teknisk dokumentation om SharePoint Framework för att ge mig den grundläggande kunskapen för att bygga hela applikationen och även hur man integrerar funktioner att kunna kalla på Microsoft Graph, att sedan använda CSS för att styla min applikation tyckte jag var en av de enklare punkterna då det endast krävdes en liten insats för att få till det utseende jag ville ha.

# Referenser

Internetkälla: <https://developer.microsoft.com/en-us/graph/graph-explorer>

Internetkälla: <https://www.tonyishere.co.uk/tag/spfx/>

Internetkälla: <https://github.com/SharePoint/sp-dev-fx-extensions/>

Internetkälla: <https://blogs.u2u.be/u2u/post/sharepoint-framework-spfx-using-the-new-microsoft-graph-client-from-an-application-extension>

Internetkälla: <https://www.techmikael.com/2017/06/accessing-microsoft-graph-resources.html>

Internetkälla: <https://docs.microsoft.com/en-us/sharepoint/dev/spfx/extensions/get-started/using-page-placeholder-with-extensions>