CHALMERS

GÖTEBORGS UNIVERSITET

Molntjänst för hantering av digitala tillträdeskort

Examensarbete inom högskoleingenjörsprogrammet Datateknik

Pierre Oskarsson

Robert Nilsson

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK

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

Slutrapport 2023

Molntjänst för hantering av digitala tillträdeskort

Pierre Oskarsson

Robert Nilsson

GÖTEBORGS
UNIVERSITET

CHALMERS

Institutionen för Data- och informationsteknik

Chalmers Tekniska Högskola

Göteborg, Sverige 2023

Molntjänst för hantering av digitala tillträdeskort

Pierre Oskarsson

Robert Nilsson

Teknisk handledare: Peter Fredriksson & Erik Hellström, Scionova AB

Handledare: Pelle Evensen, Institutionen för Data- och informationsteknik

Examinator: Lars Svensson, Institutionen för Data- och informationsteknik

Slutrapport 2023

Institutionen för Data- och informationsteknik

Chalmers Tekniska Högskola

SE-412 96 Göteborg

Sverige

Telefon +46 31 772 1000

Molntjänst för hantering av digitala tillträdeskort

Pierre Oskarsson

Robert Nilsson

Institutionen för Data- och informationsteknik

Chalmers Tekniska Högskola

Abstract

This report presents the development of a cloud-based solution for providing digital access cards via mobile phones, aimed at extending the lifespan of existing access systems and reducing electronic waste. The solution leverages various AWS services, including IoT Greengrass, DynamoDB, API Gateway, and Lambda, to enable seamless communication between the access system and the mobile application. The implementation resulted in a fully automated and transparent service that requires minimal manual intervention from administrators. The project achieved its goal of developing a cost-effective solution that supports the company's business model and is expected to meet the needs of its customers. The solution represents a successful technical project that addresses the challenges of transitioning to a wallet-less society.

The report is written in Swedish.

Keywords: Cloud, digital access card, mobile phone, access system, electronic waste, AWS

Terminologi och förkortningar

API	Application Programming Interface – Ett gränssnitt för kommunikation mellan program, system och applikationer.
Connectivity	Förmågan hos enheter, system eller nätverk att kommunicera och dela information med varandra.
HTTPS	Hypertext Transfer Protocol Secure – Utökning av http-protokollet med kryptering. Vanligt använt för kommunikation på internet.
IoT	Internet of Things – Samlingsbegrepp för olika produkter som består av inbyggd elektronik och uppkoppling mot internet.
MFA	Multi-Factor Authentication – Vid inloggning krävs mer än en identitetsverifiering, exempelvis lösenord och fingeravtryck.
MQTT	Lättviktigt kommunikationsprotokoll främst avsett för IoT. Bygger på publicera/prenumerera meddelandetransport.
MVP	Minimum Viable Product – Definierar en produkt som är den absolut minsta fungerande version som ger ett värde till användaren.
REST	Representational State Transfer – Designprinciper för webtjänster
TLS	Transport Layer Security – Kryptografiskt kommunikationsprotokoll. Ofta använt mellan webbapplikationer och servrar för säkert utbyte av information.
Use Case	Metod för att systematiskt identifiera, klargöra och organisera systemkrav. Innehåller vanligtvis; Aktör, Mål och Systemet.
User Stories	Kort beskrivning i vardagligt språk av vad en användare vill uppnå. Används för att på ett snabbt sätt formulera kravspecifikationer.

Innehållsförteckning

1 Inledning

1.1 Bakgrund

Det så kallade ”plånbokslösa” samhället innebär att fler och fler fysiska saker blir digitala och flyttar in i våra mobiltelefoner. Detta har redan skett med till exempel kreditkort, ID i form av BankID, medlemskort, biljetter o.s.v.

Ett tillträdessystem hanterar tillträde till olika typer av resurser (till exempel dörrar, skrivare, betalstationer), vanligtvis med ett kort eller nyckelbricka/tagg som läses av mot en kortläsare. Data på passerkortet/taggen/tillträdeskortet verifieras i tillträdessystemet, vilket sedan godkänner eller nekar tillträde.

Det är naturligt att vi förväntar oss att även tillträdeskortet till arbetsplatsen eller gymmet finns i vår mobiltelefon. Det finns befintliga lösningar på marknaden men de har haft svårt att slå brett, sannolikt p.g.a. investeringskostnader, försvårat handhavande och ökad administration, antaganden som styrks genom en marknadsundersökning som uppdragsgivaren genomfört. Än så länge är det traditionella plastkortet vi använder bättre ur alla dessa aspekter och det blir därför svårt för företag och organisationer att rättfärdiga byte till digitala tillträdeskort.

Nordic FrameWorks AB (”Företaget”) har skapat en lokal lösning (”Proximate”) där en mobilapplikation (”Proximate App”) kommunicerar med en egenutvecklad hårdvara (”modulen”) som placeras vid kortläsaren. Mobilapplikationen innehåller data för ett fysiskt kort som lades in manuellt, både i mobilapplikationen och tillträdessystemet. När en användare befinner sig i närheten av kortläsaren/modulen med sin mobiltelefon skickas kortdata till kortläsaren via ”modulen”. Företaget har anlitat Scionova AB för att utveckla lösningen och de fungerade som stöd i projektet. Scionova är ett teknikkonsultbolag med inriktning på Connectivity och IoT.

1.2 Syfte

Företaget ville utvärdera om det vara möjligt att sammankoppla den lokala lösningen med ett befintligt tillträdessystem på ett kostnadseffektivt sätt för uppsättning och löpande drift, och därigenom undvika den manuella hantering som nämns i stycket ovan. Det skulle göra lösningen attraktiv för kunder då det kan minimera tid för administration samtidigt som det förenklar handhavandet för medarbetarna och medverkar till organisationens hållbarhetsarbete.

Genom att möjliggöra digitala tillträdeskort i kundernas befintliga tillträdessystem var Företagets hypotes att det kan förlänga livslängden på dessa system med 40% (10 år) eller mer, vilket bidrar till minskat elektronikskrot. Denna typ av avfall ökar varje år, enbart en tredjedel återvinns [1] vilket medför att det borde minimeras så långt som möjligt. Då även kort och taggar tas bort, minskar även nyttjandet av plast som hade behövts vid tillverkning av dessa.

Därmed finns en hållbar, funktionell och ekonomisk fördel gentemot befintliga lösningar på marknaden där hela tillträdessystem behöver bytas ut för att uppnå liknande funktionalitet. Företaget bedömer att det då finns goda förutsättningar för att fler organisationer kan börja använda digitala tillträdeskort och därmed öka digitaliseringen i samhället.

1.3 Mål

Målet var att utveckla en serverlösning (”Proximate Cloud”) som hanterar applikationslogik, kommunikation mellan tillträdessystem och mobilapplikation samt lagrar nödvändiga data. Det ska resultera i en produkt som gör flödet för registrering till tjänsten automatiskt, utan inverkan från systemadministratör. Användare av mobilapplikationen kan, via den utvecklade produkten, bli tilldelade digitala tillträdeskort till de underliggande system man har behörighet till.

Aktuell status (aktiv, blockerad, o.s.v.) för användarens digitala nycklar ska presenteras för användaren i applikationen. Data hämtas från underliggande system och uppdateras vid förändrad status. All administration av behörighet och giltighet för kort hanteras fortsatt i underliggande system.

Serverlösningen ska utformas för att sömlöst och transparent, utan behov av manuell hantering av administratör, erbjuda användarna digitala tillträdeskort, samt stödja Företagets affärsmodell på ett konkurrenskraftigt sätt.

1.4 Avgränsningar

Projektet omfattar inte utveckling av applikation i mobiltelefon, denna tillhandahålls av Företaget. Även modulen (hårdvara) som kommunicerar med mobilapplikationen och simulerar tillträdeskortet tillhandahålls av Företaget.

Det omfattar inte heller API i underliggande tillträdessystem, detta finns tillgängligt i testsystemet. Projektet fokuserar på att enbart utveckla integration mot underliggande system av typen Axis A1001, vilket är ett tillträdessystem med inbyggd webbserver.

Lösningen bygger på standardiserad kommunikation, så som HTTP, MQTT och REST. Ingen egen utveckling inom detta område krävs.

I samband med att projektet definierades tillsammans med uppdragsgivare och tekniska handledare, bestämdes att en molntjänst är det bästa sättet att stödja affärsmodellen.

2 Metod

Projektet planerades och utfördes enligt agila principer med SCRUM-metodiken, där kontinuerlig iterativ utveckling sker i separata sprintar, där varje sprint ämnar leverera värde [2] [3]. Som arbetsverktyg användes Jira, ett planerings- och ärendehanteringssystem med väl utvecklat stöd för SCRUM.

Enligt denna metod delades målet upp i flera MVP:er [4]. Det gav ett experimentorienterat arbetssätt där varje version av produkten ska leverera värde till användarna av tjänsten och ge kunskap inför nästa etapp. Varje MVP innehåller en eller flera User Stories som ska uppfyllas och innefattar en eller flera av följande komponenter:

- Användardatabas
- Autentiseringstjänst till Proximate Cloud
- REST-API för kommunikation med mobilapplikation
- integration med underliggande tillträdessystem

Mjukvara som utvecklas sparas på en webbaserad lagringsplats för versionshantering (Git) [5].

För att avgöra hur väl molntjänsten stödde Företagets affärsmodell på ett konkurrenskraftigt sätt, togs fyra scenarion fram för att förse ett befintligt tillträdessystem med digitala tillträdeskort.

3 Teknisk bakgrund

3.1 Serverless computing

Tanken bakom serverless computing är att outsourca hela ansvaret för serverhantering till en tredjepartsleverantör, vilket gör att användaren slipper hantera och underhålla egna servrar. Detta inkluderar allt från hantering av infrastruktur, till skalning och underhåll av applikationer och tjänster.

I AWS finns tjänsten Lambda [6], vilken är en implementation av serverless computing. Det är en tjänst som kan aktivera funktioner/kodblock vid olika valda händelser. Exempelvis kan en Lambda aktiveras och starta en databassökning, baserat på händelsen att en godkänd förfrågan inkommit via tjänsten API Gateway. Lambdas är, i tid, kortvariga datorkörningar som enbart kräver serverresurser under den tid de exekverar sin kod.

En stor fördel med Serverless computing är att det reducerar tid och resurser som behövs för att hantera infrastruktur och underhåll. Dessutom betalar man bara för de resurser man faktiskt använder, vilket potentiellt kan minska kostnaderna jämfört med traditionell molninfrastruktur. Serverless computing möjliggör också för utvecklare att fokusera på att skriva kod och förbättra applikationer, i stället för att spendera tid på att hantera och underhålla servrar.

En annan fördel är att det möjliggör nya typer av applikationer och tjänster som tidigare inte var möjliga med traditionella servrar. Eftersom Serverless computing automatiserar skalning och resursallokering, kan applikationer skalas automatiskt för att möta efterfrågan från miljontals användare. Detta är särskilt användbart för applikationer med oförutsägbara eller varierande belastningar.

I Appendix A (kap. 8) diskuteras hållbara och etiska aspekter i serverless computing-applikationer.

3.2 Zero trust

”Zero trust" är en säkerhetsmodell för datasäkerhet som innebär att all åtkomst till resurser inom ett IT-system ska autentiseras, auktoriseras och verifieras på ett säkert sätt, även om åtkomsten sker inom organisationens interna nätverk [7]. Det innebär att ingen trafik eller användare ska ha full åtkomst till systemet, utan att

varje användare och enhet måste autentiseras och verifieras varje gång de försöker ansluta till systemet. Genom att implementera Zero trust-modellen kan organisationer minska risken för dataintrång och öka säkerheten i sina nätverk och system.

3.3 Cognito

Amazon Cognito är en tjänst vilken tillhandahåller autentisering, auktorisering och användarhantering för webb- och mobilapplikationer. Tjänsten består av två huvudkomponenter: användarpooler och identitetspooler. Användarpooler fungerar som användarkataloger vilket ger registrerings- och inloggningsalternativ för applikations-användare. Användare kan logga in direkt med ett användarnamn och lösenord eller via en tredje part såsom Facebook, Google eller Apple. Genom att använda denna färdiga och beprövade autentiseringslösning för hantering av inloggning i applikationen, behöver ingen extra utveckling göras för att säkerställa kontrollerad åtkomst till ytterligare AWS-tjänster. När en användare väl är autentiserad kan denna kopplas till en identitetspool, vilken i sin tur ger möjlighet att tilldela användaren tillfälliga behörigheter till andra AWS-tjänster som krävs för funktionalitet av applikationen.

Användarpooler erbjuder en rad funktioner, inklusive registrerings- och inloggningstjänster, ett anpassningsbart webbgränssnitt för användarinloggning samt, vilket nämndes i föregående stycke, social inloggning med olika identitetsleverantörer. De erbjuder också säkerhetsfunktioner såsom flerfaktorautentisering (MFA), kontroller för kompromitterade referenser och skydd mot kontokapning.

3.4 API Gateway

Amazon API Gateway är tjänst som tillåter användare skapa, underhålla och säkra API:er [8]. Dessa används för att hantera all trafik från/till mobilapplikationen och övriga AWS molntjänster. Den agerar som en dörr, där godkänd autentisering krävs för åtkomst till bakomliggande tjänster. Den hanterar automatiskt skalning vid olika belastningar, samt garanterar hög tillgänglighet genom flera samtidiga instanser. Via API:n aktiveras olika Lambas, vilka i sin tur interagerar med ytterligare AWS tjänster.

3.5 DynamoDB

Amazon DynamoDB är en fullständigt hanterad NoSQL-databastjänst. Det vill säga att Amazon automatiskt hanterar backup och sömlös skalbarhet och ger förutsägbar prestanda. Tjänsten erbjuder också kryptering för att skydda känsliga data.

NoSQL (ursprungligen "non-SQL" eller "non-relational") är en databasdesign som möjliggör lagring och hämtning av data på andra sätt än de tabellrelationer som används i relationsdatabaser. NoSQL finns i en mängd olika typer baserade på deras datamodell, inklusive dokument-, nyckel-värde-, bredkolumn- och grafmodeller. Amazon DynamoDB är en NoSQL-databas organiserad som nyckel-värde-par. Det innebär att data lagras som nyckel/värde-par, där varje unik nyckel fungerar som en identifierare för att hämta det associerade värdet.

3.6 IoT Core

AWS IoT Core är en integrerad plattform för att ansluta och hantera IoT-enheter. Den möjliggör säker kommunikation mellan enheter och molnet samt insamling och analys av data, exempelvis IoT Greengrass som förklaras vidare i kap. 3.7. Plattformen är skalbar och stöder olika kommunikationsprotokoll, till exempel HTTP och MQTT.

3.7 IoT Greengrass

AWS IoT Greengrass är en tjänst som utökar AWS molnkapacitet till lokala enheter och lösningar. Det erbjuder bland annat möjligheten att utföra beräkningar och köra Lambdas lokalt på en IoT enhet, för att sedan kommunicera bearbetade data till molnet. Enheten fungerar även om internetanslutning tillfälligt saknas. Vid återkoppling mot internet synkroniseras eventuella händelser. Kommunikation med AWS sker genom säkra anslutningar och certifikat. Ytterligare beskrivning av hur tjänsten användes i projektet presenteras i avsnitt 4.4 Implementation.

4 Genomförande

Enligt valda arbetsmetoder delades projektet in i en design-fas där övergripande tekniska detaljer beslutades, en värdering-fas där molnleverantörer jämfördes, en planeringsfas för att skapa tydliga delmål och slutligen själva implementationen.

4.1 Övergripande arkitektur

För att förstå vilka tekniska komponenter som behövs i systemet genomfördes en brainstorming-session för att hitta och beskriva User Stories, vilka utgör underlag för en första MVP:

a1. *Skapa digitalt kort* – Som kortanvändare vill jag få ett digitalt kort när jag loggat in med mobilapplikationen.
a2. *Aktivera digitalt kort* – Som kortanvändare vill jag kunna passera en dörr med mitt digitala kort.
a3. *Status för digitala kort* – Som kortanvändare vill jag kunna se aktuell status på mitt digitala kort i mobilapplikationen för att veta att den är giltigt.

User Stories omformades till Use Cases med syfte att få en mer detaljerad insikt i nödvändigt flöde av information.

Use Case:

a1. Registrera ny användare utan koppling till underliggande system:
    1) Kortanvändaren (”CU”) installerar Proximate App
    2) CU startar applikationen för första gången
    3) CU registrerar sig till tjänsten med epost och lösenord
    4) CU loggar in i applikationen
    5) CU skriver in för- och efternamn
    6) CU skriver in organisation och anläggning
    7) CU skriver in PIN-kod som används i underliggande tillträdessystem
    8) Uppgifterna verifieras mot databas i Proximate Cloud
    9) Digitalt tillträdeskort genereras och sparas på CU i Proximate Cloud databas
    10) Digitalt tillträdeskort laddas ner till Proximate App
    11) Digitalt tillträdeskort och information om CU visas i Proximate App

a2. Registrera ny användare med koppling till underliggande system (utökat a1):

1) Användardata från underliggande system sparas i Proximate Cloud databas
2) Kortanvändaren (”CU”) installerar Proximate App
3) CU startar applikationen för första gången
4) CU registrerar sig till tjänsten med epost och lösenord
5) CU loggar in i applikationen
6) CU skriver in för- och efternamn
7) CU skriver in organisation och anläggning
8) CU skriver in PIN-kod som används i underliggande tillträdessystem
9) Uppgifterna verifieras mot databas i Proximate Cloud
10) Digitalt tillträdeskort genereras och skickas till underliggande system
11) Databas i Proximate Cloud uppdateras
12) Digitalt tillträdeskort laddas ner till Proximate App
13) Digitalt tillträdeskort och information om CU visas i Proximate App

a3. Visa kortstatus:

1) Proximate Cloud hämtar och sparar information om förändringar av användare och tillträdeskort i underliggande tillträdessystem med jämna intervaller
2) Proximate App hämtar information om tillträdeskort från Proximate Cloud och visar status på digitalt tillträdeskort

Med hjälp av ovan Use Cases skapades ett diagram enligt *Fig. 1,* där systemets olika huvuddelar och informationsflödet mellan dem har visualiserats.

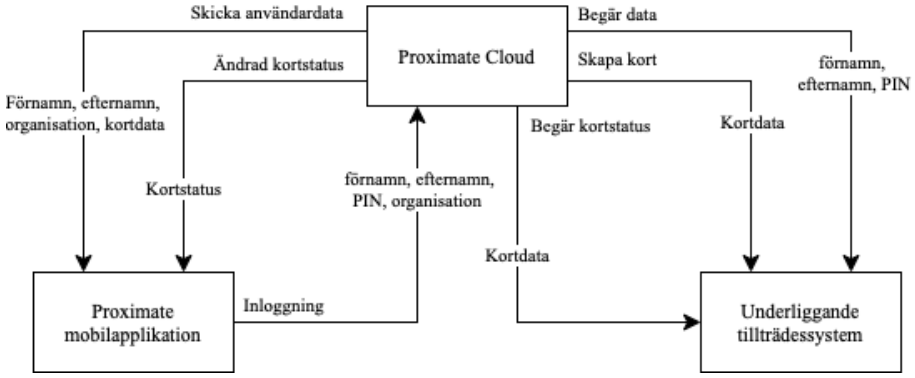
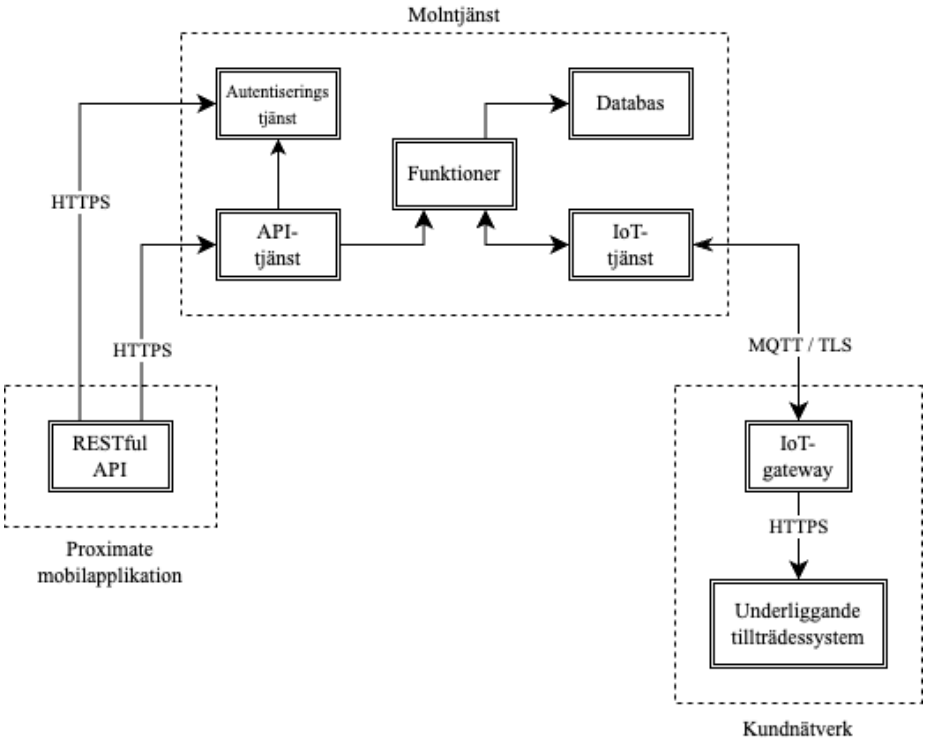


Fig. 1: Diagram för flöde av information.

Diagrammet i *Fig. 1* låg till grund för en workshop tillsammans med tekniska handledare, där preliminära tjänster och kommunikationsprotokoll bestämdes. *Fig.2* visar den tekniska arkitektur som bestämdes.

Informationen användes för att genomföra utvärderingen av molnleverantörer, se av-



snitt 4.2.

Fig. 2: Teknisk arkitektur för "serverless" molntjänst. Riktning på pil indikerar från vilken tjänst kommunikation kan initieras.

4.2 Utvärdering

Många på marknaden existerande molnleverantörer erbjuder liknade utbud av tjänster. De två största leverantörerna, sett till marknadsandelar, är AWS (Amazon Web Services) och Microsoft Azure [9]. De innehar tillsammans över 50% av marknaden, där AWS står för 34% och Azure för 21%. Eftersom det fanns goda möjligheter till support från tekniska handledare och tjänsterna var mycket väldokumenterade, valdes dessa två leverantörer med avseende på prisbild. Fyra olika hypotetiska kundbaser låg till grund för beräkningarna:

- 5 kunder, med 1 000 kortanvändare, totalt 5 000
- 50 kunder, med 1 000 kortanvändare, totalt 50 000
- 100 kunder, med 1 000 kortanvändare, totalt 100 000
- 1000 kunder, med 1 000 kortanvändare, totalt 1 000 000

Varje kundbas innehåller: autentisering för aktuellt antal kortanvändare, API-tjänst där kortstatus anropas varje timme, IoT-service där anrop görs för 5% av kortanvändarna varje dag, IoT-gateway där varje kund har en enhet, databas (läsningar från API-tjänsten och skrivningar från IoT-tjänsten) samt ”Funktioner” som agerar på tjänsternas anrop. För den största kundbasen innebär det att ca 720 000 000 Funktioner exekveras varje månad. *Tabell 1* redovisar respektive leverantörs namn på ingående tjänster samt procentuella prisskillnaden dem emellan. Ett positivt procenttal indikerar att Azure var dyrare än AWS, medan ett negativt procenttal indikerar att Azure var billigare än AWS. Fält som visar ±100% innebär att tjänsten är gratis för någon av leverantörerna vilket medför att en jämförelse inte är möjlig.

Tabell 1: Prisskillnad i procent per tjänst på Azure jämfört med AWS per månad.

Tjänst	Moln-leverantör		Antal kortanvändare			
	AWS	Azure	5 000	50 000	100 000	1 000 000
Autentiserings-tjänst	Cognito	Active Directory External Identities	0%	0%	-41%	-30%
API-tjänst	API Gateway	API Management	2%	38%	40%	53%
Funktioner	Lambda	Functions	13%	-4%	-3%	-4%
Databas	DynamoDB	Cosmo DB	37%	-80%	-80%	-11%
IoT-tjänst	IoT Core	IoT Hub	-100%	-100%	-100%	400%
IoT-gateway	IoT Green-grass	IoT Edge	-100%	-100%	-100%	-100%
Totalt			4%	29%	-6%	-4%

Fig. 3 visar en sammanfattning av den procentuella skillnaden i månadskostnad för Azure jämfört med AWS för varje kundbas med genomsnittspris som utgångspunkt. Prisbilden är till AWS fördel i mindre kundbaser, där AWS är klart billigare vid 50 000 kortanvändare. Skillnaderna jämnas ut och de båda leverantörerna närmar sig varandra när antal användare ökar.

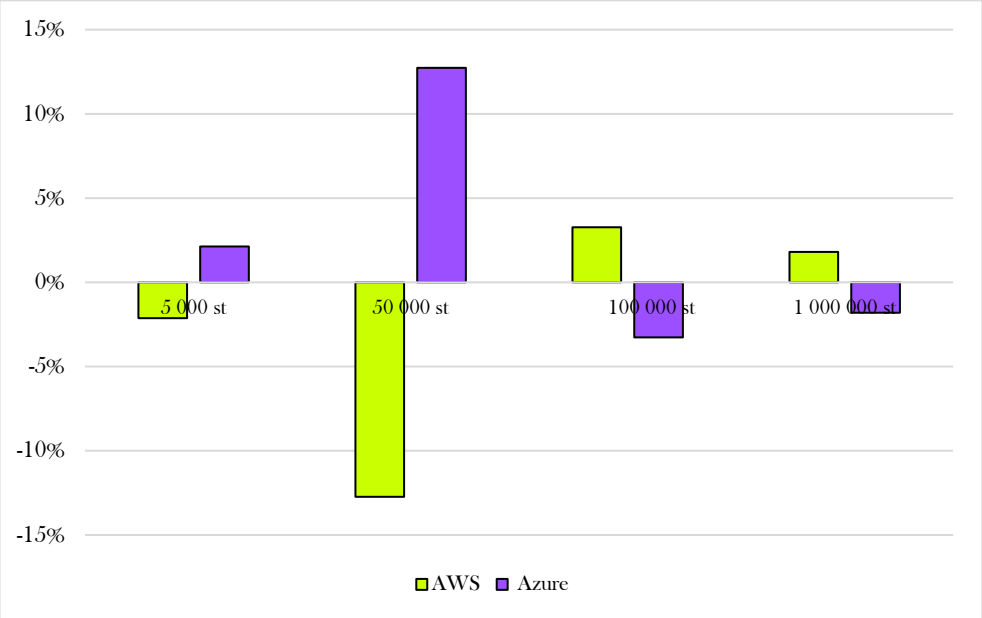


Fig. 3: Prisskillnad i procent på Azure jämfört med AWS per månad för varje kundbas.

Då en av målsättningarna var att identifiera en så kostnadseffektiv lösning som möjligt, speciellt med en till början begränsad kundbas, valdes AWS som plattform i implementationsfasen av projektet.

4.3 Systemdesign

I det här steget omvandlades den generella arkitekturen beskriven i *Fig. 2* för att passa tjänsterna i AWS. Resultatet visas i *Fig. 4*.

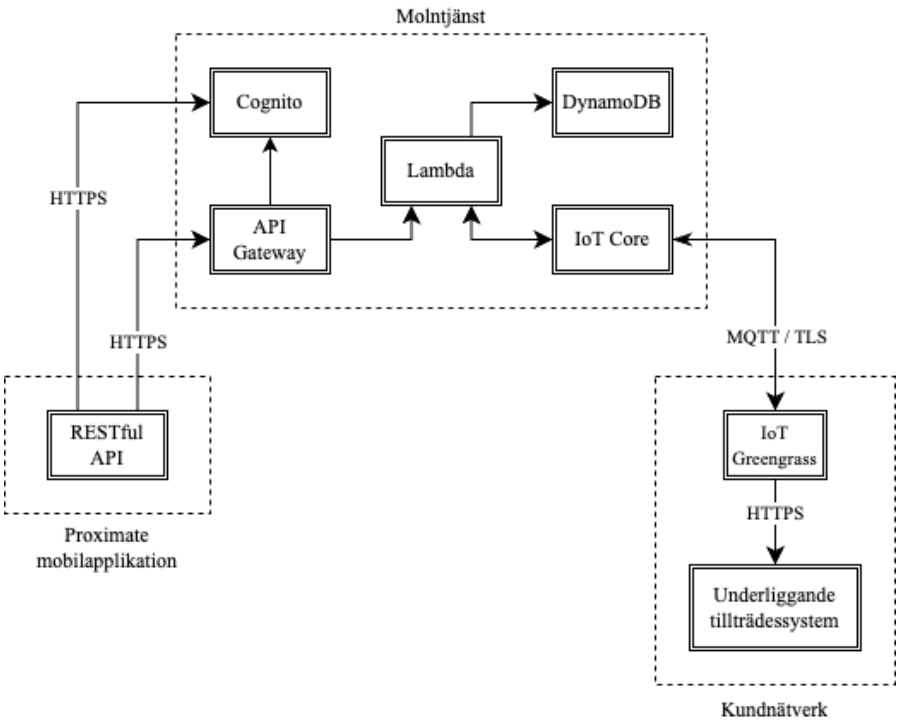


Fig. 4: AWS tjänster och hur de interagerar med varandra. Riktning på pil indikerar hur kommunikation kan initieras.

4.4 Implementation

För att genomlysa dataflödet och upptäcka eventuella problem i definierade Use Case, skapades flera sekvensdiagram där samtliga meddelande mellan de ingående tjänsterna och i vilken tidsordning de skulle ske dokumenterades.

Detta gav en tydlig bild att arbeta utefter med ett väldefinierat mål om funktioner som behövde utvecklas. Med hjälp av dessa insikter skapades en nedbruten tidsplan för MVP.a där arbetsmomenten delades in i veckolånga sprintar.

Med avseende på avgränsningarna, utformades implementationen med en integration mot tillträdessystemet Axis A1001. Systemet ansvarar för att administrera och hantera kortanvändare och deras behörigheter för tillträde i dörrmiljöer. Systemet är installerat lokalt hos brukaren och kortläsare är fysiskt anslutna till huvudenheten och innehåller en webbserver där administration sker via ett webinterface. Via ett existerande API i A1001, Vapix®, gavs möjlighet att utbyta data med systemet. Integrationen mot Proximate sker genom att kortanvändare, vilka bör ha giltig behörighet till dörrmiljöer i A1001, ges ett giltigt digitalt tillträdeskort i Proximate mobilapplikation. Kontinuerlig kontroll görs för att säkerställa att eventuella förändringar i tillträdessystemet även avspeglas i Proximate Cloud. Under implementationen av detta projekt, användes en kontrolltid på 5 sekunder. Detta för att få ett snabbt genomslag av förändringar vid tester och utvärdering. I färdig produkt kan detta komma att ändras beroende på hur mycket belastning av det lokala nätverket som kan accepteras. Kontroll med längre mellanrum ger en lägre belastning, men får som konsekvens att ändringar i underliggande system tar längre tid att detekteras och skickas till molntjänsten. Tack vare denna automatiska synkronisering så sker fortsatt all administrering och hantering av kortanvändares behörigheter i tillträdessystemet. Proximate läggs till som en transparent automatisk tilläggstjänst, vilken inte kräver ytterligare hantering för administratörer vid exempelvis förändringar i behörigheter eller för att lägga till och ta bort personer.

Underliggande tillträdessystem är inkopplade på lokala nätverk hos nyttjaren, ofta med mycket begränsad internetåtkomst genom till exempel brandväggar. För att möjliggöra datautbyte mellan AWS och det lokala tillträdessystemet utnyttjades AWS-tjänsten IoT Greengrass, vilken är en mjukvara som kan installeras på en IoT-enhet eller en dator. IoT-enheten kopplas in på det lokala nätverket och kan därmed

kommunicera med tillträdessystemet på ett säkert sätt. Den ges även tillgång till internet och kan då interagera med AWS:s övriga tjänster i molnet.

Som IoT-enhet används Raspberry Pi 4. AWS har väldokumenterade guider och instruktioner för implementation av Greengrass på denna hårdvara.

Alla meddelanden är krypterade med TLS och säkras via certifikat för autentisering. Genom en ”Zero trust”-modell har policyer för Greengrass-enheten etablerats för att enbart ge de behörigheter som krävs för att utföra dess exakta arbetsuppgifter.

IoT Greengrass består av mjukvarukomponenter som kan användas för att samla in, bearbeta och analysera data. Applikationerna kan aktiveras av händelser i molnet eller i det lokala nätverket och kan också köras periodiskt. Greengrass-enheten ansluts till AWS via tjänsten IoT Core. Den fungerar som en knutpunkt mellan AWS och IoT enheter och säkerställer att kommunikation sker på ett säkert sätt. Data kommuniceras via MQTT protokollet.

I projektet används två egenutvecklade mjukvarukomponenter:

- pacsHandlerComponent – identifierar förändringar i underliggande tillträdessystem och ser till att databasen i Proximate Cloud är uppdaterad med de senaste uppgifterna.
- lambdaGgV2CreateCardPacs – skapar nya digitala kort i underliggande tillträdessystem när en ny användare av Proximate registrerar sig.

All data relaterat till digitala kortanvändare samt uppgifter om organisationer och anläggningar vilka är anslutna till Proximate Cloud sparas i databasen DynamoDB. När en ny Greengrass-enhet ansluts för första gången, hämtas all data som är nödvändig för att tjänsten ska fungera från underliggande tillträdessystem och sparas i databasen.

Mobilapplikationen kommunicerar med Proximate Cloud via tjänsten AWS API Gateway. API Gateway utvecklades som ett REST API för kommunikation via HTTPS. För att säkerställa att enbart trafik från Proximate mobilapplikation släpps igenom till molntjänsterna användes tjänsten Cognito. Användare av mobilapplikationen skapar ett inloggningskonto genom att registrera sin epost i Cognito. Vid efterföljande inloggning mot Cognito erhålls en verifikationstoken, vilken sedan används för att medge förfrågningar och trafik mot API Gateway. Tjänsten hanterar

förfrågningar från mobilapplikationen vid registrering och skapande av nytt digitalt tillträdeskort och vid uppdatering av kortstatus. Förfrågningarna vidarebefordras till så kallade Lambda-funktioner, kodblock som kan aktiveras och köras vid interna eller externa händelser. Lambda-funktionen processar inkommande data och interagerar med övriga tjänster, såsom DynamoDB, IoT Core och CloudWatch (AWS logghantering).

5 Resultat

Projektet huvudsakliga mål var att besvara om det gick att implementera en molntjänst som stödde Företagets affärsmodell på ett konkurrenskraftigt sätt. Det andra målet var att skapa en lösning med minimal manuell hantering från administratör.

För att svara på den ekonomiska aspekten gjordes en jämförelse mellan fyra möjliga scenarion. Inhämtade priser är justerade nedåt för att ge en rättvisare bild av alternativa lösningar med hänsyn till ej synliga rabatter. Bägge författarna av denna rapport har jobbat inom säkerhetsbranschen i många år och har diskuterat befintliga lösningar och digitala tillträdeskort med många användare, systemadministratörer och inköpare. Denna erfarenhet i kombination med specifika presentationer av Proximate, ger att *Scenario 1* får antas vara det vanligast förekommande alternativet. Detta valdes således som utgångspunkt för prisjämförelsen. Den sammanlagda kostnaden för detta scenario representeras som 100% i *Fig. 5*, övriga scenarion jämförs mot denna kostnad.

Scenario 1: Kunden äger idag ett befintligt tillträdessystem vilket saknar möjlighet att använda digitala tillträdeskort. Tillträdessystemet kan dock utökas med denna funktionalitet genom att byta ut alla befintliga kortläsare till en nyare modell vilken stödjer digitala tillträdeskort, samt installera tillhörande licenser. Övriga delar av systemet behålls.

Scenario 2: Kunden byter ut det befintliga tillträdessystemet till ett helt nytt system med funktionalitet för digitala tillträdeskort.

Scenario 3: Kunden äger idag ett befintligt tillträdessystem vilket saknar möjlighet att använda digitala tillträdeskort. Kunden väljer att installera ett externt system som erbjuder digitala tillträdeskort. Det nya systemet används parallellt med det befintliga.

Scenario 4: Kunden äger idag ett befintligt tillträdessystem vilket saknar möjlighet att använda digitala tillträdeskort. Systemet kompletteras med integration mot Proximate som ger möjlighet att använda digitala tillträdeskort.

Fig. 5 visar att Proximate kan installeras på ett, för kunden, kostnadseffektivt sätt och att implementationen av Proximate Cloud stödjer affärsmodellen.

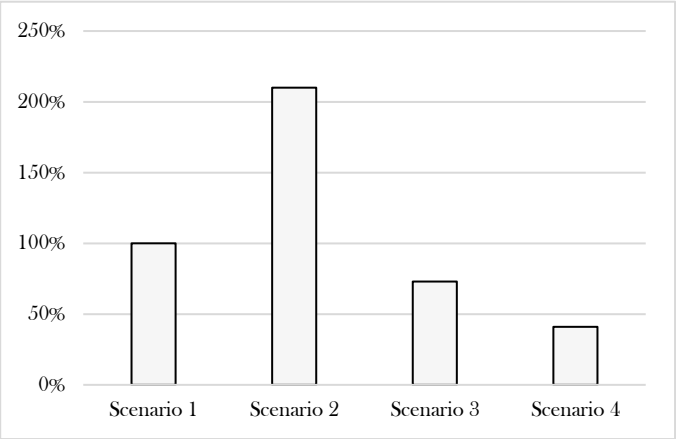


Fig. 5: Prisjämförelse mellan olika scenarion.

Den tekniska implementationen resulterade i en tjänst vilken i huvudsak vilar på nyttjande av olika tjänster från AWS. Den färdiga tjänsten opererar helt transparent och kräver ingen inblandning av administratör när den väl är installerad. Användaradministrationen sker på samma sätt som innan Proximate adderades. Alla flöden är automatiserade.

6 Diskussion och slutsats

De metoder som valdes i projektet har stöttat genomförandet på ett bra sätt. Att arbeta agilt, där experimentella små steg visar vägen, bidrog till att fokusera på rätt moment vid rätt tidpunkt och att inte spendera tid på mindre viktiga insatser. Genomförandet har dock visat att SCRUM-metodik inte tillförde förväntad nytta för en grupp med endast två personer men det kan också bero på ovana hos medlemmarna.

I designfasen valdes typen av databas med hänsyn till relaterade kostnader, där AWS erbjuder mycket låga avgifter för skrivning och läsning till DynamoDB. Inledningsvis var avsikten att använda en relationsdatabas, baserat på dess fördelar med bättre kontroll av införda data via SQL-scheman. Det hade dock krävt betydligt mer omfattande designarbete och planering. Avsaknaden av vissa kontrollmekanismer, så som tydlig definiering av attribut, krävde dock extra vaksamhet vid implementation för att säkerställa att rätt data lagras.

I traditionella relationsdatabaser används scheman för att specificera och strukturera datatyper och datafält i förväg. Detta möjliggör strikta regler och begränsningar för vilka typer av data som kan lagras och vilka fält som ska vara obligatoriska eller valfria. I en schemalös databas kan dock användaren dynamiskt skapa och lägga till nya datafält utan någon form av förhandskontroll. Detta ger en stor flexibilitet, men det innebär också att det inte finns någon mekanism som kan garantera att datamodellen följer en viss struktur eller att alla nödvändiga fält finns. Det kan resultera i situationer där vissa dokument eller poster i databasen saknar viktiga datafält eller innehåller ogiltiga eller inkonsekventa data.

Trots att avgränsningen ursprungligen gjordes för att utveckla en tjänst som skulle integrera med ett specifikt tillträdessystem, har arbetet i stor utsträckning strävat efter att generalisera så många områden som möjligt. Detta val gjordes av flera skäl. För det första ville vi skapa ett system som kräver minimalt eller inget ingripande från administratören, varken för daglig drift eller uppstart. För det andra ville vi, med en relativt liten insats utöver detta projekt, kunna utöka tjänsten med integration mot flera andra tillträdessystem. Detta åstadkoms bland annat genom generalisering av komponenter där kommunikation med tillträdessystem sker via

metoder vilka är definierade i ett interface. Integration med ett nytt tillträdessystem kräver enbart en ny klass som implementerar detta interface.

Bättre planering och större prioritering av kod-tester borde utförts i ett tidigare stadie i projektet. Att skriva tester för framför allt funktioner som agerar på data i molnet visade sig vara mer komplicerat än förutspått [10]. Om tester skrivits först, eller parallellt, i stället för som nu i slutet av projektet, kanske vissa problem hade visat sig snabbare [11]. När väl tester infördes identifierades snabbt fel som kunde åtgärdas. Hade tester skrivits först så kanske mindre kod hade krävt refaktorisering.

Utvärderingen mellan AWS och Azure visade att det skiljer väldigt lite mellan dessa tjänster, både kostnadsmässigt och tekniskt. Bedömningen är således att inlåsningseffekten är relativt låg och att det skulle vara tämligen enkelt att byta leverantör.

Projektet har lyckats uppfylla sina mål genom att utveckla en kostnadseffektiv molntjänst som stödjer företagets affärsmodell på ett konkurrenskraftigt sätt och som kräver minimal manuell hantering från administratören. Projektet kan betraktas som en framgång och den utvecklade tjänsten förväntas möta behoven hos företagets kunder på ett tillfredsställande sätt.

7 Referenser

[1] K. Lundstrom, ”A TITANIC-SIZED WASTE PROBLEM: <E-WASTE IS PILING UP, AND BRANDS ARE FINALLY CONSIDERING THE LONG-TERM RAMFICATIONS,” *Adweek,* vol. 63, nr 5, p. 2, 4 April 2022.

[2] P. Flewelling, ”The the Agile Developer's Handbook : Get More Value from Your Software Development: Get the Best Out of the Agile Methodology,” Packt Publishing, Limited, Birmingham, 2018.

[3] G. Guerrero-Ulloa, C. Rodríguez-Domínguez och M. J. Hornos, ”Agile Methodologies Applied to the Development of Internet of Things (IoT)-Based Systems: A Review,” *Sensors,* vol. 23, nr 2, pp. 790-825, 2023.

[4] H. Portman, ”Sensemaking in the Agile Forest: The Minimum Viable Product (MVP) unraveled,” *PM World Journal,* vol. 11, nr 8, pp. 1-5, 2022.

[5] S. Chacon och B. Straub, Pro Git, Springer Nature, 2014.

[6] M. Klems, AWS Lambda Quick Start - Learn how to build and deploy serverless applications on AWS, Packt Publishing, 2018.

[7] S. Rose, O. Borchert, S. Mitchell och S. Connelly, ”Zero Trust Architecture,” 08 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf.

[8] T. Vijayakumar, Practical API Architecture and Development with Azure and AWS, Colombo, Sri Lanka: Springer Science+Business Media, 2018.

[9] D. Goovaerts, ”Amazon, Google and Microsoft now own 66% of the cloud market.,” *Fierce Telecom,* p. 1, 28 Oktober 2022.

[10] S. Nachiyappan och S. Justus, ”Cloud Testing Tools and its Challenges: A Comparative Study,” *Procedia Computer Science,* vol. 50, pp. 482-489, 2015.

[11] F. Taufiqurrahman, S. Widowati och M. J. Alibasa, ”The Impacts of Test Driven Development on Code Coverage,” i *The 1st International Conference on Software Engineering and Information Technology (ICoSEIT)*, Bandung, Indonesia, 2022.

[12] S. Seth, ”Going Green with AWS - A Transition Towards a Sustainable Future,” *CIO,* p. 1, 9 Juli 2022.

[13] FN, ”Agenda 2030 och de globala målen för hållbar utveckling - ett informationsmaterial från svenska FN-förbundet,” September 2018. [Online]. Available: https://fn.se/wp-content/uploads/2018/10/Infomaterial_Agenda3030_komprimerad.pdf. [Använd 04 05 2023].

8 Appendix A: Hållbara och etiska aspekter av serverless computing

I vår tid, då elektronikavfall blir allt vanligare, har Företaget identifierat en möjlighet att minska mängden elektronikskrot genom att införa digitala tillträdeskort i kundernas befintliga tillträdessystem. Enligt företagets hypotes kan denna innovativa lösning förlänga livslängden på dessa system med upp till 10 år eller ännu längre. Detta skulle bidra till att minska den negativa påverkan på miljön.

Varje år ökar mängden elektronikavfall, och endast en tredjedel av detta avfall återvinns [1]. Det är tydligt att det är nödvändigt att minimera denna typ av avfall så mycket som möjligt för att skydda vår planet och framtida generationer.

Genom att ersätta fysiska tillträdeskort och taggar med digitala alternativ minskas också användningen av plast. Plasten som tidigare skulle ha behövts för tillverkningen av dessa kort och taggar kan nu undvikas. Detta är en positiv konsekvens av övergången till digitala tillträdeskort och en ytterligare fördel för miljön.

Därför är det viktigt att fortsätta utforska och främja tekniska innovationer som kan bidra till att minska elektronikskrot, öka återvinningen och minska användningen av resurser som plast. Genom att ta vara på digitala möjligheter kan vi alla bidra till en mer hållbar framtid.

Serverless computing har fördelar ur hållbarhetsperspektiv, eftersom det endast använder resurser när de behövs och därigenom minimerar körtid. Detta kan bidra till att minska onödig användning av energi och därmed ha en positiv effekt på miljön [12].

Dessutom kan serverless computing hjälpa till att jämna ut konkurrensen mellan startups och etablerade företag, då den tillåter mindre företag att börja smått med minimala initiala kostnader för IT-infrastruktur, för att sedan dynamiskt anpassa sig allt eftersom verksamheten växer. Detta faller inom ramen får mål 10 i FN:s globala mål för hållbar utveckling, vilket fokuserar på minskad ojämlikhet [13].

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2023
www.chalmers.se

GÖTEBORGS
UNIVERSITET

CHALMERS