CHALMERS

GÖTEBORGS UNIVERSITET

Migrering till en molnbaserad e-posttjänst för optimering av QBIS Service Desk

Fördelar och utmaningar med en molnbaserad lösning

Examensarbete inom Data- och informationsteknik

Mustafa Bawi

Abbas Faizi

Institutionen för Data- och Informationsteknik

CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2023
www.chalmers.se

Examensarbete 2023

Migrering till en molnbaserad e-posttjänst för optimering av QBIS Service Desk

Fördelar och utmaningar med en molnbaserad lösning

Examensarbete inom Data- och informationsteknik

Mustafa Bawi

Abbas Faizi

CHALMERS

Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
Göteborg, Sverige 2023

Migrering till en molnbaserad e-posttjänst för optimering av QBIS Service Desk
Fördelar och utmaningar med en molnbaserad lösning
Mustafa Bawi
Abbas Faizi

Handledare: Jonas Almström Duregård, Institutionen för Data- och informationsteknik
Examinator: Lars Svensson, Institutionen för fysik

Examensarbete 2023
Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Institutionen för Data- och Informationsteknik
Göteborg, Sverige 2023

Migrering till en molnbaserad e-posttjänst för optimering av QBIS Service Desk
Mustafa Bawi & Abbas Faizi
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola

Abstract

This thesis project aimed to explore migrating a local mail server to the cloud in purpose of optimizing the service desk for QBIS. A large part of the project was spent researching QBIS's existing infrastructure, service desk requirements, and the potential benefits and challenges of migrating to the cloud. In this project, Microsoft Azure was the cloud provider used for QBIS's needs. The project involved several steps, including setting up the cloud-based environment, implementing a SaaS application, and configuring cloud services to be integrated with the QBIS service desk. The SaaS application was not launched into production due to being unfinished in some areas. However, large parts of the structure and the requirements of the application are in place. The SaaS application is successfully in place with all Microsoft Cloud services and is fetching email data from the Microsoft Graph API. The application is also able to handle email errands to the webAPI of QBIS and is integrated with the QBIS environment. The findings of this project provided valuable insights for the company and also for other companies that are considering migrating their IT infrastructure to the cloud.

Keywords: Cloud services, Cloud Migration, local mail server, SaaS application, Service desk, Microsoft Azure, Microsoft Graph API.

Sammanfattning

Det här examensprojektet syftade till att undersöka effektiviteten av att migrera en lokal mailserver till molnet för att optimera service-desken för QBIS. En stor del av projektet ägnades åt att utforska QBIS:s befintliga infrastruktur, service-desk kraven och de potentiella fördelarna och utmaningarna med att migrera till molnet. I detta projekt användes Microsoft Azure som molntjänstleverantör för QBIS:s behov. Projektet innefattade flera steg, inklusive att konfigurera den molnbaserade miljön samt att konfigurera molntjänster för att integreras med QBIS service-desk. SaaS-applikationen lanserades inte i produktion på grund av att den var ofullständig inom vissa områden. Däremot är en stor del av strukturen och kraven för applikationen på plats. SaaS-applikationen är framgångsrikt implementerad med alla Microsofts molntjänster och hämtar mail-data från Microsoft Graph API. Applikationen kan även hantera mail-ärenden till QBIS:s webAPI och är integrerad med QBIS-miljön. Resultaten från detta projekt har gett värdefulla insikter för företaget samt för andra företag som överväger att migrera sin IT-infrastruktur till molnet.

Nyckelord: Molntjänster, Molnmigrering, Lokal-mailserver, SaaS-applikation, Service-desk, Microsoft Azure, Microsoft Graph API.

Förord

Vi är mycket tacksamma till QLogic AB som har gett oss möjligheten att få genomföra ett examensarbete under deras verksamhet. Bland annat riktar vi ett stort tack till Martin Augustsson som introducerade oss till arbetet och bjöd in oss till företaget. Vi vill också rikta ett stort tack till Erik Gjers som har varit av stor hjälp i det tekniska men även uppmuntrat oss gång på gång under arbetet. Därefter vill vi också tacka alla resterande medlemmar i QLogic AB som alltid välkomnat oss med öppna armar. Slutligen vill vi också rikta ett stort tack till vår handledare på Chalmers, Jonas Almström Duregård för hans råd och vägledning i rapportskrivandet.

Mustafa Bawi & Abbas Faizi, Göteborg, Juni 4, 2023

Ordlista

Nedan är en lista över akronymer som har använts i detta examensarbete listade i alfabetisk ordning:

AAD	Azure Active Directory
API	Application Program Interface
CRUD	Create, Read, Update, Delete
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IDE	Integrated Development Enviroment
MSAL	Microsoft Authentication Library
PaaS	Platform as a Service
REST	Representational State Transfer
SaaS	Software as a Service
SQL	Structured Query Language
URL	Uniform Resource Locator

Innehållsförteckning

Ordlista **ix**

Figurer **xiv**

1 Inledning **1**
1.1 Bakgrund . . . 1
1.2 Syfte . . . 1
1.3 Precisering av frågeställning . . . 2
1.4 Mål . . . 2
1.5 Avgränsningar . . . 2

2 Teknisk bakgrund **3**
2.1 Molnet . . . 3
2.1.1 Molnleverantörer . . . 3
2.1.2 Servicemodeller . . . 3
2.2 Databaser . . . 4
2.2.1 Structured Query Language . . . 4
2.2.2 CRUD operationer . . . 4
2.3 API . . . 5
2.3.1 HTTP . . . 6
2.3.2 HTTP-statuskoder . . . 6
2.3.3 Endpoints . . . 6
2.3.4 Webhooks . . . 6
2.3.5 Polling . . . 7
2.4 Microsoft Graph API . . . 7
2.5 Icewarp-mailserver . . . 8
2.6 Docker . . . 8
2.7 Autentisering . . . 8
2.7.1 Autentiseringsreferenser . . . 8
2.7.2 OAuth 2.0 . . . 8
2.7.3 Autentiseringsflöden . . . 9
2.7.4 Microsoft Authentication Library . . . 9
2.8 Azure Active Directory . . . 10
2.9 Kanban-metodiken . . . 10
2.10 Verktyg . . . 11
2.10.1 Visual Studio . . . 11

2.10.2 Programspråket C# . . . 11
2.10.3 .NET . . . 11
2.10.4 Microsoft SQL Server . . . 11
2.10.5 Microsoft Teams . . . 12
2.10.6 Atlassian-produkter . . . 12

3 Metod **13**
3.1 Översikt av arbetsprocessen . . . 13
3.2 Användning av Kanban-metodiken . . . 13
3.3 Planeringsfasen . . . 13
3.4 Microsoft Azure . . . 14
3.4.1 Krav på molnleverantören . . . 14
3.4.2 Pris . . . 14
3.4.3 Underhåll . . . 14

4 Systemkonstruktion **15**
4.1 QBIS nuvarande mailserver-integration . . . 15
4.1.1 Flödet för ett mail . . . 15
4.1.2 Mailserver-lösningen . . . 16
4.1.3 Email-integration-applikationen . . . 16
4.1.4 QBIS databas . . . 16
4.2 Integrationen med SaaS . . . 17
4.2.1 Mailflödet i Microsoft Azure . . . 17
4.2.2 Mail-integrationen med Microsoft Graph API . . . 18
4.2.3 Integrationen med webAPI . . . 18
4.2.4 QBIS produktionsdatabas . . . 18

5 Implementation **19**
5.1 Email-integration-applikationen . . . 19
5.1.1 Val av autentiseringsflöde . . . 19
5.1.2 Testning . . . 19
5.1.2.1 Enhetstester . . . 20
5.1.2.2 Integrationstestning . . . 21

6 Resultat **22**
6.1 SaaS-applikationen . . . 22
6.1.1 Autentisering . . . 22
6.1.2 Svarsmeddelande . . . 23
6.1.3 Uthämtning av det senaste mail-meddelandet . . . 23
6.1.4 Sändning till webAPI . . . 23
6.1.5 Hantering av misslyckade mail-meddelanden . . . 23
6.1.6 Lagring av felmeddelanden . . . 23
6.1.7 Stänga av container . . . 23
6.2 Containerlogiken . . . 24
6.3 Förbättringar jämfört med gamla lösningen . . . 24
6.3.1 Automatisering . . . 24
6.3.2 Minskat underhåll . . . 24

6.3.3 Skalbarhet 24
6.3.4 Flexibilitet 25
6.4 Ekonomiska faktorer 25

7 Diskussion **26**
7.1 Utmaningar 26
7.2 Fördelar och nackdelar 26
7.3 Inspiration för framtida molnlösningar 27
7.4 Etik och hållbarhet 27
7.5 Vidareutveckling 27

8 Slutsats **29**

Källförteckning **29**

Figurer

2.1 Figuren beskriver hur ett API interagerar med en klient som begär information från externa källor. API:t möjligör denna kommunikationen och skickar tillbaka informationen till klienten. 5
2.2 Figuren illustrerar hur processen för webhooks går till. I figuren visas det hur en händelse triggar igång en POST operation som skickar information vidare. Denna datan hanteras från POST operationen och applikationen som får informationen handlar utifrån detta. 7
2.3 Figuren illustrerar hur en autentiseringsprocess i MSAL går till. Först får applikationen en token från AAD. Denna token används sedan med en specifik endpoint för att få åtkomst till en resurs, i detta fallet är det Microsoft Graph. 10

4.1 Figuren illustrerar ett flödesdiagram för den lokala mail-lösningen. . . 15
4.2 Figuren illustrerar ett flödesdiagram för den molnbaserade SaaS-lösningen 17

5.1 Figuren illustrerar autentiserings flödet mellan två AAD i Microsoft Azure. Här visas det hur containern med programkoden befinner sig i ett separat AAD och autentiserar sig mot en applikation i QLogic AAD. 20

6.1 Figuren illustrerar hur flödet med app-logiken ser ut när Azure Container Instance startas och körs igång. 24
6.2 Figuren visualiserar månadskostnaden för mail-lösningen. 25

1
Inledning

Detta avsnitt har som syfte att förbereda läsaren inför arbetet. Här presenteras bakgrund, syfte, frågeställningar, mål samt avgränsningar för arbetet.

1.1 Bakgrund

QBIS Business Systems är en del av det större företaget QLogic AB. QBIS Business Systems är ett av Sveriges ledande tidrapporterings-, utläggs- och projekthanteringssystem. Företaget består av ett flertal anställda i olika områden som programmerare, kreatörer, relationsbyggare och problemlösare för att utveckla och bygga upp företaget [1].

För närvarande använder företag, inklusive QBIS, mail-meddelanden som sin primära metod för kommunikation med både kunder och anställda. Support-förfrågningar för QBIS service-desk kommer in via mail-ärenden. Dessa mail-meddelanden hanteras för att skapa nya ärenden samt erbjuda kundservice. För att hantera dessa mail-ärenden används ofta en service-desk-applikation som hanteringsverktyg.

Det finns många lösningar för mail-hantering på marknaden, både som lokala lösningar och molnbaserade lösningar. En molnbaserad lösning, även känt som SaaS (Software as a Service), indikerar att programvaran är tillgänglig som ett verktyg över Internet och administreras av ett tredjeparts-företag.

1.2 Syfte

Syftet med projektet är att genomföra en migrering från en lokal mailserver till en molnbaserad SaaS-modell. Genom denna migrering, strävar projektet efter att utforska och utvärdera de potentiella fördelarna och utmaningarna som en övergång till en molnbaserad mail-lösning kan medföra på en generell nivå.

Genom att genomföra migreringen kan projektet bidra till kunskapsutveckling inom området för molnbaserade mail-lösningar.

1.3 Precisering av frågeställning

Examensarbetet kommer att utgå från följande frågeställningar:

- Vilka förbättringar observeras utifrån skalbarhet, flexibilitet och underhåll med den molnbaserade SaaS-lösningen jämfört med den lokala mailservern?
- Vad finns det för ekonomiska fördelar med en molnbaserad SaaS-applikation?

1.4 Mål

Målet med examensarbetet är att flytta QBIS lokala mailserver till en SaaS-modell. Denna modell skall uppfylla krav som felsökning, hantering av mail-ärenden samt svarsmeddelanden till kunden. En viktig del av målet är att anpassa företagets service-desk-applikation till SaaS-modellen. Programmeringsspråk som service-desk-applikationen är byggt på ska också bytas till C# vilket är en primär del av målet med examensarbetet.

1.5 Avgränsningar

Examensarbetet avgränsar sig till följande punkter:

- Detta examensarbete kommer att avgränsa sig till att skala och anpassa SaaS-applikationen för mindre företag.
- Examensarbetet tar inte hänsyn till faktorer som säkerhet. Detta antas ligga under molnleverantörens uppgifter.
- Detta examensarbete avgränsar sig till Microsoft Azure som molnleverantör.

2
Teknisk bakgrund

Avsnittet presenterar relevant teori som ingår i examensarbetet. Här presenteras också en beskrivning på de tekniska verktyg som användes för att genomföra projektet.

2.1 Molnet

Molnet är ett begrepp inom datavetenskap som beskriver samlingen av datorresurser, programvaror och tjänster som görs tillgängliga via Internet. Mer specifikt är molnet idén om att tillhandahålla IT-tjänster där dessa tjänster sköts av någon annan. När en resurs eller tjänst är på molnet betyder det helt enkelt att den är lagrad på en virtuell dator över Internet istället för en hårddisk på en lokal dator [2].

2.1.1 Molnleverantörer

En molnleverantör är ett tredjepartsföretag som erbjuder molntjänster och molnlösningar till företag och organisationer. Dessa tjänster inkluderar IT-infrastruktur, applikationer eller lagringstjänster i molnet som leverantören tar hand om. Beroende på vad för preferenser och behov en kund har kan molnleverantören erbjuda specifika tjänster för situationen. I nuläget finns det en mängd olika molnleverantörer ute på markanden. De största är Amazon Web Services (AWS), Microsoft Azure och Google Cloud.

AWS startade sina tjänster år 2006. Detta satte i gång en spiral i marknaden där andra företag snabbt tog efter i trenden att erbjuda tjänster över molnet. Därefter kom Googles version ut på marknaden år 2008 som kort därefter följdes av Microsofts version år 2010 [3]. Sedan följde mindre företag in i marknaden som exempelvis IBM och Oracle.

2.1.2 Servicemodeller

När man diskuterar molnleverantörer och deras erbjudna tjänster, är det vanligt att man hänvisar till tre huvudtyper av molntjänster: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) och Software as a Service (SaaS). Dessa är tre olika servicemodeller som beskriver vilka tjänster som erbjuds under den givna modellen. De olika servicemodellerna erbjuder olika delar av infrastrukturen i en molntjänst.

IaaS är en servicemodell som ger företag och organisationer tillgång till den grundläggande infrastrukturen i molnet. Detta inkluderar servrar, nätverk och lagring som då kan användas och utnyttjas av kunden för sina egna behov. I denna modell så ansvarar kunden själv för dessa tjänster. PaaS erbjuder istället en komplett utvecklingsmiljö för kunden som då kan användas för att bygga eller testa en tjänst som företaget är intresserad av. Bland annat är databashantering, ramverksstöd och verktyg för versionshantering vanliga områden som erbjuds i denna modell. Inte heller i PaaS behöver kunden ta hand om infrastrukturen utan detta sköts av molnleverantören. SaaS är den mest kompletta servicemodellen; här erbjuds kompletta applikationer och tjänster som körs direkt hos molnleverantören. Under SaaS så har molnleverantören allt ansvar för tjänsten och underhåller samt uppgraderar den. Exempel på SaaS tjänster är Microsoft Office 365, Zoom och Dropbox [4].

2.2 Databaser

En databas är en samling av information som lagras och hanteras i en digital miljö. Exempel på information som kan lagras i en databas är användarinformation, information om en viss produkt eller annan relevant information som en organisation lagrar. I princip kan man lagra vilken information som helst i en databas så länge den korrekta miljön är uppsatt för innehållet [5].

2.2.1 Structured Query Language

Structured Query Language, känt i industrin som SQL, är ett interaktivt programmeringsspråk som används för att hantera relationsdatabaser. Uppbyggnaden av en relationsdatabas är baserat på tabeller som relaterar till varandra. SQL används för att göra dessa relationer möjliga och bygger upp logiken i en databas. [6]

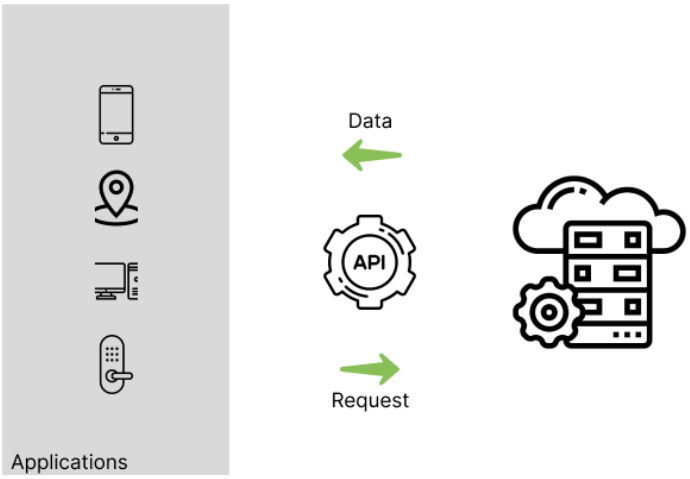
2.2.2 CRUD operationer

CRUD definierar de fyra operationerna CREATE, READ, UPDATE och DELETE som används för att utföra operationer i en digital miljö. CREATE operationen tillåter användaren att skapa ny data i exempelvis en databas. Förslagsvis kan en ny rad introduceras i en tabell inom en relationsdatabas där lagring av ny information ska insättas. READ operationen fungerar som en sökfunktion. READ tillåter användaren att söka efter specifik data och läsa av denna data för olika ändamål. Om användaren exempelvis vill hitta information om en viss anställd i en organisation kan READ operationen utnyttjas för detta. UPDATE operationen möjliggör uppdatering av existerande data. Bland annat kan ett telefonnummer för en kund i en organisation uppdateras med hjälp av UPDATE operationen. Den sista operationen DELETE i CRUD-definitionen används för att ta bort existerande data från en databas. Operationen används exempelvis om en användare slutar på ett företag och behöver då tas bort från organisationen [7].

2.3 API

API (Application Programming Interface) är en gränssnittsteknik som används för att skapa förbindelse mellan olika mjukvarusystem. Det är en uppsättning protokoll, regler och verktyg som gör det möjligt för olika program att kommunicera med varandra och utbytta information. Utvecklare kan skapa applikationer som drar nytta av befintliga tjänster och data som tillhandahålls av andra program och system genom att använda API:er.

API-lösningar består vanligtvis av två olika typer av komponenter. En typ av komponent exponerar API:er, vilket innebär att de gör API:erna tillgängliga för andra system eller applikationer att använda. Den andra typen av komponenter konsumerar API:erna, vilket innebär att de använder API:erna för att till exempel ansluta till datakällor eller tjänster som är centralt placerade i företagets backend. De exponerade API:erna finns vanligtvis på server-sidan, det vill säga i molnet eller på en lokal server. Klienterna är de API-konsumerande komponenterna. Dessa klienter är oftast mobilapplikationer, webbläsare eller inbäddade enheter [8].



Figur 2.1: Figuren beskriver hur ett API interagerar med en klient som begär information från externa källor. API:t möjligör denna kommunikationen och skickar tillbaka informationen till klienten.

2.3.1 HTTP

Hypertext Transfer protocol (HTTP) är fundamentet för WWW (World Wide Web). Protokollet HTTP används för att ladda webbsidor genom att använda länkar. HTTP kommunicerar mellan en användare och en webbsida som körs hos en webbserver. Denna kommunikation görs genom HTTP-anrop via operationer som GET och POST. GET hämtar information till användaren och POST skickar ut information till servern som användaren kommunicerar med. I en HTTP-begäran finner man olika delar, bland annat finner man versionstypen, URL, HTTP-metoden, body och en förfrågningsrubrik. Dessa delar beskriver hur en HTTP-begäran ska utformas samt vilket syfte den specifika begäran har mot användaren eller webbservern [9].

2.3.2 HTTP-statuskoder

En HTTP-statuskod är en respons i form av ett tre-siffrigt tal för att indikera hur en HTTP-begäran har slutförts. Det finns fem olika kategorier av statuskoder som kan returneras till användaren. Responsmeddelandena är numrerade mellan 1-5 följt av två stycken siffror till. Meddelanden som börjar med 1 indikerar att begäran har mottagits och håller på att behandlas. De som börjar med 2 betyder att begäran har godkänts och behandlats. Ett tal som börjar med 3 innebär att det krävs fler åtgärder för att hantera den begäran som genomförts. En statuskod som börjar med 4 indikerar att ett fel har uppstått på klientens sida. Statuskoder som börjar med 5 pekar däremot att ett fel har skett på serverns sida [9].

2.3.3 Endpoints

Inom området kring API:er är en endpoint en punkt som möjliggör kommunikation med det givna API:et. En klient som vill kommunicera med ett API för ens egna ändamål använder sig av API:ets endpoints för kommunikationskanalen. Att förändra en eller flera endpoints i ett API kan påverka hela arkitekturen för hur klienterna kommunicerar med det [10]. Exempelvis kan https://example.com/users vara en endpoint som returnerar information om alla användare. När kommunikationen då görs med en GET operation via denna endpoint får man tillbaka den informationen som API:et returnerar.

2.3.4 Webhooks

Webhooks är HTTP-begäranden som utlöses av en händelse som sker i ett system. Webhooken notifierar därefter ett destinationsystem som ger en respons på händelsen som inträffat. Webhooks körs automatiskt och användas för att möjligöra eventbaserade kommunikationssystem. Webhooks är vanliga i SaaS-plattformar och tas emot som GET eller POST metoder beroende på situationen [11].

Exempelvis kan webhooks användas i ett mail-system då de kan notifiera en destinationsaddress när ett nytt mail kommit in i en mail-inbox.

Figur 2.2: Figuren illustrerar hur processen för webhooks går till. I figuren visas det hur en händelse triggar igång en POST operation som skickar information vidare. Denna datan hanteras från POST operationen och applikationen som får informationen handlar utifrån detta.

2.3.5 Polling

Polling är en process där regelbundna kontroller genomförs för att samla in informationen och data från en enhet eller en tjänst. Till exempel kan polling användas i en programkodsfunktion för att kolla statusen för en mail-inbox, på så sätt genomförs operationer med ett godtyckligt intervall [12]. Polling sker automatiskt och konfigureras att utlösas på en viss tid. Sammanfattningsvis är polling ett viktigt verktyg för att övervaka och automatisera funktionalitet i mjukvarulösningar.

2.4 Microsoft Graph API

Microsoft Graph API är ett REST API som tillåter en användare att kommunicera med Microsofts molntjänster. Ett REST API är ett API som använder HTTP-metoder för sin primära kommunikation [13].

Microsoft Graph är porten till alla tjänster som Microsoft erbjuder i sitt moln. Dessa tjänster används av företag runt omkring i världen för sina kunder eller anställda. Ett företag kan även med hjälp av Microsoft Graph bygga applikationer för sin organisation som exempelvis en molnbaserad mail-lösning. Genom Microsoft Graph får man tillgång till data som ingår i Microsoft 365 tjänster. Dessa tjänster är bland annat Microsoft Kalender, Outlook, Exchange, Teams och mycket annat [14].

Genom att bygga applikationer på molnet möjliggör Microsoft Graph kommunikationen med tjänsterna som Microsoft erbjuder. Detta inkluderar mail- eller filhantering samt manipulering av företagsinformation. Ett företag kan bestämma hur de vill använda dessa tjänster som Microsoft Graph erbjuder baserat på intressen och prioriteringar.

2.5 Icewarp-mailserver

Icewarp-mailserver är en plattformslösning för hantering av mail-meddelanden. Icewarp används främst av organisationer som söker en tjänst för att hantera företagets kommunikation med kunder eller liknande. Icewarp har varit ett mycket populärt val för många företag eftersom plattformen är säker, skalbar och enkel att hantera. Icewarp har även möjlighet att lagra ett stort antal användare samt en stor mängd data [15].

2.6 Docker

Docker är en plattform som används för att distribuera programvara i isolerade miljöer, så kallade ”containers”. I en container finner man all relevant information om programvaran och alla kriterier som krävs för att köra programvaran. Fördelen med att skapa en applikation i en container är att den kan köras på alla möjliga plattformar, vilket innebär att samma programvara kan köras på olika miljöer. En annan fördel med Docker är att containertekniken gör det enklare att distribuera applikationer, oftast över molnet. Docker möjliggör för enkla distributioner till en molnleverantör eftersom utvecklare kan ta del av alla molntjänster med hjälp av en container utan att behöva göra ändringar i programkoden [16].

2.7 Autentisering

Autentisering är processen för att verifiera en användares identitet till en viss resurs. Oftast involverar detta ett användarnamn och lösenord som är unika för den givna användaren som försöker komma åt resursen. Det finns många olika former av autentisering inklusive lösenord, swipe-kort, foto-id, säkerhetskoder och smartkort. Metoden av autentisering beror helt på systemkraven som organisationen har för sina resurser och vilken policy de använder. Ibland kan även flera autentiseringsmetoder användas. Processen att autentisera en användare är nödvändig för att förhindra attacker och spridning av konfidentiell information för ett visst system [17].

2.7.1 Autentiseringsreferenser

För att autentisera en användare användas oftast lösenord eller liknande inmatningar, dessa kallas för autentiseringsreferenser. En autentiseringsreferens är helt enkelt en referens som användas för att autentisera användaren i ett system. Denna referens kan vara lösenord, säkerhetskoder eller specifika föremål som kan identifiera användaren genom unika koder[17].

2.7.2 OAuth 2.0

OAuth 2.0 är ett protokoll som möjliggör säker API-åtkomst genom att utföra autentisering under data-överföringsprocessen. Det finns olika flöden för att utföra denna

processen som OAuth 2.0 erbjuder. En fördel med att använda sig av OAuth2.0-autentisering är att en åtkomsttoken genereras till följd av processen. Denna token används som en nyckel av klienten för att få tillgång till den skyddade resursen. En annan fördel med OAuth2.0 är att protokollet separerar klienten från resurs-servern, vilket leder till att klienten inte behöver ge ut känslig information till en tredje-partsapplikation[18].

2.7.3 Autentiseringsflöden

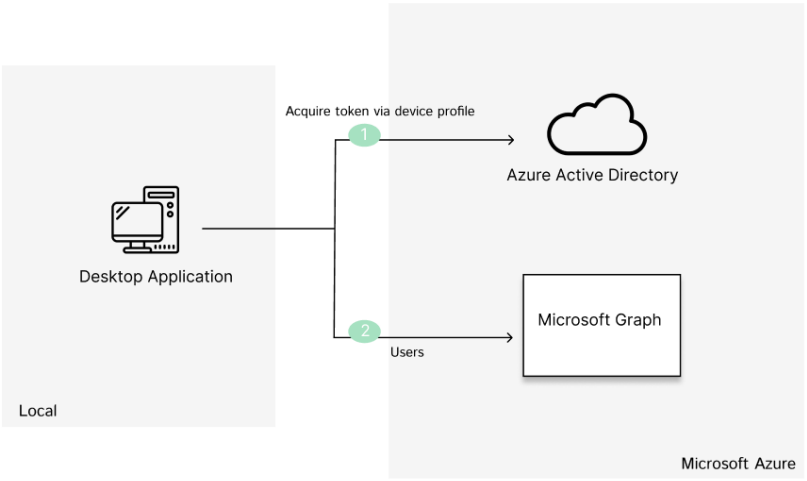
Autentiseringsflöde är en metod som används för att generera en token för att ge åtkomst till en resurs för en klient. Det finns många olika flöden av autentisering inklusive Implicit Flow, Hybrid Flow, Client Credential Flow, Device Authorization Flow och Resource Owner password flow. Beroende på kravet hos ett system kan ett specifikt flöde vara passande för att maximera säkerheten [19].

2.7.4 Microsoft Authentication Library

Microsoft Authentication Library (MSAL) möjliggör för utvecklare att få tillgång till tokens från Microsoft Identitetsplatform. Denna processen krävs för att autentisera användaren för åtkomst av säkrade webb-API:er. MSAL används bland annat för att få åtkomst till många kända Microsoft API:er som exempelvis Microsoft Graph [20].

Processen för att autentisera en användare med hjälp av MSAL går till genom att först erhålla en token från ett Azure Active Directory (AAD). När användaren fått en token från autentiseringsprocessen i AAD kan denna token användas för att få åtkomst till skyddade resurser på Microsofts plattform. För att använda MSAL i en applikation måste man först registrera applikationen i Azure portalen och konfigurera den med rätt behörigheter för ens egna ändamål. Därefter kan man på ett säkert och enkelt sätt använda MSAL i applikationen.

En fördel med att använda MSAL är att biblioteket själv hanterar förnyelse av tokens. Detta innebär att applikationen själv inte behöver förnya tokens utan detta hanteras automatiskt. Idag användas MSAL som ett verktyg i många olika plattformar som .NET, Java, Python och Android [20].



Figur 2.3: Figuren illustrerar hur en autentiseringsprocess i MSAL går till. Först får applikationen en token från AAD. Denna token används sedan med en specifik endpoint för att få åtkomst till en resurs, i detta fallet är det Microsoft Graph.

2.8 Azure Active Directory

Ett AAD är Microsofts egna lösning för hanterings- och identitetsjänster över molnet. AAD låter organisationer och användare få tillgång till många externa resurser som Microsoft erbjuder. I ett AAD ingår Microsofts API:er, Microsoft 365 och många andra SaaS-applikationer som finns tillgängliga. AAD ger möjlighet att hantera anställda i en organisation, som att exempelvis ge vissa användare fler rättigheter än andra beroende på deras roll i organisationen. Sammanfattningsvist är AAD en kritisk komponent i att få en organisation integrerad med Microsoft Azure arkitekturen [21].

2.9 Kanban-metodiken

Kanban-metodiken är ett arbetsätt som idag fått stor uppmärksamt av utvecklare runt omkring i världen. I kanban-metodiken behöver man först definiera vilken uppgift man vill åstadkomma. Därefter bestämmer man sig för vilken prioritet en viss uppgift har och rangordnar sina resterande uppgifter baserat på detta. Sedan genomför man uppgiften och validerar sina resultat innan man markerar de som klara. I en kanban-tabell har man oftast fyra listor att sortera efter. Den första listan är oftast en lista där man beskriver vilka uppgifter som skall genomföras, man rangordnar detta i prioritet då den uppgiften som dyker upp först i listan har högst prioritet. Den andra listan är oftast en “uppgiftslista” där det beskrivs vilka områden som är under utveckling. Sedan har man en “valideringslista” där tester utförs och man validerar resultatet på sina uppgifter. Sist i kanban-tabellen hittar man genomförda uppgifter där man listar alla avklarade moment i sitt projekt [22].

2.10 Verktyg

I denna sektion beskrivs de tekniska verktygen som användes under projektets gång.

2.10.1 Visual Studio

Visual Studio är en omfattande Integrated Development Enviroment (IDE) som utvecklats av Microsoft. Denna IDE erbjuder en mängd verktyg och funktioner för utvecklare att skapa en mängd olika applikationer, från desktop program till mobila applikationer och webbapplikationer. Den senaste versionen av programmet är Visual Studio 2022 och innehåller många integrerade verktyg för att effektivisera utvecklingsprocessen, såsom en fel-sökningsmotor och en kraftfull kodredigerare. Visual Studio finns tillgängligt i flera olika utgåvor, inklusive en kostnadsfri Community Edition och en mer avancerad Professional Edition[23].

2.10.2 Programspråket C#

C# är ett programmeringspråk som används för att utveckla olika program och applikationer. Språket är ett objektorienterat språk som innebär att programkoden består av en uppsättning av objekt som interagerar med varandra. Språket används främst inom utveckling för mobil-, desktop- och webbapplikationer. Språket skapades år 2000 av Anders Hejlsberg från Microsoft. Syftet med språket var att möta en hög efterfrågan på marknaden för nyare och mer moderna applikationer. C# är ett populärt programmeringsspråk som används i en rad olika tjänster och resurser [24].

2.10.3 .NET

.NET är ett ramverk som används för att skapa en mängd olika applikationer. Dessa applikationer kan vara anpassade för desktop, web eller mobil och kan köras på alla operativsystem. Kortfattat kan man säga att .NET är ett speciellt sätt att exekvera ett program för en viss miljö. .NET utvecklades av Microsoft, .NET skapades i syfte att förenkla och förbättra utveckling av applikationer. I dagsläget ger .NET stöd åt främst programspråket C# för utveckling. Fördelarna med att använda .NET är många, bland annat har .NET mycket resurser för utvecklare att ta del av på Internet. En annan fördel är att .NET förenklar programmeringen i IDE:s som Visual Studio då det finns mycket inbyggd funktionalitet som utvecklaren inte behöver tänka på under sitt arbete [25].

2.10.4 Microsoft SQL Server

Microsoft SQL server är en miljö där man kan administrera och hantera sina SQL-databaser. Tjänsten ger en mängd olika funktionaliter för att hantera databaser i en organisation. Microsoft SQL Server är en säker miljö där autentisering i form av användarnamn, lösenord och även att man är uppkopplad till rätt nätverk krävs för att få tillgång till en server [26].

2.10.5 Microsoft Teams

Microsoft Teams är en kommunikationsplattform som användes under projektets gång. I projektet genomfördes möten och kommunikation med företaget QBIS via Microsoft Teams.

2.10.6 Atlassian-produkter

Atlassian-produkter är en rad olika verktyg som används för utveckling, projekthantering och samarbete i ett team. I detta projekt användes verktygen Jira, Confluence och Bitbucket som alla är delar av Atlassian-produkter. Jira är en web-applikation som används för planering av projekt. Applikationen har utvecklats i syfte att förenkla arbetet inom programvaruutveckling. Idag är Jira en applikation som hjälper utvecklare att organisera ett arbete. Bland Atlassian-produkter finner man även applikationen Confluence som används för att dela dokumentation med andra utvecklare. På så sätt kan dokumentationen av arbetet delas på ett smidigt sätt med andra. Atlassian erbjuder ett versionshanteringsystem; detta systemet heter Bitbucket. Verktyget används för att lagra programvara inom ett visst projekt för enkel hantering samt uppdatering [27].

3
Metod

I detta avsnitt förklaras arbetets tillvägasätt före implementationen för examensarbetet. Avsnittet har som syfte att ge läsaren en bild om tankarna under förarbetet.

3.1 Översikt av arbetsprocessen

Examensarbetet började med en planeringsfas som tog upp en stor del av arbetstiden. Detta gjordes för att få en god förståelse för arbetet och även komma på tänkbara lösningar och implementationer. Därefter påbörjades implementationsfasen som tog upp majoriteten av tiden under arbetsprocessen. Projektet byggdes i .NET Core 7.0 med programspråket C#. I implementationsfasen skedde också testning parallellt med arbetet för att försäkra att programvaran fungerar korrekt. Slutligen gjordes ett slutarbete för att försäkra att alla kriterier har möts samt ytterligare testning och diskussioner med företaget.

3.2 Användning av Kanban-metodiken

I projektet användes Kanban-metodiken för att genomföra olika uppgifter under arbetets gång. Kanban är ett flexibelt och anpassningsbart agilt ramverk. Beslutet att använda kanban-metodiken gjordes tillsammans med företagets handledare. Under projektet genomfördes uppgifter i en 1-2 veckorsperiod med uppföljningsmöten med handledaren på företaget. Där utvärderades resultaten och det undersöktes om några ändringar eller svårigheter har uppstått. Under arbetsprocessen användes Jira som illustrerade projektets Kanban-tavla där olika uppgifter flyttades runt i olika kategorier beroende på deras stadie i utvecklingen. Projektet använde sig även av Confluence för dokumentation. Bitbucket användes som det primära versionshanteringsystemet under projektet tillsammans med Visual Studio IDE.

3.3 Planeringsfasen

Under planeringsfasen gjordes en förstudie om QBIS nuvarande lokala lösning. Först undersöktes de olika komponenterna i den nuvarande lösningen samt hur mail-flödet ser ut för den lokala integrationen. Detta gjordes för att få en bättre bild om hur den nuvarande implementationen fungerar mer konkret för att sedan börja tänka över den nya SaaS-lösningen.

En annan viktig del i denna fas var att göra en analys av olika SaaS-molnlösningar som kan användas för att migrera mailservern till molnet. En kritisk del i denna fas var att säkerställa att SaaS-lösningen möter alla krav som företaget har för inkommande mail. Detta inkluderar faktorer som mail-regler, säkerhet, kostnad, underhåll och skalbarhet.

Därefter utvecklades en plan för att migrera mailservern till molnet. Genom att först genomföra en planeringsfas på två veckor kunde arbetet ha en tydlig genomgång och bild av arbetet innan implementationen påbörjades.

3.4 Microsoft Azure

För att försäkra att lösningen erbjöd alla lämpliga krav för QBIS mail integration valdes Microsoft Azure som molnleverantör. Valet gjordes genom att säkerställa att Microsoft Azure uppfyller faktorer som funktionskrav, pris och underhåll.

3.4.1 Krav på molnleverantören

När det gäller företagets behov fanns det många viktiga krav att ta hänsyn till. Ett krav med den givna leverantören är att datahantering och lagring skall kunna erbjudas. Om exempelvis ett mail-ärende inte bearbetas korrekt skall detta ärende lagras så att kundens begäran inte går förlorad. Microsoft Azure har även en väldigt god tillgänglighet, vilket var kritiskt för att säkerställa att systemet fungerar korrekt under alla omständigheter. Sedan var ett krav från företaget att kunna applicera mail-regler på mail-meddelanden som kommer in i applikationen. På så sätt förhindras spam-meddelanden samt onödig hantering av mail-ärenden.

3.4.2 Pris

Pris var en mycket viktig faktor att ta hänsyn till vid valet av molnleverantör. Microsoft Azure erbjuder en rad olika prissättningsalternativ för olika budgetar och behov. Fördelen med Microsoft Azure är att den är mycket skalbar och priset kan variera beroende på storleken på applikationen. En annan nämnvärd fördel med Microsoft Azure är att det finns kampanjer riktade till kunder som förbinder sig under en längre tid under en viss licens. Detta var fördelaktigt för QBIS.

3.4.3 Underhåll

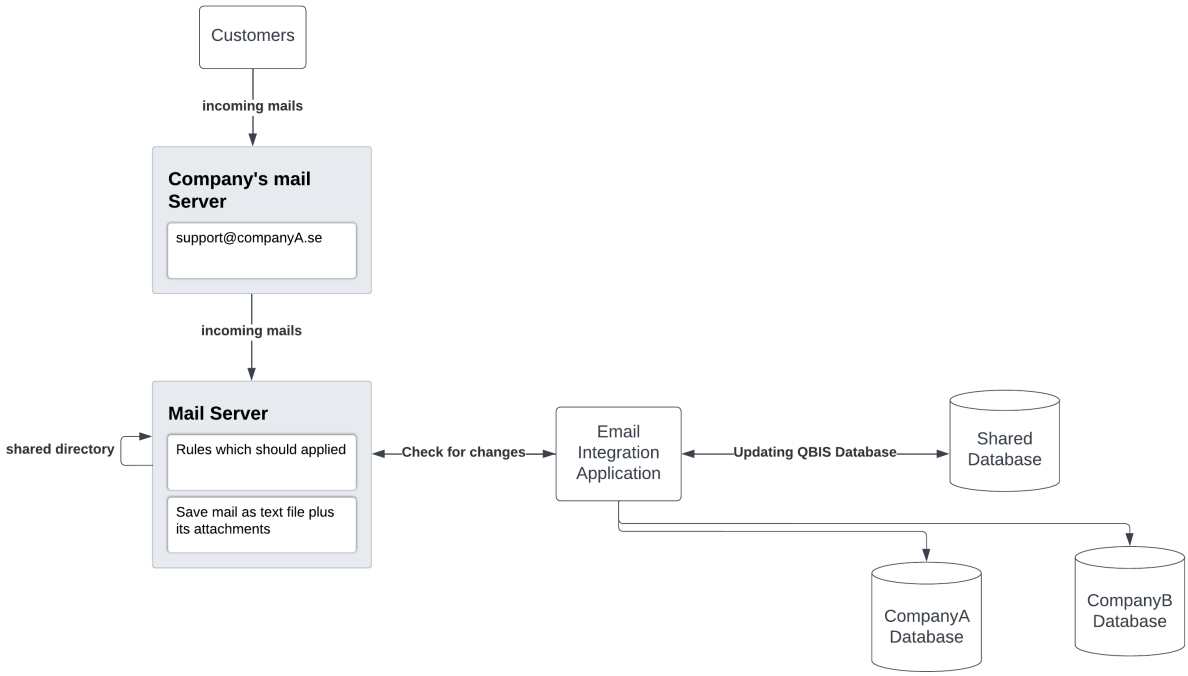
Underhållning av molntjänster kan ibland vara en utmaning. Microsoft Azure erbjuder en mängd olika verktyg för att underrätta hantering av molnresurser. Azure portal är en centraliserad plats för att övervaka och hantera alla molnresurser som en organisation använder sig av. På så sätt är det väldigt enkelt att göra ändringar eller optimeringar för de resurser som man använder sig av i molnet. Microsoft Azure ger också möjlighet att automatiskt skala och underhålla infrastrukturen; detta minskar behovet av att manuellt hantera sina tjänster.

4

Systemkonstruktion

I avsnittet systemkonstruktion beskrivs företagets nuvarande arkitektur på den lokala mail-lösningen. Därefter presenteras examensarbetets nya SaaS-lösning. Avsnittet kommer även illustrera detta i flödesdiagram med noggranna delavsnitt som förklarar alla kritiska delar.

4.1 QBIS nuvarande mailserver-integration



Figur 4.1: Figuren illustrerar ett flödesdiagram för den lokala mail-lösningen.

4.1.1 Flödet för ett mail

Figur 4.1 är ett flödesdiagram som illustrerar hur ett mail går från en kund in till QBIS miljön och hur ett service-desk-ärende skapas till följd av detta. I Figur 4.1 inleds processen med att en kund skickar ett mail till en utav QBIS kund-mailadresser. Därefter vidarebefordras detta mail till QBIS-domänen och in i Icewarp-mailservern för hantering. Genom polling-tekniken detekterar Email-integration-applikationen

nya mail-meddelanden i Icewarp-mailservern. I Icewarp valideras mail-meddelandet gentemot mail-regler och bearbetas som en textfil. Mail-meddelandet sparas sedan ner i en fil som är en delad resurs med Email-integration-applikationen. Applikationen informeras om ändringar när ett nytt mail har sparats och hanterar det genom att skicka vidare mailet till den generella databasen i QBIS miljön. Den generella databasen innehåller kopplingar till alla QBIS kunders individuella databaser. Mail-meddelandet jämförs nu med alla domäner och kunder i QBIS generella databas för att kartlägga vart detta mail-meddelandet ska sparas. Därefter uppdateras databasen med ett nytt mail-ärende som illustreras i Figur 4.1. Detta mail-ärende erhåller även ett unikt ärendenummer för identifiering.

4.1.2 Mailserver-lösningen

Den nuvarande mail-lösningen är baserad på en Icewarp-mailserver. I Figur 4.1 är denna helt enkelt benämd som Mail Server. Det är just denna mailserver som hanterar all logik för mailhanteringen hos QBIS. Bland annat applicerar denna regler på mail för att förhindra spam och se till att kriterier möts för att mail ska igenkännas. Icewarp-mailservern sparar även ner mail-meddelanden som kommer in som textfiler och sparar även anknytningar till mailet som bilder eller filer; dessa sparas i en mapp som finns lokaliserad på mailservern. I Figur 4.1 illustreras en direkt koppling mellan mailservern och Email-integration-applikationen. Denna är benämd som Shared Directory. Det är just detta directory som applikationen extraherar nya mail-meddelanden ifrån genom att använda polling tekniken. Därefter hanteras dessa mail-ärenden mot databaserna och service-desken.

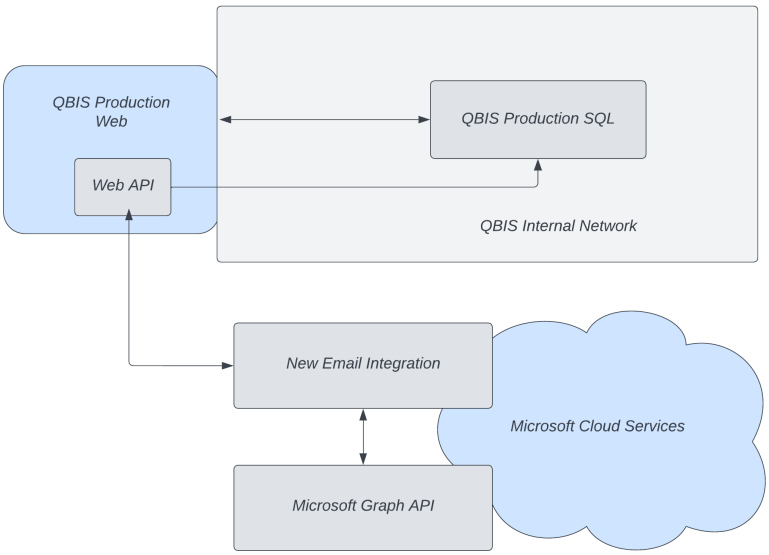
4.1.3 Email-integration-applikationen

När det kommer till Email-integration-applikationen är det många viktiga delar att ta upp. Applikationen som illustreras i Figur 4.1 innehåller all logik för att hantera mail som kommer in i Mail Server och samtidigt koppla dem till företagets databaser. Mellan Email-integration-applikationen och Mail Server i Figur 4.1 finns en koppling som kollar efter nya mail-meddelanden i Icewarp-mailservern. Dessa ändringar upptäcks via triggerfunktioner i Email-integration-applikationen. Därefter appliceras logiken för att hantera nya mail-ärenden i Email-integration-applikationen.

4.1.4 QBIS databas

Den sista viktiga delen i Figur 4.1 är uppdateringen av databasen i QBIS produktionsmiljö. Email-integration-applikationen skickar ett nytt mail och all information om det nya mail-meddelandet till en databas som innehåller information om alla kunder. I Figur 4.1 är denna databasen benämd som Shared Database och är huvuddatabasen i QBIS-miljön. Därefter finns det databaser för varje individuell kund. Syftet med denna separeringen är att koppla rätt mail-ärende till den korrekta kunden. När ett nytt mail har sparats i en databas för en kund så uppdateras QBIS service-desk. Beroende på ändringarna i databaserna skapas ett nytt mail-ärende på service-desken till följd av detta.

4.2 Integrationen med SaaS



Figur 4.2: Figuren illustrerar ett flödesdiagram för den molnbaserade SaaS-lösningen

4.2.1 Mailflödet i Microsoft Azure

I den nya mail-lösningen fungerar mailflödet i integration med Microsoft Azure. I Figur 4.2 illustreras hur Email-integration-applikationen nu befinner sig i Microsoft Azures miljö och integreras med Microsoft Graph API. Detta flöde i molnet grundas genom att köra applikationen som en Docker-container i molnet. Containern är synkroniserad med en mail-inbox och med hjälp av Microsoft Graph API kan applikationen utföra alla nödvändiga operationer för att hantera mail som kommer in. Containern startas automatiskt när ett nytt mail kommer in genom en webhook och avslutas när hantering av alla mail-ärenden genomförts. I Figur 4.2 illustreras även en koppling mellan Email-integration-applikationen och ett webAPI som befinner sig i QBIS produktionsmiljö. Eftersom QBIS databaserna endast kan nås via ett internt nätverk är det omöjligt för applikationen att spara mail-ärenden i QBIS databas via en direkt koppling. Därför möjliggörs detta genom att specificera endpoints i webAPI:et som bearbetar mailen till QBIS databaser. Efter detta har skett så uppdateras service-desken och ett nytt mail-ärende har nu skapats.

4.2.2 Mail-integrationen med Microsoft Graph API

Integrationen mellan applikationen och Microsoft Graph API skedde först genom att registrera applikationen i AAD. Därefter konfigurerades alla nödvändiga behörigheter för applikationen så att den kunde använda Microsoft Graph API, detta inkluderade bland annat förmågan att läsa och skicka mail. Därefter används OAuth 2.0-autentisering för att få åtkomst till Microsoft Graph API-tjänsterna, detta var bland annat att logga in med ett användarkonto med rätt behörigheter. När denna integreringen var gjord kunde applikationen vara synkroniserad med Microsoft Graph API för att hämta data från en mail-inbox.

När integrationen med Microsoft Graph API var genomförd kunde Email-integration-applikationen med hjälp av container teknik startas automatiskt när ett nytt mail upptäckts. Därefter hanteras detta genom specifika endpoints i Microsoft Graph API för hantering i applikationen och vidare sändningar till QBIS webAPI.

4.2.3 Integrationen med webAPI

I den molnbaserade lösningen befinner sig Email-integration-applikationen utanför QBIS-miljön i Microsoft Azure. Detta omöjliggör en direkt koppling mellan applikationen och QBIS databassystem. På så sätt måste applikationen integreras med QBIS webAPI genom en specifik endpoint. Email-integration-applikationen har som syfte att skicka information om ett mail-meddelande till en endpoint i webAPI:et. I webAPI:et sker då valideringen av mail-meddelandet mot databasen genom att identifiera vilken kund meddelandet kommer ifrån. Därefter genomför webAPI:et införandet av databaserna som skapar upp nya service-desk-ärenden för QBIS.

4.2.4 QBIS produktionsdatabas

Kopplingen till databaserna i QBIS-miljö är endast möjliga att nå via det interna nätverket för QBIS. Det är alltså omöjligt för Email-integration-applikationen att nå databaserna direkt från molnet. I Figur 4.2 illustreras det hur webAPI:et har en direkt koppling till produktionsdatabasen. webAPI:et befinner sig i det interna nätverket och kan därför nå databasen för att föra in informationen om ett visst mail. QBIS databaser är direkt synkroniserade med service-desken. Detta innebär att när ett nytt mail införs skapas ett nytt service-desk-ärende automatiskt.

5
Implementation

I detta kapitel beskrivs hur implementationen av arbetet gick till utifrån den förbestämda metoden.

5.1 Email-integration-applikationen

Den nya applikationen utvecklades först genom att analysera den tidigare lösningen samt söka efter möjliga förbättringar. Därefter skrevs den nya applikationen i programmeringsspråket C# och ramverket .NET 7.0. Ramverket .NET har även en direkt koppling till många av Microsofts molntjänster. Programvaran kunde på så sätt redan använda sig av mycket inbyggd funktionalitet som interagerar applikationen med Microsoft Azure. Detta förenklade arbetet och skapade ett enklare flöde i konstruktionen av applikationen.

Under utvecklingen av applikationen skedde arbetet via ett AAD för att få tillgång till Microsofts molntjänster. När ny funktionalitet implementerades kördes applikationen via autentisering med molnet för att se hur ändringarna påverkade applikationen i realtid. Under implementationsfasen skapades en Docker-container av applikationen för att distribuera den till Microsoft Azure. Detta gjordes för att utforska mycket av den containerlogiken som krävdes av företaget för att få applikationsflödet att fungera. Det inkluderade bland annat faktorer som automatiserade körningar av applikationen samt att minimera kostnader.

5.1.1 Val av autentiseringsflöde

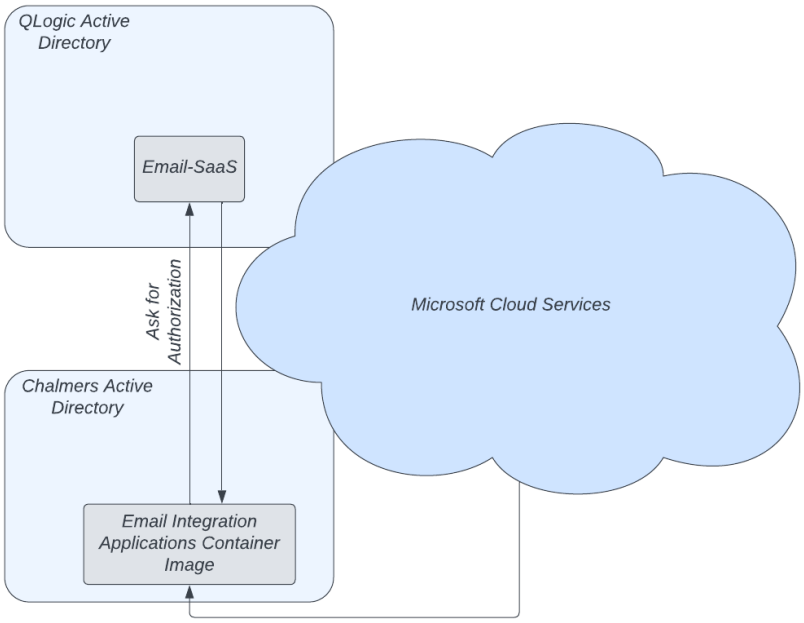
Under utvecklingen av applikationen valdes Client-Credential-flödet som slutgiltigt autentiseringsflöde. Detta val gjordes eftersom Client Credential tillåter applikationen att autentisera sig automatiskt utan användarinteraktion. Ett utav grundkriterierna för applikationen var att den skulle ha en automatiserad körning. Men eftersom applikationen måste autentisera sig mot molnet för användning av Microsoft Graph API krävdes en automatiserad autentiseringslösning. Client Credential uppfyllde dessa krav.

5.1.2 Testning

Testning var en kritisk del i implementationsfasen. Att säkerställa att programvaran hade ett önskat beteende var av högsta prioritet. Bland annat genomfördes integra-

tiontester för att säkerställa att olika delar av mjukvaran fungerade rätt tillsammans. Det var viktigt att testa samspelet mellan olika moduler i arbetet och då var denna typen av tester kritiska.

En annan nämnvärd faktor var att arbetet skedde långt utanför produktionsmiljön. Det AAD som arbetet skedde inom var ett Microsoft gratis användarkonto. Detta möjliggjorde testning av många molntjänster för att hitta en optimal lösning utan risk för att påverka företagets miljö. Däremot var huvud-applikationen i företagets miljö med anledning av autentisering. Under arbetet kördes all containerlogik för applikationen på ett separat AAD men autentiseringen skedde mot företagets AAD. Detta illustreras i Figur 5.1.



Figur 5.1: Figuren illustrerar autentiserings flödet mellan två AAD i Microsoft Azure. Här visas det hur containern med programkoden befinner sig i ett separat AAD och autentiserar sig mot en applikation i QLogic AAD.

5.1.2.1 Enhetstester

I implementationen av applikationen var en viktig del av testningen att genomföra enhetstester. Enhetstester säkerställer att delar av mjukvaran fungerar på rätt sätt. Exempelvis kan detta handla om enskilda delar i programvaran som funktioner och metoder [28]. I implementationsfasen utfördes enhetstester kontinuerligt med arbetet för att säkerställa på ett systematiskt sätt att mjukvaran fungerar som förväntat. Ett viktigt syfte med enhetstester är att detektera fel med programvaran innan det kan orsaka faktiska konsekvenser för företaget eller applikationen.

5.1.2.2 Integrationstestning

I implementationsfasen genomfördes integrationstester parallellt med enhetstester. Ett integrationstest är ett test som säkerställer att flera olika delar av programvaran interagerar på ett förväntat vis. Till skillnad från enhetstester syftar integrationstester till att säkerställa samspelet mellan olika funktioner och metoder [29]. Detta var en betydelsefull del i testningsarbetet eftersom rätt samspel mellan funktioner krävdes för att nå det önskade resultatet.

6
Resultat

I följande avsnitt presenteras och sammanfattas huvudresultaten från examensarbetet. Resultatet kommer även att kopplas till frågeställningarna och syftet med arbetet samt hur observationerna relaterar till detta. Avsnittet kommer att presentera resultat utifrån applikationens perspektiv samt de ekonomiska perspektiven.

6.1 SaaS-applikationen

Examensarbetet har uppnått en fungerande SaaS-applikation i Microsoft-Azure-miljön. I följande delavsnitt presenteras all funktionalitet som åstadkommits under arbetets gång. Applikationen körs som en container på Microsoft Azure och är avstängd tills ett nytt mail-meddelande triggar igång den.

Tabell 6.1: Tabellen illustrerar en egenskapsjämförelse mellan den lokala mail-lösningen och den molnbaserade mail-lösningen.

Lösning		Egenskaper
Ny	Gammal	
✓		Autentisering
✓	✓	Svarsmeddelande
✓		Uthämtning av det senaste mail-meddelandet
✓		Sändning till webAPI
✓	✓	Hantering av misslyckade mail-meddelanden
✓	✓	Lagring av felmeddelanden
✓		Stänga av container

6.1.1 Autentisering

Applikationen autentiserar i nuläget sig med Device Code Credential Flow. Det krävs en användare för att autentisera applikationen mot Microsoft Azure. Denna användare inmatar manuellt en kod och loggar in som en giltig användare för att få tillgång till applikationens funktionalitet.

6.1.2 Svarsmeddelande

SaaS-applikationen ger stöd för att ge ut svarsmeddelande till sändare av ett visst mail-ärende. Detta görs automatiskt av applikationen då en check görs varje 30 sekunder efter nya mail-meddelanden. När ett svarsmeddelande skickats ut för ett mail skickas inte ytterligare svarsmeddelanden, utan detta görs endast en gång per ärende.

6.1.3 Uthämtning av det senaste mail-meddelandet

Applikationen har funktionalitet att hämta ut det senaste mail-meddelandet som triggade igång containern på molnet. Detta används för närvarande som en referens för att applikationen ska veta om vilket mail som skall hanteras.

6.1.4 Sändning till webAPI

Funktionalitet för att sända ut information om ett mail till webAPI:et har implementerats. Detta görs genom att konvertera mail-meddelandet till en JSON payload. WebAPI:et sköter för nuvarande regler om validering av mail-ärenden samt att spara dessa till QBIS databaser.

6.1.5 Hantering av misslyckade mail-meddelanden

Applikationen har funktionalitet för att hantera misslyckade mail-meddelanden. Eftersom webAPI:et inte kan anses vara tillgänglig hela tiden har en mekanism implementerats för att spara undan dessa meddelanden för att skicka dem vid nästa körning istället. I nuläget sparas meddelandena i Microsoft Azure och hämtas sedan av applikationen vid nästa körning för att försöka skicka dem igen. Syftet med denna funktion är att undvika att förlora mail-ärenden på grund av tillfälliga problem med webAPI:et eller andra faktorer.

6.1.6 Lagring av felmeddelanden

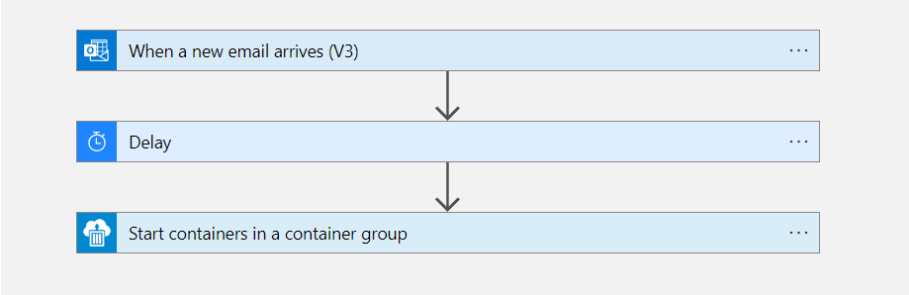
Lagring av felmeddelanden har implementerats i Microsoft Azure. Om ett fel sker under körning av applikationen är det möjligt att se information om detta felmeddelande i loggar. På så sätt kan utvecklare på QBIS snabbt felsöka ett problem med applikationen genom att undersöka loggarna i Microsoft Azure.

6.1.7 Stänga av container

Funktionalitet för att stänga av containern på Microsoft Azure har implementerats. På så sätt stängs applikationen av automatiskt när alla mail-ärenden under en given körning hanterats. Detta sker automatiskt och behöver ingen användarinteraktion för att genomföras.

6.2 Containerlogiken

Flödet för hur Docker-containern i molnet skall bete sig har implementerats i Microsoft Azure. Containern triggar igång applikationen när ett nytt mail-meddelande kommit in till en specifik mail-inbox. Därefter initieras en fördröjning på 5 sekunder för att säkerställa att applikationen har tillräckligt med tid att startas upp i Microsoft Azure. Sedan stängs containern inifrån applikationen och detta är alltså inte del av detta flödet utan det sker inuti körningen av applikationen.



Figur 6.1: Figuren illustrerar hur flödet med app-logiken ser ut när Azure Container Instance startas och körs igång.

6.3 Förbättringar jämfört med gamla lösningen

I följande delavsnitt presenteras alla observationer av förbättringar till följd av SaaS-applikationens utveckling.

6.3.1 Automatisering

Icewarp-mailservern körs idag kontinuerligt och använder polling-tekniken för att undersöka om nya mail-ärenden kommit in. I den nya molnbaserade lösningen körs applikationen endast när ett mail-meddelande kommit in genom webhooks. Detta innebär att applikationen automatiskt körs vid nya mail-ärenden samt stängs ner om det inte finns några nya ärenden.

6.3.2 Minskat underhåll

SaaS-applikationen underhålls och körs helt utanför QBIS interna system. På så sätt ansvarar inte personalen på QBIS för underhåll av applikationen. Detta leder till ett minskat ansvar och underhåll för applikationen, då ansvaret ligger på Microsoft Azure, eftersom applikationen befinner sig i molnets miljö.

6.3.3 Skalbarhet

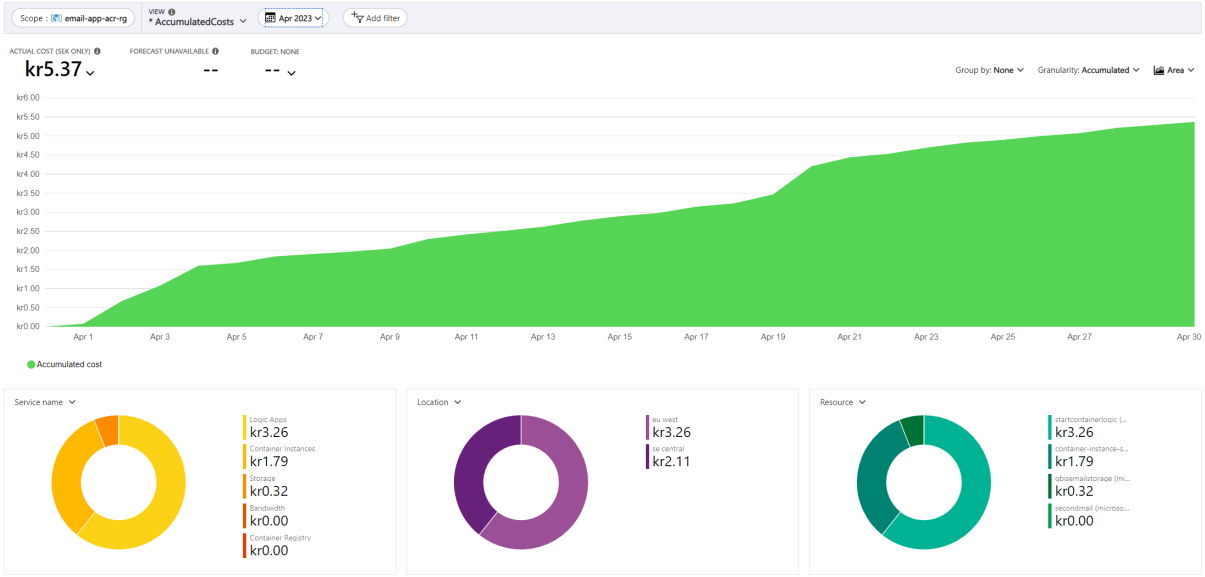
Microsoft Azure tillåter applikationen att skala upp sig. Med ökade påfrestningar eller mail-ärenden för företaget kan detta enkelt lösas genom att skapa nytt lagringsutrymme i molnet.

6.3.4 Flexibilitet

SaaS-lösningen har uppvisat en stor grad av flexibilitet. Molnet ger mer frihet till användaren över sin egen applikation och gör det enkelt att konfigurera applikationen. Man kan enkelt ta bort eller lägga till tjänster och anpassa inställningar för applikationen utan att behöva göra fysiska ändringar i servern. Då slipper användaren göra överinvesteringar på onödig hårdvara. Detta leder till snabba och smidiga beslut samt en väldigt låg omställningskostnad för företaget.

6.4 Ekonomiska faktorer

I dagsläget ligger Icewarp licensen på ca 500€ per år. Kostnaden för att hostas på den lokala mailservern adderar en ytterligare kostnad på ungefär 1500 sek/månad. Detta summeras till en månadkostnad på ungefär 2000 kr för QBIS nuvarande maillösning. QBIS får in ungefär 1500 requests per månad för sin nuvarande lokala mailserver. I Figur 6.2 illustreras hur kostnaden för 250 requests på containern ser ut. Detta summerades ihop av Microsoft Azure till en summa på 5.37 sek. Detta inkluderar kostnader för alla resurser relaterade till applikationen.



Figur 6.2: Figuren visualiserar månadskostnaden för mail-lösningen.

7
Diskussion

I följande avsnitt diskuteras resultatet och frågeställningarna för arbetet. Avsnittet kommer även att diskutera eventuella bidrag till fältet om molnbaserade lösningar samt hur arbetet kan inspirera för framtida molnlösningar. Slutligen presenteras svårigheter samt eventuella förbättringar och vidareutvecklings-punkter för projektet.

7.1 Utmaningar

Examensarbetet mötte en del utmaningar under dess gång. Dels var en stor del av dessa utmaningar under förarbetet för projektet. QBIS har inte haft någon erfarenhet av denna typ av projekt tidigare, det gällde under förarbetet att hitta en lösning från grunden. Utmaningar som uppkom till följd av detta var att utforska nya verktyg i Microsoft Azure samt att utveckla applikationen i ett nytt programmeringspråk. Under planeringsfasen uppstod betydande fördröjningar i tidsplanen till följd av detta, eftersom uppskattningarna för undersökningen överskreds. Därefter var det också kritiskt att säkerställa att applikationen kunde integreras med företagets system för att skapa ärenden för service-desken smidigt och korrekt. Eftersom SaaS-applikationen underhålls på Microsoft Azure kan den inte ha en direkt koppling till QBIS databaser. En utmaning kring detta var att introducera nya endpoints i QBIS webAPI för att skapa detta flöde mellan applikationen och databaserna.

En annan stor del av utmaningarna var att autentisera applikationen med Microsoft Graph API. Eftersom rättigheterna är väldigt begränsade i QBIS-miljö tog det väldigt lång tid att undersöka samt testa funktionalitet. Detta skapade fördröjningar i implementationen av applikationen då det krävdes konkreta dialoger med företaget för att gå framåt.

7.2 Fördelar och nackdelar

En mycket stor fördel med den molnbaserade SaaS-lösningen var att kostnaderna var mycket mindre. Eftersom applikationen endast körs under förfrågan leder detta till att det inte finns onödig kapacitet som betalas för när tjänsten inte används. När antalet förfrågningar till applikationen ökar, ökar också behovet av resurser. Microsoft Azure erbjuder mycket skalbara lösningar som gör det möjligt att succesivt öka eller minska resurser baserat på behov. Idag får QBIS in ungefär 50 mail-ärenden om dagen. Detta är ekivalent med 1500 ärenden per månad. Under en månad för detta examensarbete gjordes cirka 250 förfrågningar på applikationen i molnet. Om

applikationen hade tagit emot 1500 förfrågningar hade detta motsvarat en kostnad på 32 kronor jämfört med en tidigare kostnad på 2000kr. Detta visade sig rent procentuellt till en minskning med 98.4% av den ursprungliga kostnaden.

En nackdel med applikationen är att containern i molnet har en fördröjning på 20 sekunder innan den startar igång applikationen. Detta kan leda till besvär om en kund till QBIS ringer in live och mailar in ett ärende under samtalet. Denna fördröjningen kan leda till en försämrad kundupplevelse som kan påverka uppfattningen om företaget negativt.

7.3 Inspiration för framtida molnlösningar

Det fanns flera tillfällen under examensarbetets gång då utförandet av arbetet visade positiva lärdomar. I branschen finns det ett stort intresse för molnlösningar då ett flertal organisationer redan börjat migrera sin infrastruktur till molnet och där vill QBIS vara med i resan. Arbetets utfall samt idén om att flytta stora delar av företagets infrastruktur till molnet var av stort intresse för företaget. Examensarbetet visade att det redan finns ett intresse av att avlägsna så mycket som möjligt från företagets system, och resultatet indikerade att detta var en mer gynnsam strategi att anta. Resultatet av minskade kostnader, underhåll, flexibilitet och större skalbarhet var det som låg till grund för intresset.

7.4 Etik och hållbarhet

Examensarbetet påverkar inte några etiska aspekter. Däremot har projektet visat fördelar gällande hållbarhetsaspekten. Eftersom examensarbetet leder till minskat underhåll samt större skalbarhet är applikationen hållbar för företaget. Applikationen körs inte heller hela tiden utan aktiveras endast när nya mail-ärenden kommit in. Detta är positivt ur ett hållbarhetsperspektiv eftersom applikationen inte slösar onödigt med resurser när den inte har något användingssyfte.

7.5 Vidareutveckling

SaaS-applikationen har områden att utvecklas vidare på. Ett område att vidareutveckla är autentiseringen för applikation. I nuläget används Device Code Credential Flow som kräver att en användare manuellt autentiserar applikationen. Detta är inte tänkt för den slutgiltiga produkten. Applikationen är tänkt att autentiseras och köras självgående och för detta krävs Client Credential Flow.

Ett annat vidareutvecklingsområde är att optimera applikationen under molntjänsten att hantera flera mail-ärenden åt gången. I nuläget hanteras endast en förfrågan till applikationen medan resterande förfrågningar sätts på en väntelista. När det nuvarande ärendet har hanterats går applikationen över till nästa ärende i väntelistan. Detta kan vara problematiskt eftersom det kan komma in flera ärenden på

samma gång som kan skapa stora fördröjningar för kunderna. Om exempelvis ett akut ärenden hamnar längst ner i väntelistan kan detta skapa stora fördröjningar för kunden och företaget. Detta beror på hur många ärenden som kommer in under samma gång och kan skapa fördröjningar på flera minuter.

Att vidareutveckla loggnings-funktionaliteten för applikationen är även ett område att se över. I nuläget loggar applikationen till en textfil som inte har någon begränsning på storlek. Detta innebär att denna fil kan komma att bli väldigt stor i framtiden med många äldre loggnings-meddelanden som kanske är irrelevanta. Därför bör en begränsning på storleken av logg-filen implementeras.

8

Slutsats

Detta examensarbete demonstrerar effekterna av att migrera en lokal applikation till en SaaS-applikation under en molnleverantör. Genom att använda molntjänster som Microsoft Azure kan företag minska kostnader för infrastruktur och underhåll samtidigt som tillgängligheten och skalbarheten ökar för applikationen. Examensarbetet har även visat positiva effekter gällande att ta bort kraven på egen hårdvara som leder till ökad flexibilitet för personalen under företaget. Resultatet av projektet visar att migreringen av en lokal mailserver till molnet ledde till positiva följder inom områden kring automatisering, underhåll, skalbarhet, flexibilitet och ekonomi. Sammanfattningsvis kan det konstateras att molntjänster är ett kraftfullt verktyg för att optimera verksamheter och effektivisera deras applikationer.

Källförteckning

[1] "Om oss." [Online]. Available: https://www.qlogic.se/om-oss

[2] R. Chopra, *Cloud Computing:An Introduction.* Mercury Learning Information. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=cat07472a&AN=clec.EBC4895092&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds%7D,Year={2017},

[3] M. Saraswat and R. Tripathi, "Cloud computing: Comparison and analysis of cloud service providers-aws, microsoft and google," in *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, 2020, pp. 281–285.

[4] P. Mell and T. Grance, "The nist definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, 2011.

[5] Oracle, "What is a database?" [Online]. Available: https://www.oracle.com/database/what-is-database/

[6] E. Mohn, "Sql (structured query language)." *Salem Press Encyclopedia of Science*, 2023. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=ers&AN=87322820&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds

[7] S. Logic, "Crud - definition amp; overview." [Online]. Available: https://www.sumologic.com/glossary/crud/

[8] M. Biehl, *API Architecture*, ser. API-University Series. CreateSpace Independent Publishing Platform, 2015. [Online]. Available: https://books.google.se/books?id=6D64DwAAQBAJ

[9] CloudFlare, "What is http? | cloudflare." [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/

[10] A. Singjai and U. Zdun, "Conformance assessment of architectural design decisions on api endpoint designs derived from domain models," *Journal of Systems and Software*, vol. 193, p. 111433, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121222001352

[11] Hookdeck, "What are webhooks? how they work," https://hookdeck.com/webhooks/guides/what-are-webhooks-how-they-work, accessed 2023-03-19.

[12] A. S. Tanenbaum and H. Bos, *Modern Operating Systems.* Pearson, 2022.

[13] MSGraphDocsvTeam, "Use the microsoft graph api - microsoft graph." [Online]. Available: https://learn.microsoft.com/en-us/graph/use-the-api

[14] ——, "Microsoft graph overview - microsoft graph." [Online]. Available: https://learn.microsoft.com/en-us/graph/overview

[15] "Icewarp mail server," Jun 2022. [Online]. Available: https://www.liveagent.com/integrations/icewarp-mail-server/

[16] S. Grubor, *Deployment with Docker : A Practical Guide to Rapidly and Efficiently Mastering Docker Containers, along with Tips and Tricks Learned in the Field.* Packt Publishing, Limited. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=cat07472a&AN=clec.EBC5160905&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds%7D,Year={2017},

[17] D. Dasgupta, A. Roy, and A. Nag, *Advances in User Authentication. [electronic resource].*, ser. Infosys Science Foundation Series in Applied Sciences and Engineering. Springer International Publishing, 2017. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=cat07472a&AN=clec.SPRINGERLINK9783319588087&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds

[18] J. Chen, M. Hoppen, D. Böken, J. Reitz, M. Schluse, and J. Roßmann, "Identity, authentication and authorization in forestry 4.0 using oauth 2.0," in *2022 3rd International Informatics and Software Engineering Conference (IISEC)*, 2022, pp. 1–6.

[19] Auth0, "Authentication and authorization flows." [Online]. Available: https://auth0.com/docs/get-started/authentication-and-authorization-flow

[20] M. Documentation, "Learn about msal - microsoft entra." [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/develop/msal-overview

[21] M. Mayank, *Developing Applications with Azure Active Directory : Principles of Authentication and Authorization for Architects and Developers.*, 2019. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=edsbvb&AN=edsbvb.BV046190590&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds

[22] J. Saltz and R. Heckman, "Exploring which agile principles students

internalize when using a kanban process methodology." *Journal of Information Systems Education*, vol. 31, no. 1, pp. 51 – 60, 2020. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=bsu&AN=142056414&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds

[23] "Introduction to visual studio," Nov 2022. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-visual-studio/

[24] J. Ky, *C : A Beginner's Tutorial.* Brainy Software, 2013. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=cat07472a&AN=clec.EBC3003882&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds

[25] AWS, "What is .net?" [Online]. Available: https://aws.amazon.com/what-is/net/

[26] P. Atkinson and R. Vieira, *Beginning Microsoft SQL Server 2012 Programming.* John Wiley Sons, Incorporated, 2012. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=cat07472a&AN=clec.EBC818034&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds

[27] Nov 2022. [Online]. Available: https://www.spkaa.com/blog/what-is-atlassian

[28] M. Dziak, "Unit testing." *Salem Press Encyclopedia of Science*. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=ers&AN=137502047&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds,Year={2019},

[29] Z. Xu, W. Zhang, M. Li, X. Han, and L. Tang, "Research on component incremental integration testing technology," in *2022 IEEE 2nd International Conference on Data Science and Computer Application (ICDSCA)*, 2022, pp. 279–283.

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborgs Universitet
Göteborg, Sverige
www.chalmers.se

CHALMERS