CHALMERS

Kommunikationskanal för social verksamhet

Utveckling av en trygg kommunikationskanal med pushnotiser för fritidsgårdar, i en spelbaserad applikation för Android och iOS

Examensarbete inom högskoleingenjörsprogrammet i datateknik

JEAN-PAUL KHALAF, ERIK ULFUNG ARNBOM

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2025
www.chalmers.se

Examensarbete 2025

Kommunikationskanal för social verksamhet

Utveckling av en trygg kommunikationskanal med pushnotiser för fritidsgårdar, i en spelbaserad applikation för Android och iOS

Jean-Paul Khalaf, Erik Ulfung Arnbom

CHALMERS

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborg 2025

Kommunikationskanal för social verksamhet
Utveckling av en trygg kommunikationskanal med pushnotiser för fritidsgårdar, i en spelbaserad applikation för Android och iOS
Jean-Paul Khalaf, Erik Ulfung Arnbom

Handledare:
Sakib Sistek, Institutionen för Data- och Informationsteknik
Jenny Forsberg, Omegapoint
Examinator: Nicholas Smallbone, Institutionen för Data- och Informationsteknik

Examensarbete 2025
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: Appikonen för mobilapplikationen.

Skriven i LaTeX
Göteborg 2025

Kommunikationskanal för social verksamhet
Utveckling av en trygg kommunikationskanal med pushnotiser för fritidsgårdar, i en spelbaserad applikation för Android och iOS
Jean-Paul Khalaf, Erik Ulfung Arnbom
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola

Abstract

This report presents the design, implementation and evaluation of a push notification system integrated into an existing Unity-based mobile application for a youth center. An agile, sprint-driven methodology guided the work, which involved selecting Firebase Cloud Messaging (FCM) for cross-platform notification delivery, developing a Node.js/TypeScript REST API with Express and a PostgreSQL backend containerized via Docker, and creating an administrative web interface to trigger notifications. Key challenges addressed include secure token handling through FCM topics, platform-specific configuration in Unity for Android and iOS, and maintaining a reliable message history within the app. The system was validated through emulator and real-device tests (Google Play Console, Apple TestFlight) and user feedback sessions. The resulting solution offers a robust, scalable communication channel balancing technical functionality and usability.

Sammanfattning

Denna rapport beskriver utvecklingen och utvärderingen av ett pushnotissystem som integrerades i en befintlig Unity-baserad mobilapplikation för en fritidsgård. Arbetsmetoden byggde på agila sprintar och omfattade teknikvalet av Firebase Cloud Messaging (FCM) för plattformsoberoende notisleverans, utveckling av ett REST-API i Node.js/TypeScript med Express, samt en PostgreSQL-databas containeriserad med Docker. Systemet inkluderar säker hantering av notifierings-token via FCM topics, plattformspecifika konfigurationer i Unity för Android och iOS samt funktion för att visa meddelandehistorik i appen. Utvecklingen validerades genom emulator- och enhetstester via Google Play Console och Apple TestFlight, kompletterat med användarstudier. Lösningen erbjuder en skalbar, användarvänlig kommunikationskanal mellan fritidsgårdens administratörer och appens användare.

Nyckelord: pushnotiser, Firebase Cloud Messaging, Unity, Android, iOS, mobilapplikation, backend, Node.js, TypeScript, PostgreSQL, Docker, användartester.

Förord

Vi vill börja med att rikta varma tack till vår handledare på Omegapoint Jenny Forsberg, för hennes kontinuerliga stöd, insikter och vägledning under hela projektet. Mycket tacksamma är vi även för vår handledare Sakib Sistek vid institutionen för Data- och Informationsteknik, för hans sakkunskap, konstruktiva återkoppling och uppmuntran. Ett särskilt tack riktas till Nicholas Smallbone och Johannes Åman Pohjola, examinatorer vid Institutionen för Data- och Informationsteknik, för deras noggranna granskning och bidrag till att säkerställa uppsatsens akademiska standard. Vi vill dessutom tacka Omegapoint för att de generöst gav oss tillgång till sina lokaler och den infrastruktur som gjorde detta arbete möjligt. Slutligen vill vi tacka Reningsborg för deras viktiga insatser för social och miljömässig hållbarhet i samhället, och för att de genom sina verksamheter skapar meningsfulla möjligheter för människor i behov av stöd.

Jean-Paul Khalaf & Erik Ulfung Arnbom, Göteborg, Maj 2025

Akronymer

Nedan följer en lista över akronymer som använts i denna avhandling, sorterade i alfabetisk ordning:

API	Application Programming Interface
AWS	Amazon Web Services
FCM	Firebase Cloud Messaging
GDPR	General Data Protection Regulation
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
iOS	iPhone Operating System
OAuth	Open Authorization
REST	Representational State Transfer
SDK	Software Development Kit
SQL	Structured Query Language
UXML	Unity XML
USS	Unity Style Sheets
UI	User Interface

Innehåll

Figurer

1

Inledning

1.1 Bakgrund

Fritidsgårdar spelar en viktig roll i att stödja barns och ungdomars utveckling genom att erbjuda en trygg miljö för social interaktion och lärande. Faktum är att över 90% av svenska barn och ungdomar i åldern 8-19 år har en egen mobiltelefon, vilket innebär att en mobilbaserad kommunikationskanal har potential att nå ut brett i målgruppen [1]. I en allt mer digitaliserad värld finns ett behov av att modernisera hur fritidsgårdar kommunicerar med de barn och ungdomar som deltar i deras verksamhet, samt vid behov, med deras vårdnadshavare. Vissa kommuner har redan påbörjat denna digitala resa, till exempel erbjuder Tjörns kommun mobilapplikationen Ung på Tjörn, där ungdomar får information om aktuella aktiviteter och kan ta emot pushnotiser med tips och inspiration [2]. En digital kommunikationskanal kan möjliggöra snabb och enkel kontakt, men måste också vara säker och anpassad efter användarnas behov och förväntningar.

1.2 Syfte

Detta examensarbete syftar till att undersöka och utveckla en trygg och säker kommunikationskanal som integreras i en befintlig spelbaserad applikation för mobiltelefoner. Målet är att skapa en lösning som gör det möjligt för fritidsgårdens personal att skicka meddelanden och pushnotiser till barn, ungdomar och vid behov även deras föräldrar, utan att barnen kan kommunicera med varandra via samma kanal.

1.3 Mål

Målet med projektet är att utveckla en trygg och säker kommunikationskanal med pushnotiser som integreras i en befintlig spelbaserad applikation. För att uppnå detta ska projektet:

- Identifiera krav och säkerhetsaspekter för en säker kommunikationskanal med pushnotiser, med särskild hänsyn till användargruppernas behov, GDPR och barns säkerhet online.
- Utveckla och implementera en kommunikationsfunktion som möjliggör envägskommunikation från fritidsgårdar till barn, ungdomar och deras vårdnadsha-

vare, anpassad för både Android och iOS.

- Säkerställa att systemet skyddar användarnas integritet och minimerar risker för missbruk genom att förhindra obehörig åtkomst och upprätthålla en hög säkerhetsnivå.
- Testa och utvärdera funktionaliteten tillsammans med potentiella användare, såsom fritidsgårdspersonal, för att säkerställa att lösningen uppfyller både tekniska och praktiska krav.

1.4 Avgränsningar

Projektet fokuserar enbart på utvecklingen av en kommunikationskanal inom en befintlig spelbaserad applikation och inkluderar inte utveckling av nya spelfunktioner. Vidare begränsas projektet till envägskommunikation från fritidsgårdspersonal till ungdomar och vårdnadshavare; tvåvägskommunikation och chattfunktioner ingår inte i lösningen. Slutligen ligger långsiktig drift, underhåll och användartester bortom den tekniska implementation som beskrivs i rapporten.

Det är viktigt att notera att denna rapport inte utgör juridisk rådgivning, och att detaljerad tolkning av dataskyddslagstiftningen (GDPR) inte omfattas av projektet. För fullständig efterlevnad av dataskyddsregelverket rekommenderas extern juridisk granskning.

2

Metod

Följande kapitel beskriver arbetsmetod och de verktyg som användes under projektets gång. Det inkluderar både förberedelser inför utvecklingsarbetet och de tekniska lösningar som låg till grund för implementationen av pushnotiser i applikationen. Syftet med projektet var att utveckla ett robust och användarvänligt kommunikationssystem för en fritidsgård, där administratörer enkelt kan skicka pushnotiser till användare via en mobilapplikation.

2.1 Arbetsmetod

Projektet genomfördes med ett agilt arbetssätt inspirerat av Scrum. Arbetet delades in i veckovisa sprints med planerade mål, där varje sprint avslutades med ett möte tillsammans med handledaren Jenny Forsberg från Omegapoint, Elisabeth Valinder från Reningsborg samt de två andra projektgrupperna som arbetade med samma applikation. Valet att arbeta agilt med Scrum motiverades av projektets iterativa kravbild och behovet av snabba anpassningar, vilket en traditionell vattenfallsmetod inte hade kunnat erbjuda. Under dessa möten diskuterades framsteg, eventuella hinder samt nya krav som uppstått.

Projektgruppen delades upp i tre huvudansvarsområden: spelutveckling i Unity, utveckling av webbsida samt implementation av notiser och tillhörande backend, vilket denna rapport fokuserar på. Då en annan projektgrupp parallellt ansvarade för spelkomponenter i Unity, koordinerades arbetet genom veckovisa kodsammanfogningar snarare än ett gemensamt Git-repo. Detta val grundades i Unity-projektets storlek och struktur, som gjorde det opraktiskt att hantera via exempelvis GitHub[3]. Vidare upptäcktes det tidigt att Unity inte hanterade nya filer, eller ändringar i filer, på ett bra sätt om man inte utförde detta direkt i Unitys egna programvara, vilket ytterligare försvårade användning av Git och GitHub direkt relaterad till Unity.

Projektet utgick från en tidigare version av applikationen som utvecklats av en tidigare grupp. Det innebar att grundläggande funktionalitet såsom gränssnitt och viss spelmekanik redan fanns implementerad. Fokus låg därför på att bygga och integrera ett säkert och funktionellt system för pushnotiser ovanpå den befintliga koden.

Kommunikationsfunktionerna och pushnotissystemet utvecklades som ett separat tekniskt tillägg till spelet. Arbetet fokuserade på säker, envägsmeddelandehantering

via Firebase Cloud Messaging (FCM) och en webbsida där administratörer enkelt kan skicka meddelanden.

För att hantera lagring av användardata och loggade notiser användes en relationsdatabas i PostgreSQL. Databasens roll i systemarkitekturen beskrivs mer utförligt i avsnitt 3.5.

Sammanfattningsvis lade den agila metoden grunden för ett flexibelt och kvalitetssäkrat utvecklingsarbete.

2.2 Testning

För att säkerställa att pushnotiserna fungerade korrekt genomfördes omfattande tester. Initialt testades funktionaliteten med Android Studio-emulatorer. Därefter laddades applikationen upp till Google Play och Apple TestFlight [4] för att kunna testas på riktiga enheter. Slutligen genomfördes användartester i samarbete med personal och ungdomar på fritidsgården, som gav värdefull feedback kring systemets funktionalitet och användbarhet. Denna återkoppling användes för att iterativt förbättra applikationen.

2.3 Verktyg

Detta avsnitt beskriver de verktyg och utvecklingsmiljöer som användes i projektet.

2.3.1 Unity

Unity användes som spelmotor för utvecklingen av själva applikationen. Plattformen erbjöd en visuell och relativt lättanvänd miljö för att bygga gränssnitt, hantera användarflöden och integrera funktionalitet kopplad till spelet.

2.3.2 Utvecklingsmiljöer

För utvecklingsarbetet användes flera olika IDE:er beroende på syfte och operativsystem.

Visual Studio användes på Windows-datorer för arbete med Unity och C# [5], eftersom det erbjöd smidig integration och goda felsökningsmöjligheter.

JetBrains Rider användes på macOS i motsvarande syfte. Rider har inbyggt stöd för Unity [6] och valdes då Visual Studio inte längre stöds på macOS, och generellt anses vara det bästa alternativet på detta operativsystem.

Visual Studio Code, som är ett fristående verktyg från Visual Studio, användes huvudsakligen för backendarbete såsom Node.js-servern, konfiguration av API:er och databasanslutningar. Den höga anpassningsbarheten och stora plugin-stödet gjorde det till ett naturligt val för detta arbete.

2.3.3 GitHub

GitHub användes för versionshantering av backendkod och mindre testkomponenter. Arbetet organiserades i olika grenar (branches) med pull requests för att möjliggöra parallellt arbete och kodgranskning, både inom projektgruppen och i samverkan med extern part.

2.3.4 Xcode

Xcode är Apples officiella utvecklingsmiljö för macOS, iOS och andra Apple-plattformar. För detta projekt användes Xcode för att paketera och distribuera Unity-applikationen till iOS. Unity genererar ett Xcode-projekt vid export, vilket därefter öppnades i Xcode för vidare hantering. Där genomfördes bland annat konfiguration av appens identitet (bundle identifier), signering med rätt certifikat [7], samt inställningar för notifikationsrättigheter. Applikationen distribuerades via Xcode till Apple Developer-portalen för vidare testning med TestFlight och eventuell publicering på App Store.

2.3.5 Adobe Photoshop

För att förbereda applikationen för distribution på App Store och Google Play krävdes viss grafisk bearbetning. Adobe Photoshop användes för att redigera och anpassa bilder, inklusive appens logotyp och skärmdumpar, enligt de krav som ställs av respektive plattform [8]. Detta innefattade att ändra bildstorlek, anpassa upplösning samt optimera bildmaterialet för att uppfylla App Stores riktlinjer för visuell presentation.

2.3.6 Sammanslagning av Unity-projekt

Eftersom Unity-projektet hanterades av flera grupper med separata kodbaser användes inte GitHub som versionshanteringssystem för själva spelet. I stället genomfördes manuella, veckovisa sammanslagningar för att integrera kommunikationsfunktionen i spelet. Detta beslut grundade sig i projektets omfattning samt tekniska och ekonomiska begränsningar kopplade till Unitys molntjänster, då exempelvis Unity Version Control [9] och andra inbyggda samarbetslösningar krävde antingen betalabonnemang eller hade begränsningar på antal användare.

2.3.7 Postman

Postman användes för att testa API-endpoints och skicka testnotiser [10]. Verktyget bidrog till att validera backendlogik och säkerställa korrekt integration med Firebase Cloud Messaging.

2.3.8 Firebase

Firebase användes som plattform för att hantera pushnotiser [11]. FCM integrerades i Unity-applikationen, vilket möjliggjorde kommunikation från backend till använ-

darnas enheter. Under utvecklingen användes även Firebase Console och Firebase Admin SDK för testning, felsökning och integration.

2.3.9 Docker

Docker användes för att containerisera PostgreSQL-databasen innan den distribuerades till AWS. Genom att skapa en Docker-container kunde databasen enkelt konfigureras, köras lokalt under utveckling samt smidigt flyttas till produktionsmiljö [12]. Detta förenklade installationsprocessen, minskade risken för miljörelaterade fel och säkerställde att databasen fungerade likadant i utvecklings- som i driftmiljö.

2.3.10 Hosting

Projektets backend och databas planerades att hostas på Reningsborgs egna AWS-servrar, som körs i en Linuxbaserad servermiljö. Även databasen i PostgreSQL kommer att driftas där. Valet av egen hosting gav bättre kostnadskontroll och ökad säkerhet jämfört med externa molntjänster.

3

Teknisk bakgrund

I detta kapitel presenteras de viktigaste tekniska komponenterna och ramverk som utgör grunden för implementationen av pushnotissystemet. Varje avsnitt beskriver kortfattat vad tekniken är, varför den valdes och hur den använts i projektet.

3.1 C#

C# är det primära programmeringsspråk som används i Unity-motorn för att implementera applikationslogik, UI-komponenter och integrationer med externa SDK:er [5]. I detta projekt har C# använts för att initiera Firebase SDK, hantera in- och utgående pushnotiser samt bygga gränssnittet för notishistorik i mobilappen.

3.2 JavaScript

JavaScript [13] användes inledningvis i backenden för att skriva serverns affärslogik. Detta ändrades senare under projektets gång till TypeScript på begäran av gruppen som utvecklade webbsidan. Genom Node.js-miljön kördes JavaScript-kod som hanterade REST-API, routning och kommunikation med Firebase Admin SDK. JavaScript valdes för sin enkelhet vid snabb prototypning och breda ekosystem med tillgängliga paket.

3.3 TypeScript

Som tidigare nämnts migrerades servern under projektets gång från JavaScript till TypeScript, och användes på serversidan med Node.js för att införa statisk typkontroll och tydliga kontrakt mellan olika delar av backenden [14]. Eftersom TypeScript utökar funktionaliteten hos JavaScript[15] kunde befintlig kod köras oförändrad, samtidigt som, samt API-scheman och felhantering skötas på ett mer robust sätt än med ren JavaScript. Detta bidrog till färre buggar och bättre läsbarhet i den asynkrona koden som ansvarar för pushnotisskick, poänghantering och databasåtkomst.

3.4 Node.js

Node.js är en händelsestyrd JavaScript-körmiljö som möjliggör snabb och skalbar hantering av HTTP-förfrågningar. Backenden, byggd med Node.js och Express[16], utgör mellanlager mellan webbgränssnittet, Firebase och databasen. Den asynkrona arkitekturen är särskilt lämpad för I/O-intensiva operationer som massutskick av notiser.

3.5 SQL (PostgreSQL)

PostgreSQL är den relationsdatabas som valdes för att spara användarinformation, poäng och loggade notiser[17]. Med SQL definierades tabeller, relationsstrukturer och queries för att skapa, läsa, uppdatera och radera data. Databasen körs i Docker-containrar både lokalt i utveckling och produktionssatt på AWS, vilket garanterar konsekventa miljöer.

3.6 React

React är ett bibliotek för att bygga användargränssnitt i webbapplikationer [18]. Även om frontenden utvecklades av en annan grupp, kommunicerar deras React-app med vår backend via REST-API:er. React användes för att skapa komponenter för notisskick, poängadministration och användargodkännande, vilket ställde krav på tydliga och väldokumenterade API-kontrakt på serversidan.

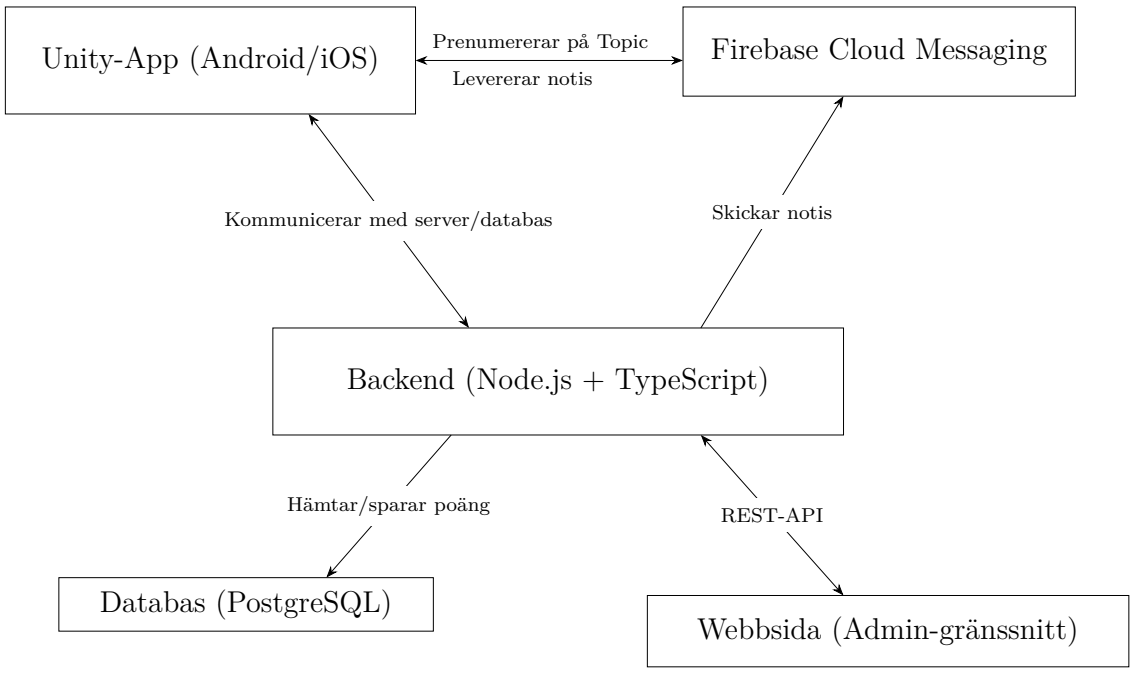
3.7 UXML

UXML är XML-baserat format för att beskriva UI-layout i Unitys UI Toolkit. I detta projekt har UXML använts för att definiera strukturen för notishistorik-sidan i appen, medan tillhörande USS (Unity Style Sheets) hanterar visuella stilar. UI Toolkit ger responsiva gränssnitt som fungerar väl på olika mobila skärmstorlekar.

4

Genomförande

Detta kapitel beskriver det praktiska arbetet som utfördes för att utveckla och implementera ett pushnotissystem i en befintlig mobilapplikation byggd i Unity. Projektet genomfördes i flera steg som inkluderade förstudie, implementation av notisfunktionalitet, utveckling av backend och databas, integration med ett externutvecklat webbgränssnitt, samt olika tester och hantering av tekniska problem. Målet var att skapa ett komplett och integrerat system som möjliggör kommunikation mellan fritidsgårdens administratörer och appens användare via pushnotiser. Figur 4.1 illustrerar systemarkitekturen.

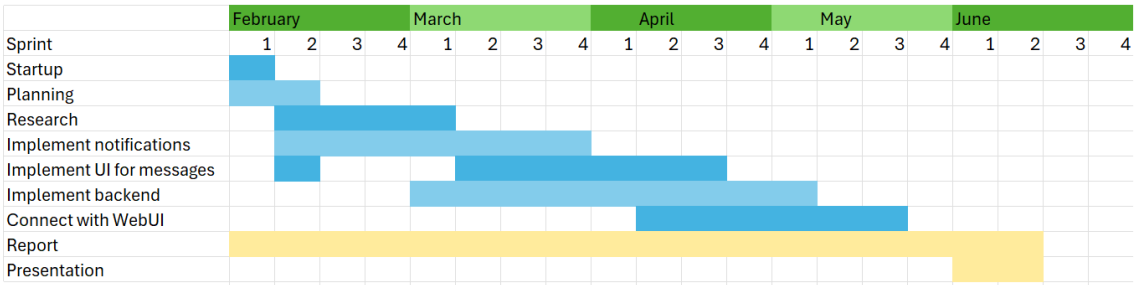


Figur 4.1: Systemarkitektur: Interaktion mellan Unity-app, Firebase, backend, databas och webbsida.

4.1 Planering

Projektet inleddes med ett uppstartsmöte under den första veckan, följt av en planeringsfas där mål, krav och tekniska lösningar diskuterades. Därefter påbörjades

undersökningsfasen parallellt med den första implementeringen av viktiga funktioner, såsom användargränssnittet för meddelanden och pushnotiser. Denna planerade arbetsstruktur lade grunden för ett organiserat och metodiskt utvecklingsarbete.



Figur 4.2: Gantt-schema

4.2 Förstudie och val av teknik

För att hitta en lämplig teknisk lösning inleddes projektet med en omfattande undersökning av olika teknologier och verktyg för att implementera pushnotiser i en mobilapplikation byggd i Unity. Genom att söka igenom dokumentation, guider, forum och tutorials kunde olika alternativ kartläggas. Efter en jämförelse föll valet på Firebase Cloud Messaging (FCM), främst på grund av dess goda stöd för plattformsoberoende notiser, enkel integration med Unity, samt ett väldokumenterat SDK [11]. Valet underlättades även av att Firebase tillhandahåller ett administrativt webbgränssnitt och ett serverbaserat SDK för backendintegration, vilket passade projektets struktur.

4.3 Implementering av Firebase i Unity

Det första konkreta utvecklingssteget var att integrera Firebase i Unity-applikationen [19]. Detta moment visade sig mer komplext än väntat, då konfigurationen skiljer sig åt mellan Android och iOS [20]. Arbetet påbörjades med Android genom att skapa en Firebase-projektstruktur och integrera nödvändiga SDK-filer. Efter detta arbetades det med att initiera Firebase i Unitys startflöde och hämta den token som varje enhet tilldelas. Denna token används normalt sett för att adressera notiser till en specifik mottagare, vilket inledningsvis var planen för projektet.

4.4 Övergång från tokens till topics

När den tekniska grunden hade etablerats och hanteringen av enhetstokens hade testats identifierades att den valda metoden kunde stå i strid med Dataskyddsförordningen (GDPR) [21], eftersom tokens kan betraktas som personuppgifter om de går att koppla till en individ eller en specifik användarenhet. I samråd med handledare och med beaktande av dataskyddsprinciperna fattades därför beslutet att i stället använda den topic-baserade funktionaliteten i Firebase Cloud Messaging

(FCM). Topics möjliggör gruppering av enheter utan att identifierande information behöver lagras, vilket innebär att notiser kan skickas till en grupp mottagare, till exempel samtliga användare, utan att backend-systemet har kännedom om deras individuella enhetsidentiteter. Detta eftersom prenumerationen på ett topic sker lokalt på respektive enhet och information om enhetens tillhörighet inte görs tillgänglig för avsändaren. Denna lösning bedömdes som mer lämplig ur ett integritetsperspektiv, samtidigt som den förenklade utvecklingsprocessen och möjliggjorde bredare och mer generella notisutskick utan att känslig information behövde behandlas.

4.5 Utveckling av backend

Efter att notisfunktionen var stabilt integrerad i applikationen riktades fokus mot att bygga en backend som kunde fungera som länk mellan webbsidan och Firebase. Backendutvecklingen gjordes i Node.js och TypeScript [14], där det skapades ett REST-baserat API med Express [16] för att hantera olika typer av kommunikation. Bland annat byggdes funktioner för autentisering, hantering av notiser, poängsystem och användarvalidering. Firebase Admin SDK integrerades i servermiljön för att möjliggöra programmatisk kommunikation med FCM från backend. Denna lösning möjliggjorde att en extern webbsida kunde trigga notisutskick direkt från sin användarvy utan att behöva interagera med Firebase-konsolen.

4.6 Ursprunglig design

I den ursprungliga versionen av appen som togs över för detta projekt, kan ses i figur 4.3, möttes användaren av en startsida med en grön bakgrund. Gränssnittet bestod av tre vertikalt staplade knappar: ”Poäng”, ”Farm” och ”SkaparSafari”. Poängknappen ledde till en skärm där man genom ett engångslösenord, som i sin tur erhölls genom att logga in som administratör via kugghjulsknappen, kunde tilldela poäng till olika användare. Farmknappen ledde till själva spelet vilket fortsatte att utvecklas av den externa spelgruppen för projektet. SkaparSafari var ett annat spel som tidigt under projektets gång bestämdes att skrotas. Det fanns vid detta stadium inget stöd för att välja grupp eller ange ett användarnamn, vilket var nödvändigt för önskade funktioner relaterade till notiserna. Detta enkla upplägg fungerade som utgångspunkt för samtliga förbättringar som implementerades under projektets gång.

Figur 4.3: Ursprunglig hemskärm

Figur 4.4: Ursprunglig poängsida

4.7 Säkerhetsmöte

För att säkerställa säkerheten i projektet hölls den 1 april 2025 ett möte [22] med Pontus Hanssen, säkerhetsresearcher och penetrationstestare på Omegapoint, där värdefull information och förslag kring tänkbara skyddsåtgärder togs emot. Råd som avsåg webbplatsen eller andra delar av projektet som utvecklats av annan part, har utelämnats. Pontus rekommenderade att samtliga API-endpoints skyddas med API-nycklar där vikten av regelbunden rotation togs upp. Mer omfattande lösningar som OAuth bedömdes bli för tidskrävande i projektets nuvarande fas, samt att denna nivå av säkerhet bedömdes vara tillräcklig med hänsyn till projektets omfattning.

4.8 Integration med webbsida

Webbsidan utvecklades av extern part, men kontinuerlig dialog fördes med den ansvariga utvecklaren. Genom denna kommunikation kunde det säkerställas att backend exponerade rätt API-endpoints. Samarbetet gjorde det möjligt att koppla ihop backenden med webbsidan och därigenom möjliggöra funktioner såsom notisutskick, poänghantering och användargodkännande. Webbsidans interaktion med server kan ses i figur 4.1

4.9 Databaskoppling och poängsystem

Efter att notisflödet mellan webbsida, backend och mobilapp var på plats utökades systemet med ett närvaropoängsystem. Detta krävde att användardata och poäng lagrades i en relationsdatabas. En PostgreSQL-databas [17] sattes upp och containeriserades med Docker [12], vilket förenklade testning och framtida driftsättning. Backend kopplades därefter till databasen och det implementerade funktioner för att skapa, läsa, uppdatera och radera data i databasen.

4.10 Implementation av gruppval

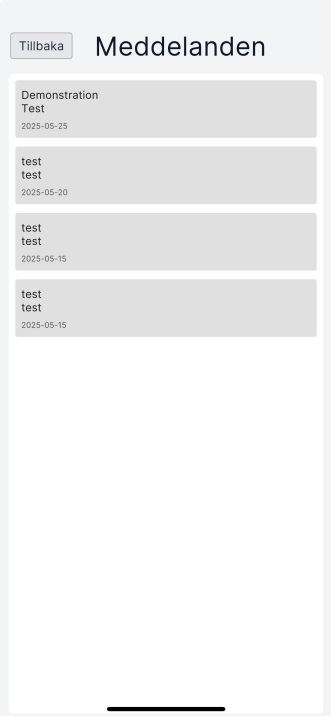
På önskemål från Reningsborg, som har flera ålderskategorier och grupper på fritidsgården, implementerades i appen en ruta som man når genom att klicka på Färg-knappen i huvudmenyn (se figur 5.2). Detta öppnar upp en ruta med en knapp för varje gruppfärg. Varje färgknapp är bunden till och prenumererar på motsvarande FCM-topic, vilket ser till att man tar emot notiser avsedda för rätt grupp.

4.11 Meddelandehistorik

För att stärka tillförlitligheten i det notissystem som utvecklats implementerades även en funktion i applikationen som visar användaren ett arkiv över tidigare mottagna meddelanden. Bakgrunden till denna funktion var insikten att användarna inte alltid har möjlighet att läsa pushnotiser i realtid. Dessutom finns risker med att viktiga meddelanden försvinner, exempelvis vid enhetens omstart, byte av telefon, rensning av notiser eller om användaren av misstag sveper bort en notis utan att läsa den. Därför behövdes ett sätt att återge notisinformationen i efterhand, direkt i applikationen, på ett sätt som är både överskådligt och lättillgängligt.

Den tekniska lösningen består av ett system där alla notiser som skickas från backend till Firebase Cloud Messaging även loggas i en databas tillsammans med metadata såsom tidpunkt, målgrupp (topic) och notistext. När användaren öppnar appens historiksida görs ett API-anrop till backend som returnerar alla notiser kopplade till de topics användaren är prenumererad på under de senaste 30 dagarna. Detta gör att varje användare får ett personligt anpassat meddelandeflöde, utan att det krävs individuell lagring av notifieringsdata direkt på klienten.

Meddelandena presenteras i appen i omvänd kronologisk ordning med de nyaste först. Varje notis innehåller en rubrik, datum samt meddelandetext. Gränssnittet är designat för att vara lättöverskådligt, med stor, tydlig text och enkel navigering anpassad för mobilanvändning, se figur 4.5. Funktionen fungerar som en digital anslagstavla, vilket innebär att användaren i efterhand kan kontrollera vad som har kommunicerats från fritidsgården, exempelvis tid och plats för aktiviteter, förändringar i schema eller andra meddelanden av betydelse. På så vis minimeras risken att information går förlorad. Vidare kan endast användare som verifierats av administratör via webbsidan, se meddelandehistoriken samt ta emot notiser. Detta stoppar obehöriga som laddar ner applikationen från att få information från fritidsgården, men tillåter dessa ändå att ta del av spelet i appen.



Figur 4.5: Sida för meddelanden

4.12 Testning och felsökning

Pushnotiserna testades först i Android Studio Emulator för att säkerställa grundläggande funktionalitet. Därefter laddades testversioner av applikationen upp till Google Play Console och Apple TestFlight [4] för att möjliggöra tester på riktiga enheter. Under testfasen genomfördes både tekniska och användarbaserade tester tillsammans med personal på fritidsgården. Feedback från dessa tester låg till grund för justeringar i funktionalitet, notistext och felhantering.

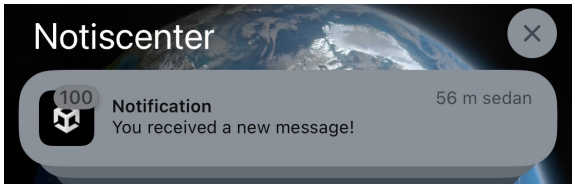
Testning av funktioner och API-anrop till servern gjordes först till en Postgresdatabas på personlig dator. Tester genomfördes över lokalt nätverk och efter att korrekt funktionalitet säkerställts, skickades dockeriserade lösningen vidare för hosting på AWS server.

Under utvecklingen och testningen av appen identifierades ett tekniskt problem i hanteringen av pushnotiser på iOS. Vid tester i både simulator och på fysiska enheter visades samma notis upprepade gånger, vilket resulterade i en oändlig rundgång där användaren fick se flera ”You recieved a new message!” i rad. Se figur 4.6.

Felet berodde på vår manuella förgrundsvisningsfunktion för notiser. Syftet var att säkerställa att notiser alltid visades, men funktionen aktiverades även på iOS, trots att operativsystemet redan erbjuder inbyggt stöd för förgrundsnotiser. Detta ledde till att appen ständigt försökte rendera samma notis på nytt.

Som lösning infördes en plattformsberoende kontroll som skiljer mellan Android och iOS. På Android behålls den manuella förgrundsvisningen, medan appen på iOS nu enbart förlitar sig på operativsystemets egna mekanismer för att visa notiser när appen är aktiv. Vid efterföljande tester på iOS-enheter bekräftades att varje pushmeddelande endast levereras och visas en gång, och problemet med rundgången är därmed åtgärdat.

Vidare tog utvecklingen ett abrupt stopp då anropen för att prenumerera respektive avprenumerera på FCM-topics via Unity SDK inte fungerade som förväntat. Efter omfattande felsökning identifierades felet genom ett öppet ärende på GitHub[23]. Problemet visade sig ligga i Android-implementeringen av Firebase Unity SDK, medan motsvarande iOS-funktionalitet fungerade korrekt utan åtgärd. För att kringgå buggen användes en native-implementation i Java på Android, vilket korrigerade funktionaliteten för anropen.



Figur 4.6: Visualisering av bugg på iOS

5

Resultat

Detta kapitel redogör för resultatet av det genomförda projektet. Fokus ligger på vad som uppnåddes tekniskt samt hur det färdiga systemet presterade i praktiken, inklusive funktionella egenskaper, stabilitet, och återkoppling från användare. Resultatet utvärderas utifrån projektets syfte och mål som formulerades i inledningen.

5.1 Funktionell lösning

Vid projektets slut levererades en fungerande pushnotislösning integrerad i den befintliga Unity-applikationen. Lösningen möjliggör att administratörer via ett webbgränssnitt,som utvecklats av annan part, kan skicka notiser till samtliga användare genom användning av Firebase Topics. Applikationen på både Android och iOS kan ta emot notiser även när den körs i bakgrunden, förutsatt att användaren gett tillstånd.

Systemet stödjer även tilldelning av närvaropoäng via backend, där data lagras i en PostgreSQL-databas och presenteras på användarsidan. Backend fungerar som central logik för både datalagring, autentisering och kommunikation med Firebase Cloud Messaging.

5.2 Uppdaterad design

I figur 5.1 visas den uppdaterade layouten och designen för startsidan. I figur 5.2-5.3 illustreras färgvalspopupen, valet av en färg kopplar appen till motsvarande användargrupp, vilket styr vilka pushnotiser som levereras. Användare kan prenumerera på flera grupper, men endast notiser för den aktuella gruppen visas. När ett användarnamn har angetts i popupen som visas i figur 5.3, gråas inmatningsfältet ut och låses för att förhindra ytterligare namnändringar eller skapande av nya identiteter.

Användarnamnet kontrolleras mot databasen där användarnmanen är definerade som unika (UNIQUE). Om användarnamnet redan används nekas registreringen, det förhindrar att flera användare antar samma identitet. Med tanke på projektets tidshorisont och avsaknaden av en formell inloggningsfunktion ansågs detta vara det optimala sättet att säkra unika användaridentiteter. Vid avinstallation nollställs både gruppval och användarnamnsval, vilket gör det möjligt att ange nya val vid nästa installation.

Figur 5.1: Uppdaterad hemskärm

Figur 5.2: Färgvalspopup: oanvänd

Figur 5.3: Färgvalspopup: använd

5.3 Stabilitet och prestanda

Efter ett antal iterationer och omfattande testning kunde pushnotiser skickas och tas emot på ett tillförlitligt sätt. På Android fungerade notiser stabilt tidigt i projektet, medan iOS-implementationen krävde ytterligare konfiguration i Xcode. Efter justeringar i kodfilerna så fungerade även detta.

Poängsystemet visade god prestanda vid interaktion via webbsidans gränssnitt och backend-API. All data lagrades korrekt och var tillgänglig för framtida användning och analys.

5.4 Användarfeedback

Vid användartester med fritidsgårdens personal och av studenter från en paralell grupp, visade sig systemet vara lätt att använda. Fritidsgårdens personal uppskattade att få uppdateringar direkt i mobilen, och lyfte fram enkelheten i att hantera kommunikation via ett webbaserat gränssnitt.

5.5 Identifierade begränsningar

Under testfasen identifierades vissa begränsningar:

- Notiser skickades ibland dubbelt till iOS-enheter vid vissa testfall, detta löstes genom justering av notislogiken i Unity.
- Vid första installationen på vissa Android-enheter krävdes manuell aktivering av notisbehörighet.
- Unitys filstruktur medförde ibland problem vid sammanslagning, särskilt då Unity automatiskt bytte namn på resurser. Vilket försvårade den annars simpla sammanslagningen.

Trots dessa hinder kunde samtliga kritiska funktioner färdigställas enligt plan och kravställning.

5.6 Måluppfyllelse

Projektet uppfyllde i hög grad de mål som sattes upp. En kommunikationskanal som möjliggör säker envägskommunikation mellan fritidsgården och dess ungdomar kunde realiseras. Systemet uppfyller de tekniska krav och möjliggör framtida vidareutveckling såsom tvåvägskommunikation. Sammanfattningsvis visar resultaten att systemet är både tekniskt robust och användarvänligt, vilket bekräftar att projektmålen nåddes.

6

Diskussion

6.1 Diskussion kring tekniska val och arkitektur

Nedan återfinns diskussion kring tekniska val i projektet

6.1.1 Valet av Firebase Cloud Messaging

Projektet valde FCM av flera skäl. För det första är FCM en, för projektets ändamål och volymer, gratis cross-platform messaging-tjänst som officiellt stöds av Google för Android, iOS, webb och Unity, vilket möjliggjorde en sömlös integration med Unity-motorn [11].

För det andra erbjuder FCM en inbyggd topics-baserad notislösning där användare kan prenumerera på grupper eller ämnen utan krav på individuella tokens eller lagring av enhetsidentifierare, vilket förenklade hantering av GDPR-efterlevnad och minskade systemets komplexitet.

Dessutom är arkitekturen mycket skalbar och garanterar pålitlig leverans även vid hög belastning. Slutligen bidrog den väl dokumenterade SDK:n och den aktiva användarbasen till en hög utvecklingstakt och minimerade riskerna i projektets tidiga faser.

6.1.2 Tekniska val i backend

Ett viktigt designval var beslutet mellan en molnbaserad databaslösning som Firebase Firestore [24] och en traditionell relationsdatabas i PostgreSQL. Även om Firestore erbjuder enkel skalbarhet, realtidsuppdateringar och sömlös integration med Firebase-ekosystemet, föll valet till slut på PostgreSQL, eftersom Reningsborg redan driver en egen PostgreSQL-instans och kunde integrera en ny modul direkt i sin befintliga infrastruktur. Genom att containerisera databasen med Docker säkerställdes samma arbetsflöde för lokal utveckling, testning och produktion, vilket motsvarade Reningsborgs önskemål och underlättade driftsättningen på deras AWS-servrar.

Valet föll på en Node.js-baserad backend implementerad i JavaScript, och senare TypeScript, och Express, främst för att det ramverket var välbekant för teamet och tillät snabb prototypframtagning. TypeScript tillförde stark typkontroll och bättre

autokomplettering, vilket ökade utvecklingstakten och minskade risken för typrelaterade fel. Detta utöver, som tidigare nämnts, att TypeScript var ett önskemål från teamet som arbetade på webbsidan. Express, som är ett webbramverk, gjorde det enkelt att definiera REST-endpoints. Dessutom var ekosystemet runt Node.js välutvecklat och erbjöd omedelbar kompatibilitet med Firebase Admin SDK, vilket underlättade integrationen mot FCM-topics och andra Firebase-tjänster. Sammantaget gav dessa teknologival en bekväm och stabil grund för både utveckling och framtida utbyggnad.

6.1.3 Säkerhet kring API-nycklar

I projektet valdes en enkel API-nyckel-lösning för att skydda anrop mot backend endpoints, istället för en mer omfattande OAuth-lösning. Under säkerhetsmötet [22] bedömdes risken för obehörig åtkomst som låg, med hänsyn till projektets begränsade omfattning och att inga känsliga personuppgifter hanterades. Vidare fanns det inte tillräckligt med tid att spendera på den potentiella och mer omfattande OAuth-lösningen. Samtidigt finns det risk att API-nycklar som ligger inbäddade i klientappar kan avslöjas via dekompilering av applikationen eller liknande metoder[25]. Trots detta bedömdes metoden vara acceptabel och förenad med låg risk, inte minst med tanke på appens begränsade användarbas och de få funktioner som exponeras. För framtida versioner, i takt med att applikationen växer och behovet av mer detaljerade behörighetskontroller ökar, kan det vara lämpligt att övergå till ett OAuth-baserat system för att höja säkerhetsnivån.

6.2 Hållbarhetsaspekter

Följande avsnitt behandlar hållbarhetsaspekter relaterade till projektet, med fokus på samhälleliga, etiska och ekologiska perspektiv.

6.2.1 Samhälleliga aspekter

Appen bidrar till att stärka fritidsgårdarnas roll som trygga mötesplatser för barn och ungdomar genom att möjliggöra snabb, säker och effektiv kommunikation. Genom att erbjuda en digital kanal för information och notiser kan fritidsgårdspersonal snabbt nå ut med viktiga meddelanden, vilket skapar en mer strukturerad och proaktiv verksamhet. Detta förbättrar inte bara den interna organisationen utan stärker även den sociala sammanhållningen i lokalsamhället, då ungdomar får möjlighet att delta i aktiviteter och evenemang som främjar gemenskap och delaktighet.

En ytterligare fördel är att appen, genom att begränsa möjligheten för barn att chatta med varandra, potentiellt kan bidra till minskad mobbning, eller åtminstone att detta inte sker på appen. Vidare innebär lösningen att informationen blir lättillgänglig för både ungdomar och deras vårdnadshavare, vilket ökar transparensen och tryggheten. Sammantaget moderniserar och effektiviserar appen fritidsgårdarnas kommunikationsstrategier och tillför ett tydligt mervärde för både verksamheten och samhället i stort.

6.2.2 Etiska aspekter

Projektets etiska aspekter är av stor vikt, särskilt då det involverar hantering av barns och ungdomars personuppgifter. Applikationen utformas på så sätt att barnen och ungdomarna inte kommer att kunna kommunicera med varandra utan endast ta del av information från fritidsgården. Denna designprincip bidrar till att minimera risken för mobbning och säkerställer att kommunikationen enbart sker mellan fritidsgårdspersonalen och de unga användarna, samt eventuellt deras vårdnadshavare. Vidare följs alla relevanta regelverk såsom GDPR, för att garantera en säker och ansvarsfull hantering av personuppgifter, med hänsyn till relevanta avgränsningar för projektet. Genom att integrera dessa etiska riktlinjer skapas en trygg och kontrollerad digital miljö där användarnas integritet och välbefinnande alltid prioriteras.

6.2.3 Ekologiska aspekter

Appen förväntas under nuvarande förutsättningar ha ett litet antal aktiva användare. Detta medför att användningen av molnbaserade tjänster och servrar inte blir särskilt stor, vilket resulterar i en minimal miljöpåverkan. Eftersom vi enbart behandlar mjukvara, en app som laddas ner av användarna, och med vetskapen om att majoriteten redan äger en smartphone, är den totala ekologiska belastningen låg.

7

Slutsats

I detta examensarbete har ett pushnotissystem utvecklats och framgångsrikt integrerats i en befintlig Unity-baserad mobilapplikation för Android och iOS. Systemet uppfyller de ursprungliga kraven på säker, envägskommunikation från fritidsgårdspersonal till ungdomar och deras vårdnadshavare, och har verifierats genom tester på emulatorer samt fysiska enheter med stabila resultat. Slutsatsen bekräftar att den utvecklade lösningen effektivt möter det identifierade behovet av snabb och säker kommunikation, vilket var det ursprungliga problemet i inledningen.

Tre centrala bidrag kan identifieras:

- **Topics-baserad distribution:** Genom att ersätta enhetstokens med FCM-topics har risken för personuppgiftsincidenter minimerats. Lösningen minimerar hanteringen av personuppgifter och minskar risken för oavsiktliga dataläckor genom att undvika lagring av individuella enhetsidentifierare i backend.
- **Modulär och skalbar arkitektur:** En Node.js och TypeScript-baserad backend med Express, Dockeriserad PostgreSQL och Firebase Admin SDK ger en flexibel plattform för meddelandehantering och poängsystem. Arkitekturen möjliggör enkel drift i molnmiljö och framtida kapacitetsuppskalning.
- **Notishistorik för ökad tillgänglighet:** Historikfunktionen i appen, som visar de senaste 30 dagarnas notiser i omvänd kronologisk ordning, ökar transparensen och säkerställer att viktig information alltid finns tillgänglig, även om realtidsnotiser av misstag raderas.

Efter att personalen på fritidsgården använt systemet upplevdes funktionerna som både användarvänliga och pålitliga.

Rekommendationer för fortsatt utveckling

- Införa filtrerings- och sökfunktioner i notishistoriken för att förbättra användbarheten vid stora mängder meddelanden.
- Utreda möjligheterna till end-to-end-kryptering av pushinnehåll för att ytterligare stärka sekretessen.
- Utforska begränsad tvåvägskommunikation, exempelvis genom realtidsdatabaser eller WebSocket-lösningar, för att stödja bekräftelser och interaktiva åtgärder.

- Utforska möjligheter för att åstadkomma en mer tillgänglig och användarvänlig app, exempelvis för människor med färgblindhet.

Med denna grund har systemet potential att skala upp mot fler användargrupper och utökade funktionaliteter, samtidigt som kraven på datasäkerhet och användarvänlighet fortsatt kan tillgodoses.

Litteratur

[1] Internetstiftelsen, ”Barnen och internet,” Internetstiftelsen, Kapitel 10 i Svenskarna och internet, 2023, https://svenskarnaochinternet.se/app/uploads/2023/10/svenskarna-och-internet-2023-kap-10-barnen-och-internet.pdf, Hämtad 2025-04-23.

[2] Tjörns kommun, *Appen Ung på Tjörn*, https://www.tjorn.se/kultur-fritid-och-turism/funkis-fritid/appen-ung-pa-tjorn, Hämtad 2025-04-23, 2025.

[3] GitHub, Inc. ”About large files on GitHub,” GitHub. (2025), URL: https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github#repository-size-limits (hämtad 2025-04-20).

[4] Apple Inc. ”TestFlight Beta Testing and App Store Connect,” Apple Developer. (2025), URL: https://developer.apple.com/testflight/ (hämtad 2025-05-16).

[5] Microsoft Corporation. ”C# Guide,” .NET Documentation. (2025), URL: https://learn.microsoft.com/dotnet/csharp/ (hämtad 2025-04-29).

[6] JetBrains s.r.o. ”Rider Documentation,” JetBrains. (2025), URL: https://www.jetbrains.com/help/rider/ (hämtad 2025-04-29).

[7] Apple Inc. ”Preparing your app for distribution,” Apple Developer. (2025), URL: https://developer.apple.com/documentation/xcode/preparing-your-app-for-distribution (hämtad 2025-04-30).

[8] Adobe Inc. ”Adobe Photoshop User Guide,” Adobe. (2025), URL: https://helpx.adobe.com/photoshop/user-guide.html (hämtad 2025-05-16).

[9] Unity Technologies. ”Unity Version Control – Unity DevOps Documentation,” Unity Technologies. (2025), URL: https://docs.unity.com/ugs/en-us/manual/devops/manual/unity-version-control (hämtad 2025-06-11).

[10] Postman, Inc. ”Postman Learning Center,” Postman. (2025), URL: https://learning.postman.com/docs/ (hämtad 2025-05-12).

[11] Google. ”Firebase Cloud Messaging,” Firebase. (2025), URL: https://firebase.google.com/docs/cloud-messaging (hämtad 2025-05-02).

[12] Docker, Inc. ”Get Started with Docker,” Docker Documentation. (2025), URL: https://docs.docker.com/get-started/ (hämtad 2025-05-12).

[13] Mozilla Foundation. ”JavaScript - MDN Web Docs,” MDN Web Docs. (2025), URL: https://developer.mozilla.org/docs/Web/JavaScript (hämtad 2025-05-12).

[14] Microsoft Corporation. ”TypeScript Documentation,” TypeScript. (2025), URL: https://www.typescriptlang.org/docs/ (hämtad 2025-05-12).
[15] G. Bierman, M. Abadi och M. Torgersen, ”Understanding TypeScript,” i *ECOOP 2014 – Object-Oriented Programming*, R. Jones, utg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, s. 257–281, ISBN: 978-3-662-44202-9.
[16] OpenJS Foundation. ”Express - Fast, unopinionated, minimalist web framework for Node.js,” Express. (2025), URL: https://expressjs.com/ (hämtad 2025-05-12).
[17] PostgreSQL Global Development Group. ”PostgreSQL Documentation,” PostgreSQL. (2025), URL: https://www.postgresql.org/docs/current/ (hämtad 2025-05-12).
[18] Meta Platforms, Inc. ”React - A JavaScript library for building user interfaces,” React. (2025), URL: https://reactjs.org/docs/getting-started.html (hämtad 2025-05-13).
[19] Google. ”Set up a Firebase Cloud Messaging client app with Unity,” Firebase. (2025), URL: https://firebase.google.com/docs/cloud-messaging/unity/client (hämtad 2025-05-12).
[20] Android Developers. ”Create and Manage Notifications,” Android Developers. (2025), URL: https://developer.android.com/training/notify-user/build-notification (hämtad 2025-05-13).
[21] EUR-Lex, *Regulation (EU) 2016/679 of the European Parliament and of the Council*, Official Journal of the European Union, Articles 4(5) och 5(1)(c) om pseudonymisering och dataminimering, 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj (hämtad 2025-05-25).
[22] P. Hanssen, *Säkerhetsmöte om projektets skyddsåtgärder*, Personligt möte hos Omegapoint, Interna mötesanteckningar, 1 april 2025.
[23] maniaks1, *“Android Firebase 8.0.0 TokenReceived called only on application reinstall”*, https://github.com/firebase/quickstart-unity/issues/1088, Hämtad 2025-05-19, 2021.
[24] Google, *Firestore Documentation*, https://firebase.google.com/docs/firestore, Hämtad 2025-04-27, 2025.
[25] L. Wei, H. Huang, S.-C. Cheung och K. Li, *How Far are App Secrets from Being Stolen? A Case Study on Android*, 2025. arXiv: 2501.07805 [cs.CR]. URL: https://arxiv.org/abs/2501.07805.

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

CHALMERS