Webbportal för föreningsdriven fritidsgård

Utveckling av en webbportal med administrativt gränssnitt för integration med pedagogiska spel

Examensarbete inom högskoleprogrammet Datateknik

Jacob Eklund Bergfalk

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK

CHALMERS TEKNISKA HÖGSKOLA

Göteborg 2025

www.chalmers.se

Examensarbete 2025

Webbportal för föreningsdriven fritidsgård

Utveckling av en webbportal med administrativt gränssnitt för integration med pedagogiska spel

Jacob Eklund Bergfalk

CHALMERS

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborg 2025

Webbportal för föreningsdriven fritidsgård
Utveckling av en webbportal med administrativt gränssnitt för integration med pedagogiska spel
Jacob Bergfalk

Handledare: Hanna Ek, Data- och Informationsteknik
Examinator: Nicholas Smallbone, Data- och Informationsteknik

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

Skriven i LaTeX
Göteborg 2025

Webbportal för föreningsdriven fritidsgård.
Utveckling av en webbportal med administrativt gränssnitt för integration med pedagogiska spel
Jacob Eklund Bergfalk
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola

Sammanfattning

Reningsborg är en organisation som, utöver sina Second Hand-butiker, driver en fritidsgård och andra sociala verksamheter med fokus på barn och ungdomar. I syfte att digitalisera fritidsgårdsverksamheten efterfrågade Reningsborg en lösning som består av ett pedagogiskt spel samt en webbportal. Projektet kommer fokusera på designen och utvecklingen av portalen, vart syfte är att underlätta hanteringen av fritidsgårdens deltagande, poängutdelning samt utskickning av push notifikationer, vilket utgör ett centralt verktyg för fritidsledarnas arbete. Genom ett rollbaserat behörighetssystem har portalen byggts med fokus på både säkerhet och användarvänlighet, där åtkomsten till olika funktioner styrs av fritidsledarens behörighetsnivå. Utvecklingen har skett med hjälp av moderna webbutvecklingsteknologier som React, Typescript, Express och Firebase. Projektet har följt en iterativ arbetsmetod, där funktionaliteten kontinuerligt har utvärderats och justerats efter nya behov. Projektets huvudsakliga mål har varit att skapa en skalbart, användarvänligt och flexibelt system, anpassat för fortsatt utveckling.

Nyckelord: Digitalisering, Rollbaserat behörighetssystem, Datavisualisering, Användarvänlighet, Skalbarhet, Webbutveckling, React, nodejs, Firebase.

Förord

Jag vill speciellt tacka min handledare Jenny Forsberg på OmegaPoint och även Elisabeth Valinder hos Reningsborg. Dessutom vill jag tacka min handledare, Hanna Ek på Chalmers Tekniska Högskola.

Jacob Eklund Bergfalk, Göteborg, Juni 2025

Innehåll

Figurer

1

Inledning

I flera socioekonomiskt utsatta områden i Göteborg, såsom Tynnered, ligger gymnasiebehörigheten bland grundskoleelever på en lägre nivå än rikssnittet. Den låga behörigheten kan i sin tur påverka ungas framtidsutsikter negativt och öka risken för socialt utanförskap [1]. För att skapa en säker miljö och motverka detta arbetar fritidsverksamheter i området aktivt med att skapa trygga och meningsfulla aktiviteter och miljöer för barn och unga. Ett av sätten är att uppmuntra barnen att tillbringa mer tid på fritidsgårdar, där fritidsledare kan inspirera till en mer positiv inställning till skolan och hjälpa dem bygga en tryggare och mer stödjande social krets.

Pedagogiska spel har blivit en alltmer populär metod för att uppmuntra lärande och social interaktion hos barn och ungdomar [2]. Genom att kombinera ett interaktivt spel med ett belöningssystem kan spel skapa motivation och engagemang. I detta sammanhang har en mobilapplikation utvecklats för att uppmuntra barn att besöka fritidsgården genom att belöna barnen för deras närvaro, vilket senare kan användas för att göra roliga aktiviteter. Syftet är att öka engagemanget och göra fritidsgården en attraktiv säker miljö där barn kan spendera sin fritid, och minska chansen att barn attraheras till kriminella kretsar.

1.1 Reningsborg

Reningsborg är en organisation som arbetar för att skapa möjligheter för människor i utsatta situationer genom sociala insatser, utbildning och arbetsintegration [3]. Genom sitt arbete vill Reningsborg bidra till ett mer inkluderande samhälle där fler får möjlighet att utvecklas och förbättra sina livsvillkor. En central del av Reningsborgs verksamhet är deras second hand-butiker, vars överskott går till sociala projekt både i Sverige och internationellt.

En viktig del i Reningsborgs arbete är även deras fritidsverksamhet, där de skapar en trygg och stimulerande miljö där barn och ungdomar kan vistas i efter skoltid och på lov. Fritidsgården erbjuder aktiviteter som syftar till att stärka ungdomarnas sociala samhörighet och motivation för studier. Genom att skapa en plats där unga kan umgås, utvecklas och få stöd, arbetar Reningsborg aktivt för att minska risken för utanförskap och kriminalitet.

1.2 Syfte

Som en del av Reningsborgs arbete har de utvecklat en mobilapplikation för att uppmuntra fler barn och ungdomar att delta i fritidsaktiviteter. Genom ett belöningssystem, där barn och ungdomar samlar poäng för deras närvaro, sitt bemötande och sitt engagemang, vill de öka motivationen, stärka gemenskapen och bidra till att de vistas på fritidsgården.

Tidigare hanterades deltagande och poängen för närvaro, förhållningsregler och kompisinsatser manuellt via Exceldokument, vilket har varit tidskrävande och svårt att samordna. Det saknades även en tydlig kommunikationsstruktur mellan fritidsledarna och deltagarna.

Syftet med examensarbetet är därför att utveckla en webbportal som fungerar som ett administrativt gränssnitt för Reningsborg. Portalen ska underlätta hanteringen av appens användare, grupper och poängutdelning. Dessutom ska portalen effektivisera kommunikationen från fritidsledare genom att göra det möjligt att skicka riktade notifikationer till olika användargrupper på ett smidigt och säkert sätt.

1.3 Mål

Detta examensarbete fokuserar på att utveckla en webbportal som fungerar som en administrativ plattform för applikationen. Webbportalen ska användas av fritidsledare för att hantera användare, dela ut poäng, samt skicka riktade push notifikationer från gränssnittet till mobilapplikationen. Gränssnittet ska ge en tydlig överblick över användardata och erbjuda ett enkelt och intuitivt sätt att skicka notifikationer samt att se statistik över användarnas deltagande.

Projektet syftar till att utveckla webbportalen med fokus på användarvänlighet, flexibilitet och skalbarhet. Dessutom är det centralt att spara en minimal mängd data och minimera insamlingen av personuppgifter. Systemet ska vara säkert och syftar till att stödja inloggning, användarhantering och säkerhet.

1.4 Avgränsningar

General Data Protection Regulation (GDPR) är en lagstiftning inom EU som syftar till att skydda individens personuppgifter och ge individen ökad kontroll över hur datan hanteras. När det gäller digitala tjänster finns det särskilda regler för att säkerställa att integriteten skyddas och att det samlas in en minimal insamling av information. Barn och ungdomar är en särskilt skyddad grupp, vilket betyder att det finns strikta krav på samtycke och datalagring [4].

I detta projekt är GDPR centralt eftersom gränssnittet och mobilapplikationen är utvecklad för barn och ungdomar. För att uppfylla GDPR kommer applikationen och webbportalen begränsa insamlingen av data. För att inte samla data delas barnen in i färgkodade grupper, som motsvarar en åldersgrupp. Autentisering och notifikationshantering sker via Firebase, vilket följer etablerade säkerhetsstandarder och skyddar användarens information från obehörig åtkomst.

Examensarbetet fokuserar endast på utvecklingen av webbportalen och omfattar utveckling av och koppling mellan användargränssnittet, backend och del av databasen. All utveckling sker inom ramen för webbportalen och innefattar inte integrering eller modifiering av mobilapplikationen.

Dessutom innefattas projektet inte av utveckling av notifikationskommunikationen, endast gränssnittet som notifikationerna skickas från.

Ytterligare kommer hemsidan inte anpassas för mobilt bruk utan kommer endast utvecklas för dator. Medan detta reflekterar deras nuvarande lösning som sker via Exceldokument, är tidsbegränsning anledningen till varför sidan inte är anpassad för mobilt bruk.

2

Metod

I detta kapitel redogörs arbetsmetoden, överväganden, teknikval och de viktigaste verktygen som har används under projektets gång.

2.1 Arbetsmetod

Projektet har genomförts med en iterativ utvecklingsmetod. Detta möjliggjorde en flexibel process där både gränssnittets design och dess funktionalitet kunde justeras löpande under arbetet. Detta var idealt eftersom portalens mål justerades under projektets gång, vilket betydde att portalen kunde förbättras utifrån feedback och nya önskemål. Genom att arbeta iterativt, är det möjligt att säkerställa att slutprodukten uppfyllde både funktionella och säkerhetsmässiga krav [5].

Projektet utvecklas jämsides med andra examensarbeten som hanterar mobilapplikationen och notifikationsfunktionaliteten. I den gemensamma utvecklingsgruppen används en agil arbetsmetod, vilket innebär att projektets omfattning kunde justeras under projektets gång, vilket var fördelaktigt eftersom mål och struktur ändrades i samband med att kraven förtydligades.

2.2 Prototypdesign

För prototypdesign och gränssnittsplanering användes Figma, vilket är ett program för grafisk redigering och prototypframställning [6]. Genom att skapa en prototyp på gränssnittet är det enklare att få en bättre förståelse för vilka funktioner som behöver utvecklas samt få ett konkret mål att sträva efter.

2.3 Utvecklingsmiljö

Visual Studio Code valdes som huvudsaklig utvecklingsmiljö för projektet på grund av dess inbyggda stöd för Javascript och Node.js, dess stöd för modern webbprogrammering, dess ekosystem av funktionalitetstillägg samt dess integration med GitHub för versionshantering.

Som tilläggsverktyg användes Prettier för automatiserad omformatera kod och Live Server vilket gör det möjligt att se en förhandsgranskning av gränssnittet i projektets tidigare faser.

För att hantera och starta projektets utvecklingsmiljö användes Docker Compose, vilket gjorde det enkelt att hantera projektets komponenter, beroenden och bibliotek. Compose gjorde det möjligt att garantera att slutresultatet beter sig likadant lokalt som det gör på en server.

2.4 Säkerhetsstrategi

För att säkerställa att portalen uppnår projektets säkerhetsmål används en säkerhetsfilosofi för utvecklingen. På grund av brist på personlig kompetens inom säkerhet, kommer en öppen filosofi användas. Det betyder att brister hellre upptäcks och visas upp än att systemet förlitar sig på att portalen blockerar intrång. Däremot kommer säkerhetsprinciper fortfarande utvecklas.

3

Teknisk Bakgrund

Detta kapitel kommer behandla och beskriva de tekniska begrepp och koncept som förekommer i rapporten.

3.1 Hypertext Markup Language

Hypertetxt Markup Language (HTML) är det standardiserade märkspråket som används för att strukturera innehåll på webbsidor [7]. HTML definierar rubriker, stycken, bilder och andra statiska element på en hemsida, och definierar hemsidans grundläggande uppbyggnad, snarare än dess utseende och beteende [8]. I moderna webbapplikationer används HTML ofta i kombination med andra tekniker såsom CSS och JavaScript, för utseende och funktionalitet respektive.

3.2 Cascading Style Sheets

Cascading Style Sheets (CSS) är det standardiserade stilspråket inom webbutveckling och definierar utseendet på ett HTML dokument. CSS gör det möjligt för utvecklare att styra färger, typsnitt, positionering och responsivitet på hemsidans komponenter [9].

CSS tillämpar en cascading princip, vilket betyder att flera stilar och regler kan påverka en komponent. I Moderna webbapplikationer finns CSS i olika former, som CSS moduler och ramverk för att hantera utseendet. Exempel på detta kan vara Tailwind eller Bootstrap. Dessa ramverk har egna definitioner och sätt att använda styling, men i grunden utnyttjar dessa CSS [10].

3.3 JavaScript

JavaScript är ett dynamiskt skriptspråk eller programmeringsspråk som tillåter komplex funktionalitet på en hemsida. JavaScript körs direkt i webbläsaren och är en grundpelare inom modern webbutveckling. JavaScript används exempelvis för att ändra i Document Object Model (DOM), hantera händelser och kommunicera med servrar [11].

Språket är händelsedrivet och dess komponenter reagerar på användarens handlingar i HTML dokumentet, såsom klick, tangenttryck eller inmatningar. På grund av detta är det möjligt att ändra hemsidans innehåll i realtid.

3.4 TypeScript

TypeScript är ett programmeringsspråk utvecklat av Microsoft som bygger på JavaScript men introducerar *static typing* [12]. Detta betyder att deklarerade variabler är fördefinierade och kontrolleras under kompilering, vilket leder till bättre kodkvalitet, ökad förståelse, och mindre buggar. Till skillnad från JavaScript, om en variabel deklareras, kan den inte ändras till en annan senare i programmet. Under kompilering omvandlas TypeScript till standardiserad JavaScript, som stöds av majoriteten av webbläsare. Detta betyder även att ramverk och bibliotek som är byggda för JavaScript även fungerar för TypeScript [12].

3.5 React

React är ett populärt JavaScript-bibliotek utvecklat av Meta för att bygga dynamiska och interaktiva webbgränssnitt [13]. Biblioteket baseras på en komponentbaserad arkitektur, vilket gör det enkelt att återanvända kod och hantera gränssnitt på ett resurseffektivt sätt. React är ett kraftfullt verktyg för att bygga moderna webbapplikationer. React använder en virtuell DOM som optimerar uppdateringar och förbättrar prestandan genom att minska mängden omladdningar av webbsidan. React har ett stort ekosystem och stöd för tredjepartsbibliotek vilket gör att React är ett kraftfullt verktyg för att bygga moderna webbapplikationer [13].

Vite är ett byggverktyg som optimerar moderna ramverk som React, vilket möjliggör snabbare uppstarter och omkompileringar vid kodändringar. Vite används ofta i kombination med ramverk som React för att skapa snabba och responsiva webbapplikationer.

3.6 Node.js

Node.js är ett plattformsoberoende JavaScript-runtimemiljö. Till skillnad från traditionell JavaScript som endast körs via en webbläsare, möjliggör Node.js att exekvera JavaScript utan ett webbgränssnitt [14]. Detta gör det möjligt att skapa fristående applikationer i JavaScript. Node.js används främst för backend programmering, HTTP och databaskommunikation.

Node.js bygger på öppen källkod och har ett stort ekosystem av bibliotek som styrs via pakethanteraren Node Package Manager (NPM), vilket underlättar installation och hantering av biblioteken [15].

Ett vanligt bibliotek är Express.js. Biblioteket bygger ovanpå Node.js för att strukturera backendlogik, förenkla routing och hantering av API-endpoints [16].

3.7 PostgreSQL

Structured Query Language (SQL) är ett standardiserat språk som används för att hantera och manipulera data i relationsdatabaser. Med SQL kan man skapa tabeller, infoga data, hämta och radera information i en databas och fungerar som grunden i hur kommunikationen sker i en relationsdatabas [17].

PostgreSQL är en relationsdatabas med öppen källkod och är särskilt känt för sin flexibilitet och integritet vilket gör att det används för projekt i varierande storlek. PostgreSQL bygger på SQL, vilket betyder att det använder sig av SQL för att hantera databaserna [18].

3.8 Docker

Docker är en plattform som används för att paketera, bygga och köra webbapplikationer och dess beroenden i isolerade miljöer som kallas för containers. En container innehåller allt som en applikation behöver för att fungera, vilket gör det enklare att hantera beroenden, bibliotek eller konfiguration oavsett om applikationen körs lokalt eller om den är lagrad på en server. Det gör det också enkelt att flytta applikationen mellan olika miljöer utan att behöva konfigurera och ladda ner beroenden, bibliotek och databaser igen [19].

En container är en lättviktig, fristående och säker körmiljö, där en applikation kan köras isolerat utan att påverka resterande system. Detta minimerar risken mellan olika projekt och förenkla både installation och felsökning.

För att hantera flera containrar samtidigt används verktyget Docker Compose, vilket är ett verktyg som gör det möjligt att definiera en komplett applikation. Compose består av en konfigurationsfil `docker-compose.yml` som definierar varje del av systemet, såsom användargränssnittet, backend och databasen, där varje del representeras av en egen container [19].

3.9 Firebase

Firebase är en central del av projektet, då det erbjuder en kostnadsfri och flexibel lösning för att hantera notifikationer, autentisering och användarhantering. Som en molnbaserad plattform utvecklad av Google, erbjuder Firebase en rad tjänster för både webb- och mobilapplikationer. I detta projekt används framför allt två tjänster: **Firebase Cloud Messaging (FCM)** och **Firebase Authentication**.

Firebase Cloud Messaging används för att skicka notifikationer från webbportalen till mobilapplikationen. Med FCM kan administratören skicka riktade meddelanden till specifika grupper av användare [20].

Firebase Authentication används för att hantera administratörernas inloggning och åtkomstkontroll [21]. Authentication erbjuder inbyggd funktionalitet för autentisering via E-post och lösenord och möjliggör rollbaserad åtkomst genom så kallad `customClaims`. För att hantera detta på ett säkert sätt används två olika SDK:er från Firebase:

- **Firebase Admin SDK** används i backend och innehåller känsliga funktioner för kontohantering, såsom att skapa och radera användare samt tilldela eller justera roller. Eftersom denna SDK har tillgång till känsliga funktioner, kräver den en högre åtkomstnivå och kan endast användas i backend [22].
- **Firebase Client SDK** används för inloggning och autentisering av användare. Denna SDK har inte tillgång till många känsliga funktioner, vilket gör att det går att använda i användargränssnittet för att göra enklare funktioner [23].

Genom att kombinera dessa verktyg är det möjligt att begränsa användargränssnittet från att få tillgång till känslig information i onödan, medan fortfarande vara möjligt att nyttja säkerheten som kommer med en säker backend. Plattformens molnbaserade lösning gör systemet säkert och skalbart.

4

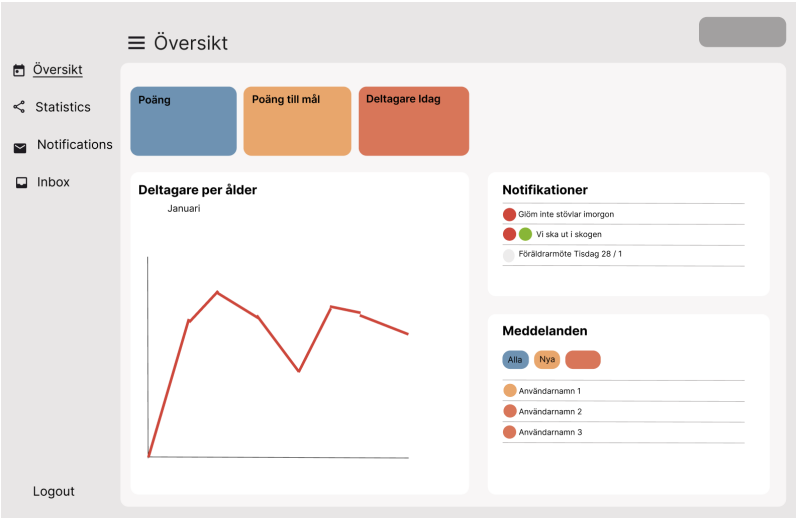
Genomförande

Följande kapitel beskriver utförandet av arbetet, vilket inkluderar designen av hemsidan, dess funktionalitet samt tekniska beslut under projektets gång.

4.1 Design och planering

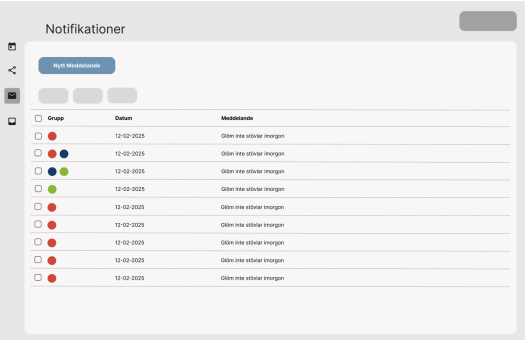
Projektets inledningsfas bestod av att ta fram en visuell prototyp av portalen. Designen utgick från identifierade behov för framtidsbehov och deras tidigare lösning samt från diskussioner med Reningsborg fritidsledare. Prototypen fungerade som ett konkret mål att arbeta mot under utvecklingen. Genom att definiera strukturen visuellt kunde flera designbeslut tas, vilket underlättade under utvecklingsprocessen.

Prototypen hämtade inspiration från andra administrativa gränssnitt, som ofta delade gemensamma visuella designelement som bidrar till tydlig struktur och användarvänlighet. Eftersom ett centralt mål med portalen var att skapa ett användarvänligt och skalbart system, valdes flera av designelementen att återanvändas och anpassas för portalen. Ett tydligt exempel var designen på sidan och valdes att skapa en översiktssida, som fungerar som en landningssida och presenterar aktuell information och statistik för administratören (se figur 4.1).

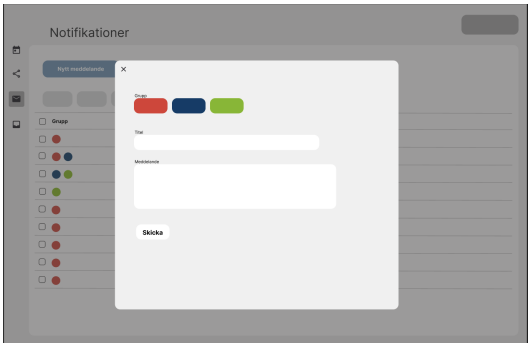


Figur 4.1: Visuell prototyp för översiktssidan

Strukturen för de funktionella sidorna i portalen, såsom användarhantering och notifikationerna, inspirerades från andra digitala verktyg såsom e-post webbgränssnitt. Eftersom en av portalens huvudfunktioner bygger på data i listformat, exempelvis användarna eller notifikationerna, var den etablerade strukturen (se figur 4.2 (**a**)) och hanteringen av element i listor en viktig inspiration.



(a) Prototyp på notifikationshanteringen



(b) Prototyp för utskick av notifikation

Figur 4.2: Prototypbilder som definierade designen av projektet.

Även fast en färdig prototyp för användarhanteringen inte var färdigställd, fanns det tidigt ett önskemål att göra det enkelt att tilldela poäng. Denna sida var planerad att följa en liknande struktur som notifikationerna, med en lista på alla användare och funktionalitet för poänghantering.

För att hantera individuella element i listorna, valdes en modell där varje element kan hanteras individuellt. Detta möjliggör en mer detaljerad kontroll, vilket är särskilt relevant för användarlistan, där fritidsledaren ska kunna tilldela flera olika typer av poäng som baseras på individuella prestationer.

4.2 Implementering av användargränssnitt

Användargränssnittet är byggt med React i kombination med TypeScript och Vite. Dessa verktyg valdes för att möta projektets krav på säkerhet, användarvänlighet och skalbarhet. Reacts komponentbaserade utvecklingssätt möjliggör en modulär utvecklingsmetod där varje komponent ansvarar för en del av funktionaliteten, vilket bidrar till bättre underhåll och vidareutveckling.

Utvecklingen inleddes genom att bygga upp den initiala strukturen och identifiera sidoindelningen från de funktionella behov som framkommit från prototypen. De huvudsakliga sidorna var:

- **Överblick** som fungerar som en landningssida för administratören.
- **Statistik** vilket visar aktuell statistik om fritidsgården och dess deltagande.
- **Notifikationer** som hanterar att skicka ut notifikationer och se en överblick på tidigare notifikationer

- **Användarhantering** för att dela ut poäng och få en överblick på alla användare som använder applikationen.

Dessutom kunde en ytterligare sida identifieras och implementeras under projektets gång, vilket var användarhantering för administratörerna. Sidan framkom från behovet att visa transparens kring vilka personer som har åtkomst till portalen och respektive rättigheter.

För att minska beroenden och göra sidan mer återanvändbar utvecklades även generiska komponenter vid sidan av. Dessa identifierades när liknande kod kunde användas på flera ställen. Exempel på komponenter som bröts ut till egna komponenter är `Pagination`, `DropDown-meny` och `dialogrutor`.

Ett medvetet designbeslut var att separera logiken från gränssnittet. För funktionerna som hanterade API-anrop skrevs logiken i separata filer och moduler som sedan importerades till respektive page. Detta beslut togs eftersom portalen ska uppnå en fokuserad struktur och bidra till skalbarhet och flexibilitet.

4.2.1 Autentisering med AuthContext

För att hantera autentisering i webbportalen implementerades en central komponent, `AuthContext`. Denna komponent fungerar som ett övergripande kontextlager som omsluter hela användargränssnittet och gör autentiseringsrelaterad information tillgänglig för alla delar av portalen. `AuthContext` utvecklades tidigt i projektet eftersom flera delar av portalen behövde tillgång inloggningsstatus, användarinformation och behörighetsnivåer.

Firebase Authentication valdes som lösning för inloggningshantering. För att bidra till en snabb och säker inloggning valdes användningen av en lösning där autentisering sker både i användargränssnittet och i backend. Användargränssnittet ansvarar för att initiera inloggningen genom Firebase Client SDK, som returnerar en temporär identifikations token. Denna token används sedan vid varje förfrågan till backend, där Firebase Admin SDK verifierar och tolkar identifikations token. Genom denna struktur kan användarinformation som e-post, användarnamn och behörighetsnivå göras tillgänglig i portalen via `AuthContext`.

4.2.2 Utveckling av egna komponenter

I samband med att portalen växte i mål och funktionalitet identifierades ett tydligt behov av återanvändbara komponenter. Det fanns ett särskilt behov för det i notifikations- och användarfliken, eftersom dessa har flera visuella och logiska likheter, vilket gjorde det naturligt att utforma generiska komponenter som förenklar utvecklingen.

React är ett komponentbaserat ramverk som uppmuntrar modulär och återanvändbar kod, vilket gör att det finns en stor mängd färdiga komponenter tillgängliga online. Under projektet övervägdes användningen av externa komponenter, däremot togs beslutet att utveckla majoriteten själv istället, delvis för att undvika framtida licensproblem men också för att få större kontroll över komponenterna. Det går att läsa mer om licenser och motiveringen i 6.5.

4.3 Backend

Utvecklingen av Backend inleddes efter att första versionen av användargränssnittet var utvecklad. Eftersom webbportalen hanterar känslig information såsom användaruppgifter och riktade notifikationer till mobilapplikationens användare, låg ett stort fokus på att skapa en flexibel, skalbar men främst säker lösning.

Systemet byggdes med Node.js med Typescript och Express, där arkitekturen delades in i tre lager: Router, Controller och Middleware. Denna uppläggning valdes för att separera ansvar och underlätta för framtida vidareutveckling och underhåll. Routern ansvarar för att definiera de olika API-endpoints som användargränssnittet kommunicerar med, exempelvis för att hämta användardata, skicka notifikationer eller uppdatera poäng. Dessa endpoints är kopplade till en tillhörande Controllerfunktion, som innehåller logiken för att bearbeta inkommande förfrågningar och data samt hantera kommunikation med Firebase-tjänsterna och databasen.

Mellan dessa lager finns Middleware som ansvarar för att hantera autentisering och behörighetskontroll. Middleware innehåller funktioner som körs innan en förfrågan når sin Controllerfunktion, varav de främsta funktionerna är `verifyToken` och varianter av `verifyRole`.

4.3.1 Tokenverifiering och rollhantering

När en förfrågan skickas från användargränssnittet till backend skickas även en temporär token. För att inte ge ut känslig information måste denna token verifieras mot Firebase Authentication för att se om användaren är behörig. Där används metoden `verifyToken`, vars uppgift är att autentisera förfrågningarna.

En token som kommer från användargränssnittet är en känslig bit av information som innehåller tillräckligt med information för att se vilken användare som har gjort anropet. På grund av att det är känslig information är denna token endast giltig under ett begränsat tidsintervall, i detta fallet är det en timme. `verifyToken` roll är att verifiera denna token med en inbyggd verifieringsmetod som inkluderas i Firebase Admin SDK [24]. Om token anses vara giltig så fortsätter förfrågan till sin controllerfunktion, annars returneras ett autentiseringsfel och administratören loggas ut.

```
// StaffHandler
routes.get("/fetchUsers", verifyToken, requirePartowner, fetchUsers);
routes.post("/updateRole", verifyToken, requirePartowner, updateRole);
routes.post("/createNewStaff", verifyToken, requirePartowner, createNewStaff);
routes.post("/deleteStaff", verifyToken, requireOwner, deleteStaff);
routes.post("/transferOwnership", verifyToken, requireOwner, transferOwnership);

// Group colors
routes.get("/getGroupAndColors", verifyToken, getGroupAndColors);
routes.get("/getGroupDefinitions", verifyToken, getGroupDefinitions);
routes.post("/updateGroups", verifyToken, updateGroups);
```

Figur 4.3: Exempel på hur en förfrågan hanteras i routern. Först anges endpointen (t.ex `"/deleteStaff"`), följt av middleware (t.ex `verifyToken` och `requireOwner`). Sist anges controllerfunktionen (t.ex `deleteStaff`), som innehåller logiken för att hantera förfrågan.

`verifyToken` ansvar är alltså att autentisera om en användare är registrerad, men inte vilka behörigheter den har. Av den anledning har det utvecklats ett rollsystem som nyttjar `customClaims`. Rollsystemet är ett hierarkiskt system som begränsar åtkomsten till känsliga funktioner som att radera användare av administratörer som saknar tillräckliga behörigheter. För att bekräfta att en användare har tillräckliga rollbehörigheter finns funktionerna `verifyRole`, som kräver specifik rollbehörighet som kan ses i figur 4.3.

Funktionerna är alltså ett säkerhetslager, vilket betyder att ingen annan del av programmet behöver hantera behörighetskontroller. Däremot har rollsystemet även ett ytterligare syfte som gör systemet säkrare, vilket förklaras i mer detalj i 6.2.1.

4.3.2 VerifyToken och skillnaden mot VerifySession

I detta projekt används token-baserad hantering, men det finns även en mer traditionell autentiseringsmetod som kallas för Session-hantering, vilket betyder att användargränssnittet sparar användarinformation lokalt [25]. En nackdel med sessioner till skillnad från token-baserad hantering är risken med CSRF attacker, som exponerar session-information, vilket kan innehålla känslig data. På grund av portalens behov, valdes en token-baserad autentisering för att skapa en mer säker och flexibel lösning.

5

Resultat

Resultatet av projektet är en webbportal utvecklad för att stödja och underlätta fritidsgårdens hantering av närvarohantering, poängsystem, användare och utskick av riktade notifikationer från fritidsledare. Portalens huvudsakliga funktioner omfattar användarhantering, poängutdelning, notifikationssystem och administrativ översikt. Portalen är rollbaserad och byggd med ett tydligt fokus på enkelhet, säkerhet och skalbarhet.

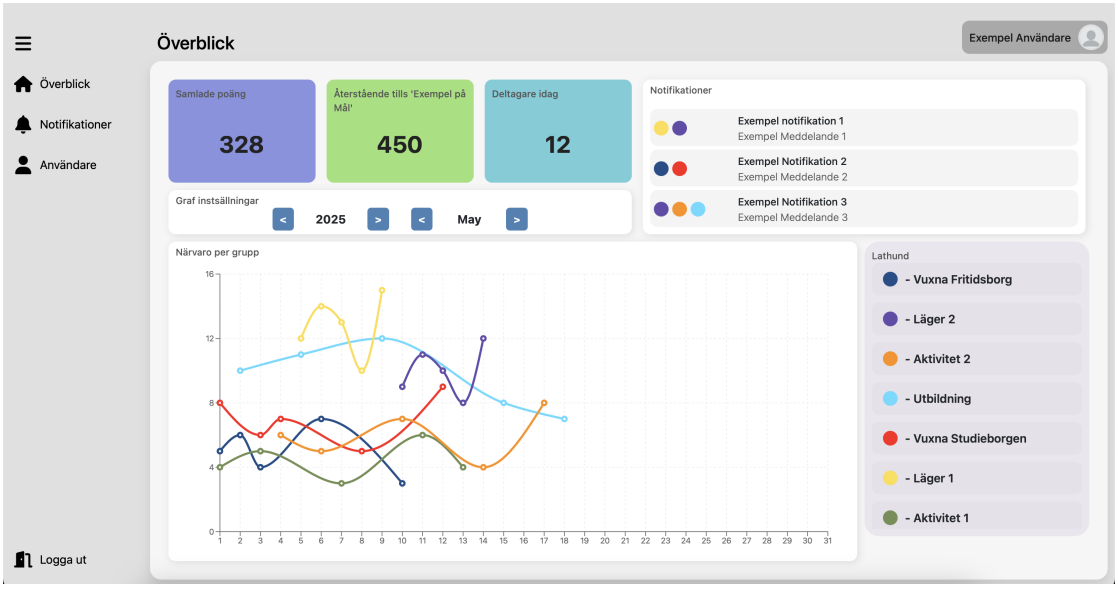
Under utvecklingen förändrades hanteringen av användargrupper. Ursprungligen var systemet utformat för att varje användare skulle tillhöra endast en grupp, vilket skulle kopplas till en åldersgrupp. I senare skede av projektet ändrades detta till att stödja flera grupper per användare, vilket gör systemet mer flexibelt och framtidssäkrat. Denna förändring påverkade flera delar av systemet och diskuteras mer ingående i diskussionskapitlet 6.3.

I projektets planeringsfas diskuterades en funktion för meddelanden mellan fritidsledare och applikationens användare, vilket även går att se i prototypen. Denna funktion togs bort tidigt från projektets mål då den bedömdes problematisk utifrån GDPR lagstiftning. Av denna anledning är meddelandefunktionen inte en del av det slutgiltiga resultatet.

5.1 Översiktssidan

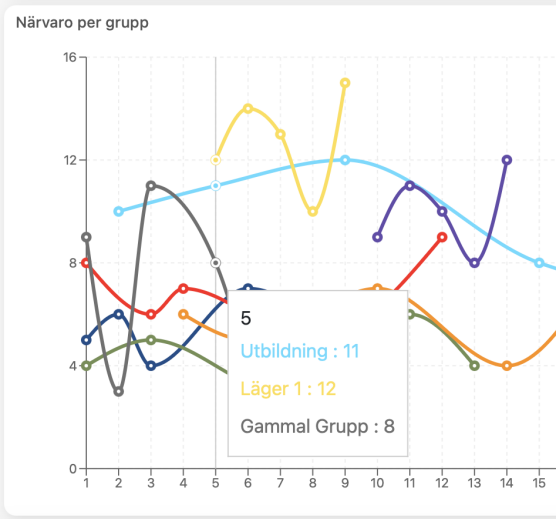
Översiktssidan fungerar som portalens landningssida och ger en snabb överblick på fritidsgårdens aktivitet och deltagande (se figur 5.1). Sidan är uppdelad i fyra delar som tillsammans presenterar och sammanfattar relevant information. Sidan gör det enklare för administratörer att direkt få en helhetsbild på aktuell information utan att behöva navigera vidare i portalen.

Högst upp på sidan kan administratören se det aktuella målet som fritidsgården arbetar mot. Information som målets beskrivning, det totala antalet poäng som krävs, intjänade poäng och dagens poäng visas här. Värdena uppdateras automatiskt utifrån användarnas aktivitet. Bredvid rutorna finns det även en ruta som presenterar de tre senaste notifikationerna som skickats från portalen.



Figur 5.1: Översiktssidan

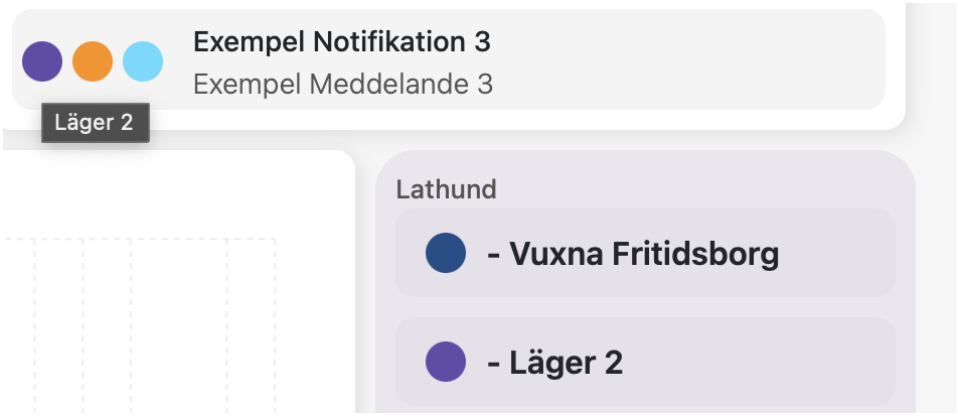
Den huvudsakliga funktionen av sidan är grafen som visualiserar fritidsgårdens deltagande under en månadsperiod (se figur 5.2). Statistiken hämtas från tabellen `attendance_log` i databasen, som uppdateras automatiskt varje gång närvaropoäng tilldelas till mobilanvändarna. Administratören kan navigera mellan månader och år för att jämföra deltagandet under olika perioder. Genom att föra muspekaren över grafens datapunkter visas detaljerad information om deltagandet under dagen.



Figur 5.2: Inzoomad bild på grafen

Grupperna representeras visuellt med unika färger, vilket speglas i grafen. För att underlätta finns det en kompletterande komponent som visar en översikt på de nuvarande gruppdefinitionerna och deras tillhörande färger och fungerar som en lathund (se figur 5.1). I hela portalen är cirklarna interaktiva, och gruppnamnet kan ses genom att hålla muspekaren över en cirkel, vilket kan ses figur 5.3.

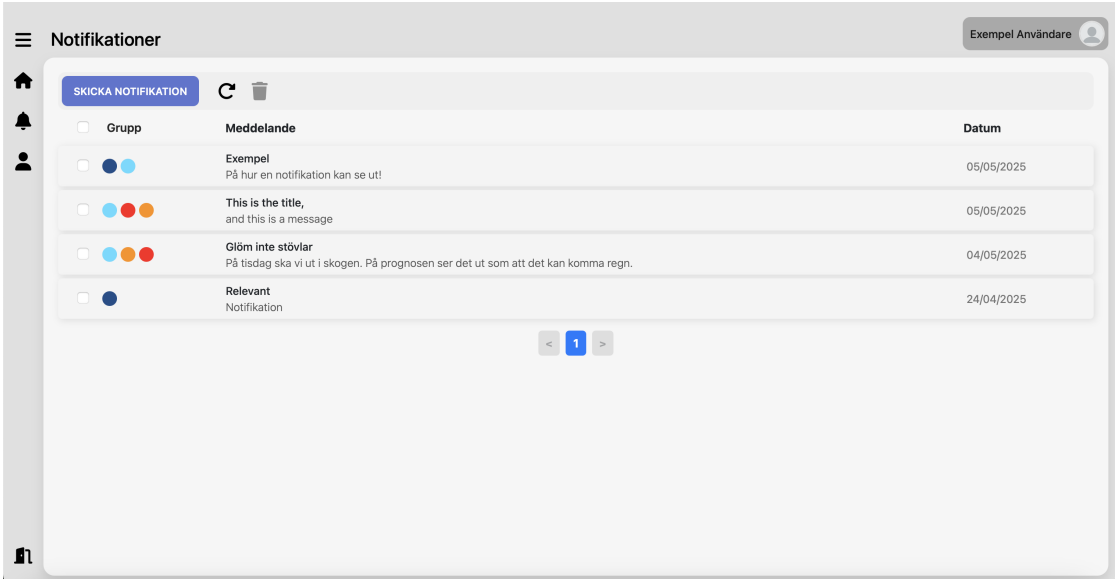
Grupper som tidigare använts men inte längre finns definierade markeras automatiskt med en grå färg. I grafen möjliggör detta fortsatt åtkomst till historisk statistik utan att behöva förvirra färgerna och underhålla gamla definitioner.



Figur 5.3: Muspekaren hovrar ovanför en färgcirkel

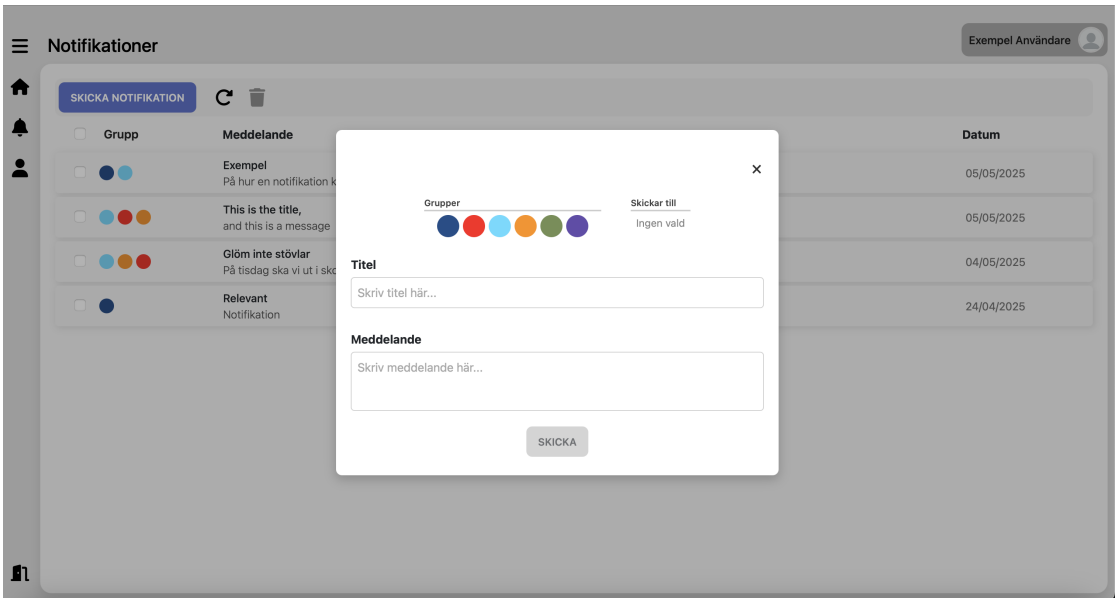
5.2 Gränssnittet för Notifikationer

En av portalens viktigaste funktioner är att hantera och skicka notifikationer till mobilapplikationens användare. Via notifikationssidan (se figur 5.4) har handledarna möjlighet att se en tydlig överblick på tidigare utskick och möjlighet att skicka nya notifikationer via en dialogruta (se figur 5.5).



Figur 5.4: Notifikationssidan

För att skicka en notifikation öppnar administratören en dialogruta där det är möjligt att fylla i en titel, ett meddelande och välja mellan en eller flera grupper som notifikationen skickas till (se figur 5.5).



Figur 5.5: Dialogruta för att skicka en notifikation

Grupper representeras som färgcirklar och flyttas från en lista med de tillgängliga till de valda grupperna genom att klicka på en grupp. Denna design gör det enkelt att veta exakt vilka grupper som administratören skickar notifikationen till. Precis som i resterande delar i portalen kan administratören hålla muspekaren över en cirkel för att se vilken grupp som den representerar (se figur: 5.3).

När en notifikation skickas sparas den automatiskt i en databas och blir synlig i listan. Varje notifikation visas med

- Titel,
- Meddelande,
- Datum,
- och vilka grupper den skickades till.

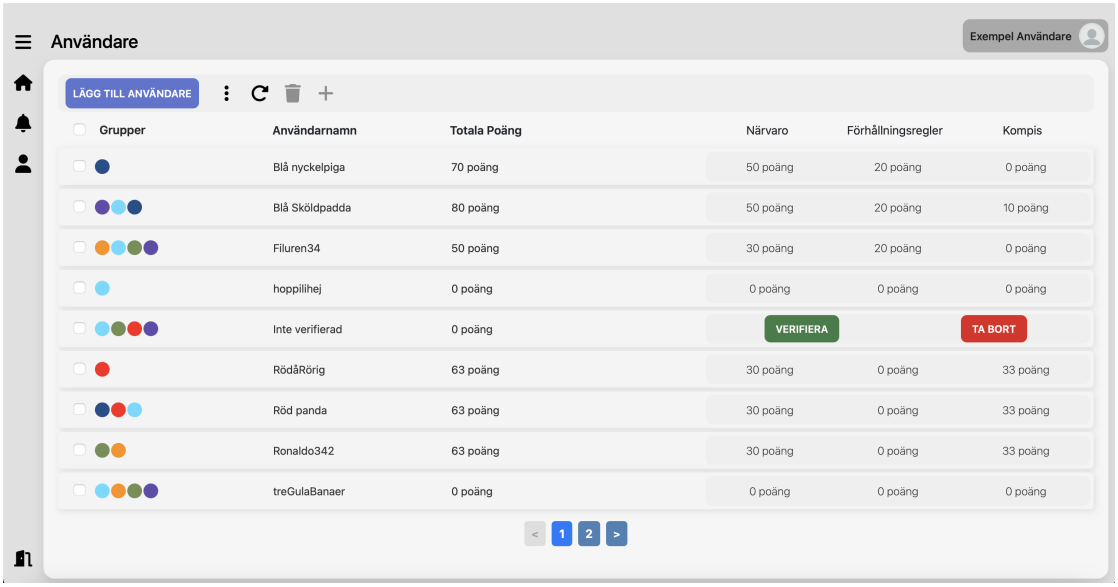
Portalen är byggd för att hantera ett stort antal sparade notifikationer. För att administratören ska få en översikt på alla notifikationer är listan paginerad, vilket innebär att notifikationerna delas upp över flera sidor, med ett begränsat antal per sida. Användare kan enkelt bläddra igenom sidorna genom att antingen tycka på pilarna eller respektive sidnummer.

För att underlätta navigering och sökning i listan finns det även funktionalitet för att sortera notifikationerna genom att klicka på respektive kolumnrubrik. Genom att trycka respektive kolumntitel är det möjligt att sortera notifikationerna. Det går endast att sortera för en kolumn in taget och är utformad för att underlätta vid sökning i listan.

5.3 Användarhantering för Mobilapplikationens Användare

Portalen innehåller två separata sidor för användarhantering, en för mobilapplikationens användare `Users` och en för webbportalens administratörer `Staff`. Båda sidorna bygger på ett liknande användargränssnitt men skiljer sig från varandra med målgrupp, syfte och underliggande logik.

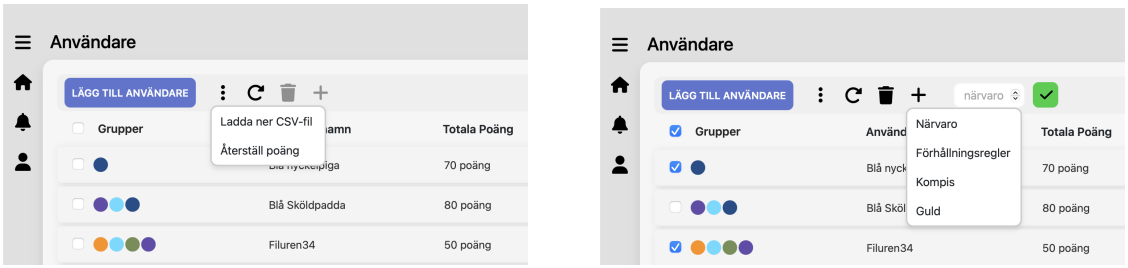
Sidan för mobilapplikationens användare gör det möjligt för handledarna att enkelt se en överblick över alla användarna, deras användarnamn, vilka grupper de tillhör och hur många poäng de har inom kategorierna 'Närvaro', 'Förhållningsregler' och 'Kompispoäng'. Poängen används för att främja deras deltagande i aktiviteter, bra uppföranden och deras sociala insatser.



Figur 5.6: Användarhantering för mobilapplikationen

Alla användare visas i en paginerad lista med kolumner för deras Användarnamn, deras totalpoäng, hur många poäng de har per kategori samt vilka grupper de tillhör.

För att skydda mot obehöriga användare från att få tillgång till känslig information, stödjer även systemet verifierade och overifierade användare. När en person laddar ner och registrerar sig i appen skapas automatiskt en overifierad användare, vilket betyder att denne inte kan ta emot notifikationer eller bli tilldelade poäng. Det är administratörernas jobb att verifiera de användare som de vet är godtyckliga, detta går att se direkt i listan om en användare är overifierad och uppmanas att verifiera eller radera användaren (se figur 5.6).



(a) Hantering av extra funktioner **(b)** Hantering av poängtilldelning

Figur 5.7: De två olika dropdown-menyerna som visar sidans funktionalitet.

Vid inläsning av sidan hämtas användardata från backend via funktionen `fetchUserAccounts`, som sammanfogar tabellerna `users` och `points` och returnerar den aktuella statusen på varje användare. Backends funktion är alltså att konvertera information från databasen till ett användbart format för användargränssnittet, och gör det möjligt för administratören att:

- **Tilldela Poäng** till en eller flera användare samtidigt via funktionen `updatePoints`, som uppdaterar databasen med respektive poängvärde och uppdaterar närvaron i taballen attendance_log.
- **Lägga till nya användare** via dialogrutan, där backend hanterar skapandet av en ny användare med en tom poäng tabell. Detta gör användaren direkt verifierad.
- **Återställa Poäng** via funktionen `resetPoints`, som återställer samtliga poäng i användarlistan.
- **Exportera Användarlistan** till en CSV fil för arkivering.
- **Radera Användare**, via `deleteUsers`, som raderar samtliga valda användare.

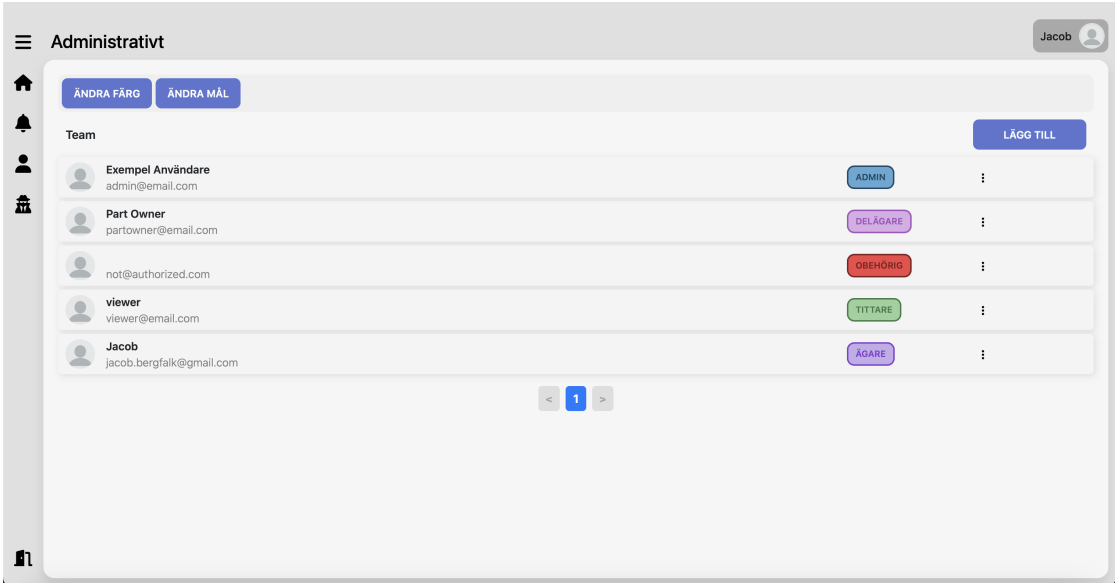
Dessa funktioner är tillgängliga i gränssnittet genom två dropdown-menyer. Se figur 5.7 för en visuell översikt på dessa menyer och deras placering i gränssnittet.

Syftet med poängsystemet är att motivera användarna att kämpa mot gemensamma mål. För att underlätta administrationen finns det möjlighet att ladda ner den aktuella användarlistan som ett Excel-ark, och när ett mål är uppnått finns det funktionalitet för att återställa alla poäng, vilket gör det smidigt att starta en ny poängperiod eller termin.

5.4 Användarhantering för Administratörer

Sidan för administratörshantering är gjord för att hantera de konton som har åtkomst till portalen och administratörens rättigheter. Dessutom innehåller sidan funktionalitet för att hantera gruppdefinitioner samt det aktuella målet som mobilapplikationens användare kämpar mot. Precis som användarhanteringen av mobi-

lapplikationens användare bygger sidan på en paginerad lista där varje administratör visas med användarnamn, e-post och vilken dess behörighet (se figur 5.8).



Figur 5.8: Användarhantering för Webbportalen

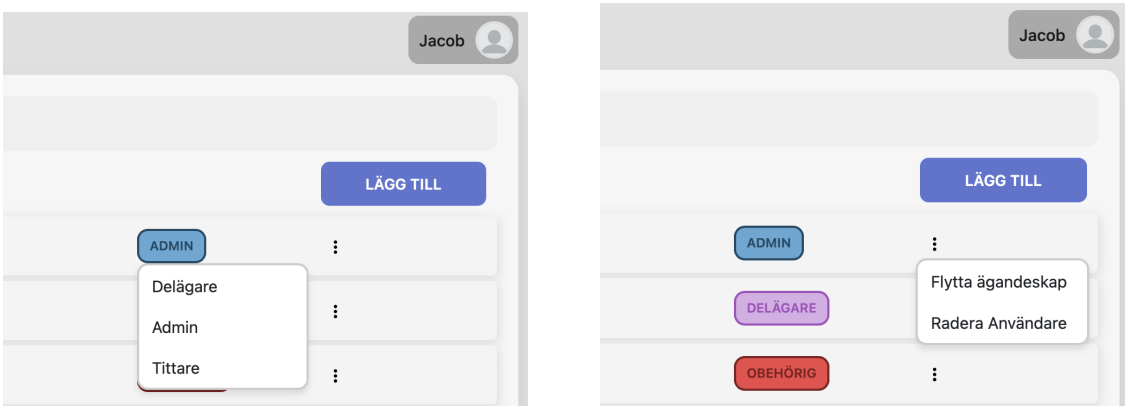
Alla administratörer hanteras genom Firebase Authentication, där varje administratör identifierad med ett unikt User ID (UID). Vid inläsning av sidan hämtas användardata från backend via funktionen `fetchUsers`, som med hjälp av Firebase Auth hämtar de administrativa rollerna.

De administrativa rollerna definierar vilka rättigheter administratören har och de kan tilldelas tre hierarkiska användarroller via portalen:

- **Tittare** har endast läsrättigheter och kan inte göra några ändringar i portalen.
- **Admin** har tillgång till nästan hela portalen, och kan skicka notifikationer, uppdatera användarpoäng och hantera mobilanvändare.
- **Delägare** har åtkomst till Administratörssidan och kan hantera, radera eller uppdaterar resterande administratörer.

Utöver dessa finns det ytterligare två tekniskt definierade roller:

- **Ägare** är den användaren med fullständiga rättigheter över portalen. Till skillnad från **Delägare** har Ägaren möjlighet att radera eller nergradera en Delägare. Rollen är unik och kan inte raderas, däremot kan ägaren flytta sitt ägande om denne önskar det.
- **Obehörig** markeringen är inte en direkt säkerhetsåtgärd utan fungerar som en varningssignal om något avvikande har inträffat, såsom ett intrångsförsök eller en otillåten kontoregistrering. Åtkomstkontroller hanteras i backend och säkerställer att obehöriga konton inte får tillgång, medan markeringen varnar. På grund av säkerhetsbegränsningar i Client SDK utgör denna ett viktig varningsmarkör för att upptäcka och hantera potentiella problem.



(a) Hantering av administratörsbehörigheter.

(b) Hantering av extra funktionalitet.

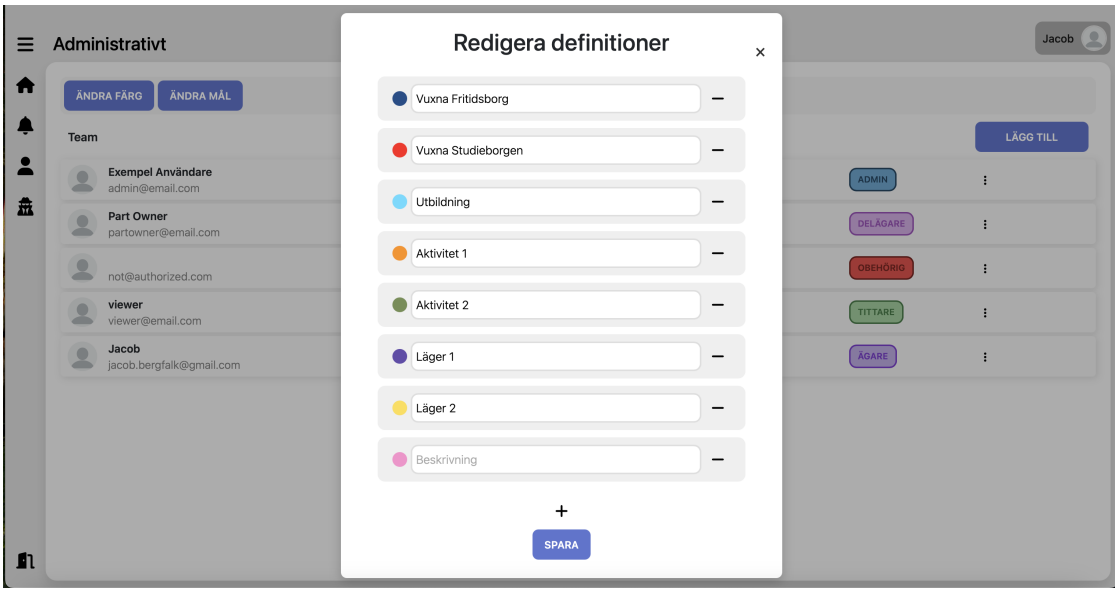
Figur 5.9: De två olika dropdown-menyerna för att hantera administratörerna.

Funktionalitet för administrationssidan är:

- **Skapa nya användare**: Via en dialogruta kan både en Ägare och Delägare lägga till nya administratörer genom att ange användarnamn, e-post och ett lösenord. Funktionen stödjer både E-post validering och lösenordsvalidering.
- **Byta roll**: Genom en dropdown-meny är det möjligt att ändra en administratörs roll. Systemet kontrollerar vilken behörighet administratören har och gör ändringen om det uppfylls.
- **Radera administratör**: En bekräftelsedialog visas innan ett konto raderas. Endast ägare kan radera en administratör. Den enda som har möjlighet att radera en delägare är ägaren.
- **Överföra ägandeskap**: Ägaren har möjlighet att flytta sin roll till en annan administratör. Då får den förra ägaren titel som delägare.

Gruppdefinitionerna är definierade med två tabeller i databasen.En som representerar vilka färger som tillåts: `colors`, och en som representerar de nuvarande grupperna som finns tillgängliga: `groups`. Vilka färger som finns och dess hexvärden är hårdkodade och fördefinierade i databasen. Det är möjligt att ändra eller ta bort befintliga definitioner eller lägga till nya via en dialogruta, men är begränsad till de befintliga fördefinierade färgerna (se figur 5.10).

Lösningen behöver vara flexibel för att inte skapa konflikter i resterande system. Exempelvis, om definitionerna ändras i framtiden, representeras de äldre grupperna av en grå cirkel istället. Även fast gruppen inte har någon form av funktionalitet, är det fortfarande möjligt att se gruppnamnet genom att hålla muspekaren över cirkeln, vilket kan ses i figur 5.3.

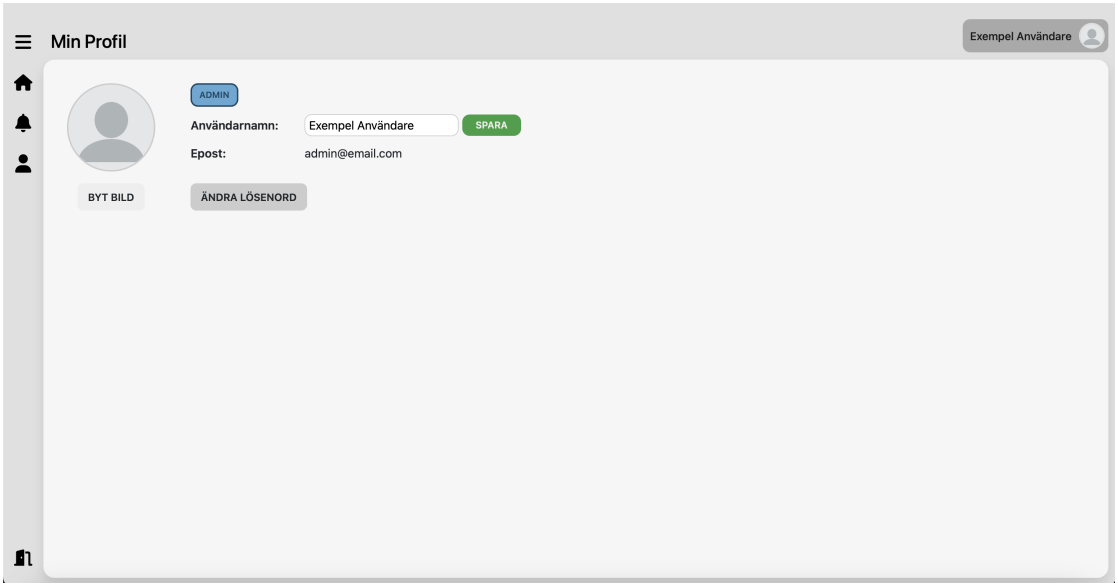


Figur 5.10: Gruppdefinitioner

Dessutom har administratörer möjlighet att ändra det nuvarande målet som mobilapplikationens användare kämpar efter via en dialogruta.

5.5 Kontohantering

Administratörer loggar in via en enkel sida där användaren anger sin e-postadress och ett lösenord. Funktionen använder Firebase Authentication för inloggningsfunktion och vid lyckad inloggning navigerar användaren till portalens överblickssida. Vid felaktiga uppgifter visas ett enkelt felmeddelande, vilket är på grund av att sidan är menad för intern användning.



Figur 5.11: Profilsidan

Alla Administratörer har tillgång till en profilsida (se figur 5.11) där grundläggande kontoinformation visas såsom:

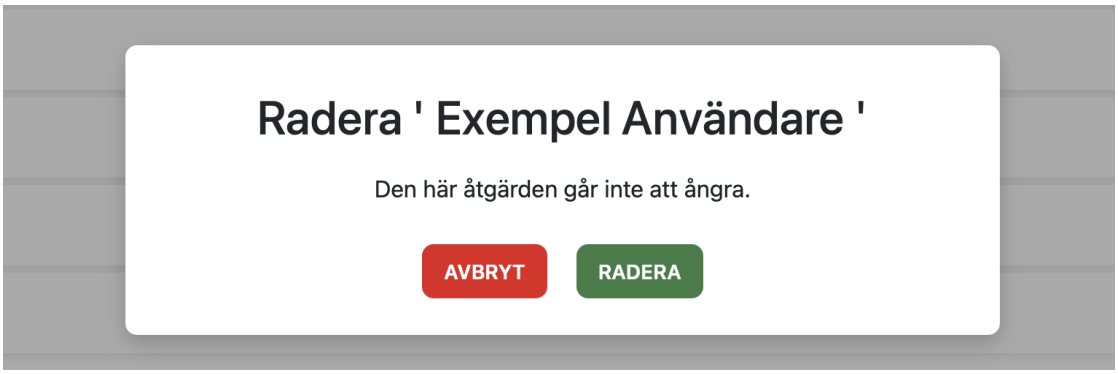
- **Profilbild** visas, men är inte implementerad.
- **Användarnamn** som kan uppdateras direkt på sidan.
- **Epost** som är kopplat till den inloggade administratören.
- **Administrativ roll** som den inloggade administratören har.
- **Lösenordsändring**, vilket inte är implementerat.

5.6 Interaktiva dialogrutor

En återkommande funktion i portalen är användningen av dialogrutor, vilket är ett grafiskt kontrollelement som visas ovanpå huvudinnehållet. Dialogrutorna används för att få input från administratören eller bekräfta åtgärder. Portalen använder två huvudsakliga typer av dialogrutor, *bekräftelsedialoger* och *inmatningsdialoger*. Båda är designade för att vara återanvändbara och följa en enhetlig visuell stil.

5.6.1 Bekräftelsedialoger

Bekräftelsedialoger används i situationer där administratören initierar en åtgärd som påverkar känslig information, till exempel vid återställning av alla poäng eller vid radering av en användare. Dialogrutans struktur består av två huvudkomponenter: `confirmModal`, som hanterar det visuella utseendet, och `onConfirm` som används i resten av portalen och styr textinnehåll och knappar, ett exempel visas i figur 5.12.



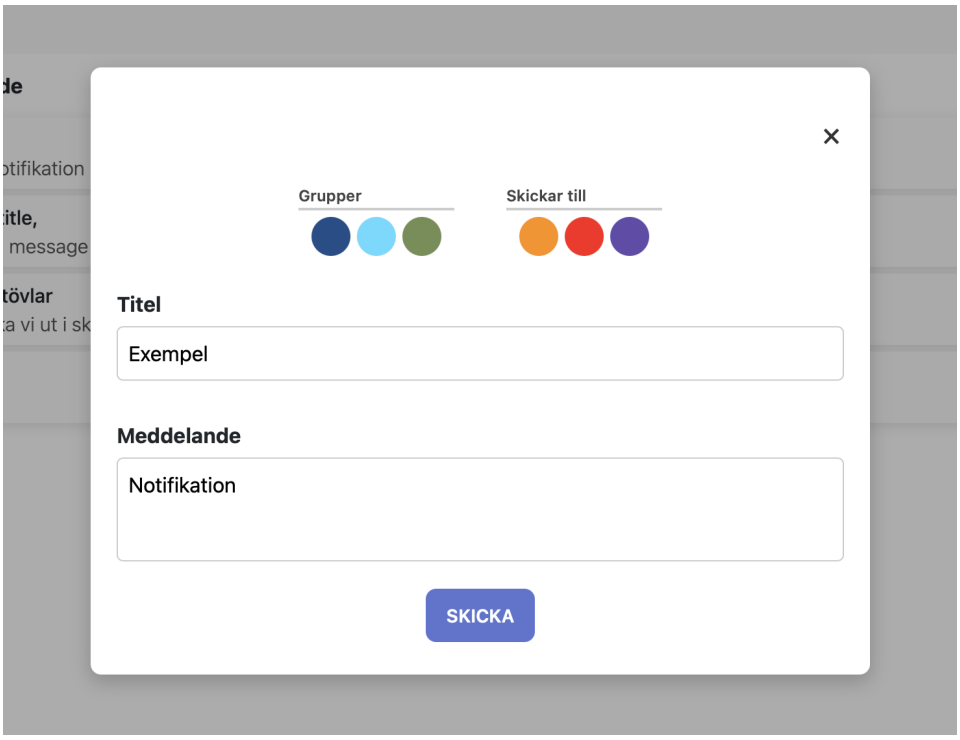
Figur 5.12: Exempel på hur bekräftelsedialogen kan se ut

Dialogen följer ett tydligt designmönster: ingen åtgärd genomförs om inte administratören aktivt bekräftar åtgärden. Genom att klicka utanför dialogrutan eller på avbryt, stoppas åtgärden utan förändring.

5.6.2 Inmatningsdialoger

Den andra typen av dialogruta används för att samla in information från administratören. Inmatningsdialogerna har ett liknande utseende och struktur som bekräftelsedialogerna, men skiljer sig i sitt syfte (se figur 5.13). De används bland annat för att:

- Lägga till en ny användare eller administratör
- Skicka en notifikation
- Ändra gruppdefinitionerna
- Uppdatera aktuella målet för användarna.



Figur 5.13: Exempel på hur en inmatningsdialog kan se ut

Inmatningsdialogerna är mer flexibla och funktionellt omfattande än bekräftelsedialogerna. Detta innebär dock att de ofta är mer specifikt utformade för ett ändamål, även om de bygger på gemensamma komponenter. Alla dialogrutor följer samma visuella riktlinjer, vilket skapar en enhetlig och användarvänlig upplevelse i portalen.

5.7 Backend

Backends arkitektur är precis likt den som nämns i 4.3, vars huvudsakliga funktion är att fungera som ett kommunikationslager mellan användargränssnittet och databasen. Samtliga API-Endpoints kräver en autentiserad användaren och sker via VerifyToken och VerifyRole, vilket förklaras utförligt i 4.3.1.

5.7.1 API Endpoints

Portalens användargränssnitt är uppdelat i funktionella sidor där varje sida använder sig av specifika Endpoints. Av den anledningen är Controller även uppdelad i respektive ansvarsområde, vilket tabellerna nedanför sammanfattar vilka Endpoints är kopplade till vilken funktionalitet i portalen.

SessionHandler

Metod / Endpoint	Query	Beskrivning
POST /sessionLogin	None	Loggar in en användare via Token
POST /sessionLogout	None	Loggar ut en användare

GroupDefinitions

Metod / Endpoint	Query	Beskrivning
GET /getGroupAndColors	None	Returnerar gruppdefinitioner och färger
GET /getGroupDefinitions	None	Returnerar definitionerna i View Group_definitions
POST /updateGroups	definitions	Uppdaterar grupperna som finns i databasen

statisticsHandler

Metod / Endpoint	Query	Beskrivning
GET /fetchActiveGoal	None	Hämtar aktuellt mål och poängstatus
POST /fetchAttendanceLog	year, month	Hämtar närvarostatistik för en specifik månad
POST /resetActiveGoal	name, points	Tar bort det nuvarande målet och placerar in ett nytt med givet namn och poäng

userHandler

Metod / Endpoint	Query	Beskrivning
GET /fetchUserAccounts	None	Returnerar användarkonton och deras poäng
POST /addUserAccount	username, groups	Lägger till en ny verifierad användare via funktionen i användargränssnittet
POST /updatePoints	selectedUsers, type, value	Uppdaterar användarpoäng och närvaro
POST /deleteUsers	selectedUsers	Raderar valda användare
POST /resetPoints	None	Nollställer alla användares poäng
POST /verifyUser	username	Verifierar användare som hittills bara är registerade i appen

notifications

Metod / Endpoint	Query	Beskrivning
GET /fetchNotifications	limit	Hämtar de senaste notifikationerna. Om limit är angiven returneras en begränsad mängd
POST /deleteNotifications	SelectedNotifications	Raderar de specificerade notifikationerna från databasen

staffHandler

Metod / Endpoint	Query	Beskrivning
GET /fetchUsers	None	Returnerar administratörkonton och dess respektive roll
POST /createNewStaff	username, email, password	Skapar en ny administratör med rollen `Tittare`
POST /deleteStaff	target	Raderar den valda administratören
POST /transferOwnership	target	Överför ägarskap
POST /updateRole	target, newRole	Uppdaterar administratörens roll

5.7.2 Databas

Systemet använder sig av en relationsdatabas där tabellerna är utformade för att vara tydligt separerade för att förenkla underhåll. De huvudsakliga tabellerna i databasen är:

- `users`: Innehåller alla mobilanvändare med information som användarnamn, grupptillhörigheter och om användaren är verifierad.
- `points`: Innehåller poängen för varje användare, fördelar på vilken poängtyp det är.
- `notifications`: Innehåller alla notifikationer med titel, meddelande, grupper och datum.
- `groups`: Representerar alla aktiva grupper och dess definition.
- `colors`: Alla tillåtna färgkoder.
- `attendance_log`: Daglig närvaro aggregerat per grupp. Uppdateras automatiskt när närvaropoäng tilldelas.
- `active_goal`: Innehåller information om det nuvarande målet, poängen som behövs och samlade poäng.

Dessutom finns det två stycken views eller vyer som förenklar och minskar logik i användargränssnittet:

- `group_definitions`: Sammanfogar `groups` och `colors` för att sammankoppla gruppen med respektive hexvärde.
- `total_points`: Beräknar varje användares totalpoäng och grupperar poängen per kategori.

6

Diskussion

Syftet med projektet var att utveckla ett administrativt gränssnitt för Reningsborg, med målet att underlätta fritidsgårdens hantering och administrering av användare, grupper, poängsystem och kommunikation med deltagare. Gränssnittet skulle fungera som ett intuitivt verktyg, med fokus på användarvänlighet, säkerhet och framtida flexibilitet. Det huvudsakliga syftet med portalen var att stödja funktionalitet för att:

- Tilldela poäng till användare
- Se en överblick på mobilappens användare
- Gränssnitt för att skicka riktade notifikationer till mobilapplikationens användare

Samtliga av målen har uppfyllts i projektets slutliga version. Funktionerna är implementerade, testade och har använts av praktiska skäl under projektets gång. Användargränssnittet är utformat med ett fokus på användarvänlighet och backend har strukturerats för att vara säkert och flexibelt. Administratörernas roller hanteras via en säker Middleware och databasen lagrar endast den mest nödvändiga informationen.

Samtidigt finns det delmål i dessa som inte har uppfyllts. Exempelvis saknas det stöd för CSP (Content Security Policy), vilket är en viktig komponent i modern webbsäkerhet för att skydda mot attacker från externa källor. Även fast portalen är säkrad med Firebase Authentication, rollhantering och körs på Reningsborgs interna Server, innebär avsaknaden av vissa säkerhetsfunktioner att det finns utrymme för vidare förbättring.

Diskussionen fokuserar därför vidare på styrkor, begränsningar och potentiella förbättringsområden utifrån projektets mål, den tekniska lösningen och användarperspektivet.

6.1 Responsiv Design och Användarvänlighet

Användarvänlighet har varit en central utgångspunkt i utvecklingen av portalen. Eftersom målgruppen för portalen består av användare med varierande teknisk kompetens, har stor vikt lagts på att utforma gränssnittets design så intuitivt och förutsägbart som möjligt. Målet var att användaren ska känna igen sig och förstå hur systemet fungerar och vad som förväntas av dem. På grund av detta var inspirationen av redan etablerade verktyg väldigt centralt i utvecklingen av portalen.

En medveten designprincip som tillämpats är att positiv återkoppling inte sker i form av en bekräftelsedialog. Administratören får istället indirekt bekräftelse genom att resultatet av handlingen visas direkt i gränssnittet. Ett exempel är tilldelningen av poäng, vilket ger en bekräftelse genom att listan uppdateras med de nya poängen, utan att en dialogruta visas. Administratörerna kan alltså förvänta sig att förfrågan besvaras.

I portalen förekommer även gråmarkerade element. Dessa representerar antingen information, såsom tidigare gruppdefinitioner som inte längre används, eller funktioner som inte är aktiverade, till exempel ofullständig inmatning av information, såsom ett användarnamn som inte är ifyllt vid skapandet av en användare. Alla element på hemsidan är responsiva till en viss nivå.

En annan viktig designaspekt är att sidan inte är uppbyggd för vertikal scrollning. Istället är portalen konstruerad som en fast administrativ vy, där all information och funktionalitet ryms på skärmen. Detta förstärker upplevelsen att arbeta i ett strukturerat administrativt gränssnitt, snarare än en traditionell hemsida. Däremot finns det komponenter som behöver vara dynamiska och anpassningsbara. Ett exempel är användarlistorna som måste variera i storlek beroende på antalet användare. Listorna är därför anpassningsbara för att scrolla inom en fast vy, vilket endast aktiveras om administratören är tillräckligt inzoomad.

6.2 Backend och Säkerhet

Ursprungligen var backend menat att vara ett relativt enkelt system, där huvudsakliga syftet var att hantera dataöverföring mellan databasen och användargränssnittet. Den initiala idén var att efter en administratör hade verifierats, skickades all information av användarna och notifikationerna till användargränssnittet, där all filtrering, sortering och presentation hanterades. Endast vissa åtgärder, såsom att radera eller lägga till användare, var tänkta att utföras av backend.

Denna lösning var motiverad av antagandet att systemet skulle hantera en relativt begränsad mängd användare och notifikationer, vilket gjorde det acceptabelt att belasta gränssnittet. Däremot, i takt med att projektet växte och komplexiteten ökade, blev det tydligt att det fanns ett behov av en mer omfattande, strukturerad och säkrare backend.

Istället för att användargränssnittet skulle anpassa sig utifrån vad som finns i backend, utvecklades backend utifrån gränssnittets behov. Varje API-endpoint skapades för att uppfylla ett specifikt funktionellt behov, vilket resulterade i en målinriktad och behovsanpassad backend. Backends arkitektur bygger på en så kallad *layered structure*, där ansvar fördelas mellan router, controller och middleware. Detta visade sig vara särskilt värdefullt i en iterativ utvecklingsprocess, där krav och mål ändrades över tid.

6.2.1 Firebase Authentication

En särskilt viktig komponent för systemets säkerhet var integrationen med Firebase Authentication. Detta möjliggjorde säker hantering av inloggning och profilhantering, men möjliggjorde även rollbaserad åtkomst genom användningen av `customClaims`. I början användes rollerna främst för att begränsa åtkomsten till viss funktionalitet, exempelvis att endast tillåta ägare åtkomst till användarhanteringen. I takt med att projektet växte blev rollhanteringen en avgörande del i portalens säkerhet.

Användargränssnittet använder sig av Firebase Client SDK för inloggning, vilket även inkluderar en teknisk sårbarhet, det är tekniskt möjligt att skapa nya administratörer direkt i gränssnittet. Detta kan i teorin utnyttjas för att skapa konton som försöker agera som administratörer. Däremot, genom att backend kontrollerar varje användares roll innan en förfrågan, motverkas denna begränsning. Det beror på att roller som sätts via `customClaims` endast är tillgängligt från Firebase Admin SDK, vilket inte är tillgängligt från gränssnittet.

Detta innebär att även om ett konto skapas via klienten kommer den inte tilldelas någon administrativ behörighet, och visas med en obehörig roll i gränssnittet, vilket även kan ses i figur 5.8. Rollerna fungerar alltså som ett extra säkerhetslager och en nödvändig förutsättning för att få åtkomst till portalens funktionalitet och information.

Rollsystemet, som inte var en del av den ursprungliga prototypen, kom att bli en en av de viktigaste säkerhetskomponenterna. Det säkerställer att endast auktoriserade användare får åtkomst till portalen, samt att ytterligare validering av behörighetsnivå sker innan en förfrågan godkänns.

6.2.2 Begränsningar och Förbättringspotential

Trots att systemet fungerar som förväntat, finns det vissa tekniska begränsningar att beakta. Flera av nuvarande controllerfunktioner returnerar hela innehållet i en databastabell, exempelvis alla användare eller notifikationer. Denna lösning kommer fungera i dagsläget, eftersom mängden data är hanterbar, men risken är att gränssnittet inte kan hantera detta i framtiden i samband med att användarbasen växer och fritidsledarna skickar notifikationer. Idealt sett skulle funktionerna endast returnera en begränsad mängd data eller en sida i taget, och rensa cache på element som inte är relevanta. Däremot är detta en funktionell begränsning, då nuvarande databas inte har möjlighet att hålla kolla på vilka element som redan har

returnerats.

En annan förbättringsåtgärd skulle vara att implementera loggning av alla känsliga förfrågningar som utförs av administratörer. Genom att dokumentera och spåra till exempel rolländringar, raderingar eller utskick av notifikationer, ökar transparensen. Det stärker ansvarstagandet från administratörerna och bidrar till en ökad säkerhetsnivå.

6.3 Reflektion kring den iterativa arbetsprocessen

Projektet nyttjade en iterativ arbetsprocess, där utvecklingen av portalen skedde parallellt med andra utvecklingsprojekt i samband med en agil arbetsmetod. Detta innebär att portalens funktionalitet kontinuerligt kunde utvärderas och anpassas i samband med att nya behov uppstod. Arbetsmetoden gjorde det möjligt att arbeta flexibelt med fokus på problemlösning, vilket underlättade när det inte fanns konkreta mål för portalen i början. Det gav möjlighet och utrymme att testa lösningar och förbättra dom, vilket skulle vara svårare med en linjär arbetsmodell.

Även fast metoden visade sig vara en styrka, blev det dessutom en utmaning. Den största bristen visade sig vara när grundläggande struktur behövdes korrigeras och omfattande förändringar behövdes göras.

Ett konkret exempel var hanteringen av användargrupper. Från början var systemet utformat för att varje användare skulle tillhöra endast en grupp, där varje grupp skulle representeras av en ålderskategori som var kopplad till en färg. Detta låg som en grund för användarhanteringen, visning av grafen, statistik och notifikationssystemet. Däremot uppkom önskningar från Reningsborg sent i processen att en användare kunde vara del i flera grupper och att gruppdefinitionerna kunde ändras i framtiden.

Medan detta är en bättre lösning för systemets syfte, innebar det att den tidigare lösningen blev föråldrad. Grupperna var initialt definierade i användargränssnittet och saknade koppling till varandra och backend, utöver de förvalda hex-värderna för färgerna. För att integrera den önskade funktionaliteten behövdes grundläggande ändringar göras, och ett helt nytt system utvecklades för att hantera gruppdefinitionerna. Den nya lösningen uppfyllde de kraven som Reningsborg hade och det är nu möjligt att ändra befintliga, lägga till nya och ta bort gruppdefinitioner, samt är den nya lösningen integrerad i resten av systemet.

Det största problemet med denna förändring var de strukturella följdeffekter som behövdes hanteras i samband med det nya systemet. Funktionerna som var byggda för det förra systemet påverkades och flera delar av systemet behövde justeras eller omarbetas. Dessa följdeffekter visar svagheterna i att inte ha en mer genomarbetad planeringsfas från början. Att tidigare identifiera vilka delar som behövdes byggas upp från grunden hade minskat omarbetning och frigöra mer tid för vidareutveckling.

6.4 Icke uppnådda mål

Den iterativa arbetsprocessen möjliggjorde stor flexibilitet i utvecklandet, men innebar att viss funktionalitet behövde byggas om eller förskjutas till senare delar av arbetet. Dessa förändringar var nödvändiga för att skapa ett flexibelt och skalbart system, men betydde även att vissa delar av den ursprungliga planeringen behövdes prioriteras bort.

6.4.1 Meddelanden

Vid projektets start fanns det två huvudsakliga mål: att kunna skicka riktade notifikationer till olika grupper, samt att skapa ett gränssnitt för meddelanden mellan användare och administratörer. Medan första målet blev realiserat, kunde inte meddelandefunktionen påbörjas. Meddelandesidan var menad att vara en trygg direkt kontaktkanal från användarna till administratörerna, som kunde användas för enkel kommunikation såsom frågor eller för diskret kommunikation för att lyfta personliga problem.

Tidigt i projektet var det tydligt att denna funktion inte var en nåbart mål. Den främsta anledningen var kring GDPR lagstiftning och Apples och Androids användarvillkor kring meddelanden och chattfunktionalitet för barn. En möjlig lösning var att begränsa funktionen till användare över en specifik ålder, men systemet kan inte hantera åldersuppgifter enligt GDPR begränsningar och det var inte tekniskt möjligt att säkerställa detta på ett tillförlitligt sätt. För att undvika en juridisk gråzon valdes funktionaliteten bort.

6.4.2 Statistik

Under planeringsfasen uppkom en önskan att kunna visa övergripande statistik om fritidsgården och dess engagemang. Sidan skulle också ge en större mening för rollen Tittare, som i dagsläget har en begränsad överblick på portalen. Bland annat fanns idéer om att visa statistik om:

- Antal aktiva konton över tid
- Deltagande per grupp
- Vilka aktiviteter som får mest deltagare

Likt meddelandefunktionen var GDPR en av de största begränsningarna. Endast en begränsad mängd data kunde sparas per användare, vilket i sin tur påverkade vilken information som kunde visualiseras. Medan exemplet ovan inte ser ut att strida mot GDPR, krävs det att systemet sparar när en användare har befunnit sig på fritidsgården. Det väckte även frågan om den statistik som kunde presenteras verkligen var relevant eller användbar för administratörerna.

Utöver de juridiska aspekterna spelade även en tidsfaktor en stor roll. Eftersom andra delar av systemet prioriterandes, blev det svårt att argumentera för att utveckla en sida som potentiellt inte var relevant för administratörerna i slutändan. Däremot kunde vissa delar av den planerade statistiken fortfarande användas, och har blivit implementerat i översiktssidan, exempelvis närvarografen och dagens deltagare.

6.4.3 Profilhantering

Till skillnad från statistik- och meddelandesidorna, som aldrig påbörjades, kunde en fungerande men förenklad version av profilhantering levereras. Denna täcker den grundläggande funktionaliteten som krävs, men saknar flera funktioner som initialt planerades. Målet var att administratörerna själva skulle ha större kontroll över sitt konto och hade bland annat möjlighet att:

- Byta lösenord,
- Ändra E-post adress,
- Redigera sin profilbild
- och nya administratörer skulle registrera sig via en inbjudningslänk från en ägare eller delägare

Idén var att utnyttja Firebase till fullo, varav många av dessa funktioner är tillgängliga via Firebase Authentication, men flera tekniska hinder och prioritering gjorde att detta inte realiserades. Till exempel finns det inget sätt att återställa ett lösenord idag, vilket gör att den smidigaste lösningen är att radera och skapa ett nytt konto istället.

En viktig planerad aspekt med profilhanteringen var profilbilder. Funktionaliteten var menad att spela en stor roll i meddelandesidan, men också vara betydande för att enklare differentiera mellan administratörer i portalen. Idag finns det en dedikerad plats för profilbilden, däremot finns det inget stod för att ladda upp eller ändra bilden. På profilsidan finns det finns en dedikerad knapp för att byta profilbild, men det finns ingen implementerad funktionalitet bakom den.

Det främsta hindret för detta var dels tidsbrist, men också flera praktiska begränsningar. För att hantera profilbilder kräver inte bara hantering av filuppladdning och lagring, utan kräver också stöd för olika filformat, bildkomprimering och ytterligare säkerhetsåtgärder för att förhindra intrång.

Dessutom hade utveckling av registeringssidan och ett fungerande system för lösenordsåterställning varit väldigt krävande. För dessa skulle det behövas en e-postserver för e-postutskick, vilket låg utanför projektets tekniska ramar. Istället valdes det att prioritera utvecklingen av kärnfunktioner som direkt arbetar mot portalens huvudsyfte.

6.5 Licenser

Under utvecklingen fattades beslutet att begränsa användningen av externa bibliotek för funktionalitet. Eftersom Reningsborg har varierade tekniska kompetens inom området var målsättningen portalen krävde så lite tekniskt underhåll som möjligt. Exempel på funktionalitet som övervägdes mot ett externt bibliotek är:

- Dropdown-menyer
- Konfirmationsrutor
- Graf
- Nedladdning av filer

Medan många av dessa var kostnadsfria att använda, visade det sig att de antingen hade en begränsad licens eller copyright avtal som inte är lämplig för distribution eller kommersiellt bruk, vilket skapade osäkerhet. Det uppstår problem om en licens ändras i framtiden eller om en komponent eller beroende slutar fungera. Eftersom vissa av dessa komponenter påverkar portalens grundfunktionalitet så skulle hela portalen påverkas om det blir problem i framtiden. För att minska risken utvecklades dessa komponenterna manuellt istället, vilket ger full kontroll över komponenterna och säkerställer att grundfunktionaliteten fortsätter fungera.

Det finns dock undantag där tredjepartsbibliotek används, speciellt i situationer där det är orimligt eller för tidskrävande att bygga eget. Exempel på detta är bibliotek för grafen eller för nerladdning av användardata. Skillnaden mellan dessa bibliotek, gentemot dropdown-menyn och konfirmationsrutan, är att dessa endast påverkar sin egen funktionalitet. Om det skulle uppkomma problem kan dessa bara plockas bort eller bytas ut.

6.6 Användarutvärdering

Även fast användarvänligheten är en central del av projektet, är en av de största begränsningarna i projektet att portalen inte har systematiskt testats av externa användare. Det saknas alltså konkret data eller strukturerad feedback från personer utanför projektgruppen . En mer omfattande utvärdering hade gett insikter om vilka funktioner som upplevs användarvänliga, vilka delar som upplevs otydliga samt vilka aspekter som väcker frustration. Den typ av information hade varit värdefull för att stärka portalens användarvänlighet och tydlighet.

Eftersom projektet genomfördes med en iterativ arbetsmetod, delades feedback från Reningsborg under loppet av utvecklingen. Denna återkoppling låg till grund för flera förändring och förbättringar under projektets gång, exempelvis hur gruppdefinitionerna hanteras eller hur användarna tilldelas poäng. Ett stort designändring var ändringen från att tilldela poäng till enskilda användare, till att flera användare tilldelas samtidigt, vilket kan ses i figur 5.7.

Även fast detta bidrog till användarvänligheten av portalen, finns det begränsningar i att endast utgå från interna perspektiv. En utvärderingsprocess med externa användare hade skapat en mer objektiv och strukturerad förståelse av användargränssnittets styrkor och svagheter.

Att användarundersökning inte genomfördes är en konsekvens av den iterativa arbetsmetoden. Projektets fokus låg på att utveckla de funktioner som Reningsborg önskade och på de prioriteringar som ansågs viktiga för slutprodukten. Detta innebär att flera funktioner lågprioriterades eller inte blev utvecklade i tid tills projektets slut. I slutändan baserades resultatet i stor utsträckning på den återkoppling som Reningsborg gav under projektets gång, vilket kan anses som en styrka i form av anpassning, men också en svaghet i form av användarvänlighet och generaliserbarhet.

6.7 Framtida arbete

Även fast portalen har uppnått sitt huvudsakliga mål, finns det flera områden som identifierats som potentiella förbättringsområden.

6.7.1 Profilbilder

I nuläget finns det ingen implementation eller stöd för profilbilder. Detta skulle kunna användas för att göra notifikationer och eventuella framtida meddelanden mer personliga och skulle bidra till ett större ansvarstagande av administratörerna. Medan funktionaliteten är delvis förberedd, krävs en omfattande genomtänkt lösning som kan hantera filuppladdning, lagring och hantering av olika bildformat på ett säkert sätt. En genomtänkt lösning kommer bidra till en mer personlig lösning mellan användarna och administratörerna.

6.7.2 Bredare användarprofiler

En brist med nuvarande system är att det inte finns en mer konkret användarprofil, både för administratörer men även för mobilapplikationens användare. Som tidigare nämnt i avsnitt 6.4.3, skulle en mer robust och användarvänlig alternativ vara att införa en metod att själv kunna återställa sitt lösenord. Firebase egna lösning förutsätter att det behövs en fungerande e-postserver, vilket låg utanför detta projekt.

Ytterligare kan användarhanteringen av mobilapplikationens användare också förbättras. Medan användarna är begränsade av GDPR lagstiftning, skulle en potentiell lösning för att överföra användarnas uppgifter, antingen automatiskt eller manuellt, till en annan mobil vara en önskad funktion. Detta kräver en annorlunda hantering av användarna i backend, och skulle vara utanför möjligheterna för detta projekt.

6.7.3 Säkerhet

Även om portalen i dagsläget bygger på och är beroende av Firebase Authentication, finns det tydliga begränsningar i hur säkerheten hanteras. Systemets nuvarande säkerhetsfilosofi bygger i stor utsträckning på transparens snarare än en barriär. Det vill säga att hellre upptäcka och visa potentiella intrång än att förlita sig på att intrång inte sker. Däremot skulle nästa steg vara att kombinera detta till en säkrare design, som både blockerar bättre samt är transparent när intrång väl sker.

Dessutom skulle sidan nyttjas av att införandet av loggningsfunktioner som behåller information om vilka handlingar som varje administratör gör på hemsidan. Skulle återigen bygga på nuvarande filosofi att det ska vara transparent och handlingar skulle vara spårbara.

6.7.4 Mobilanvändning

Portalen är i dagsläget utvecklat primärt för datoranvändning, vilket avspeglar Reningsborgs tidigare arbetsmetod, där administrationen skett via Exceldokument. Mobilanpassning har därför inte prioriterats under utvecklingsarbetet. Trots detta har det framkommit önskemål kring att portalen kan användas på mobila enheter, såsom telefon eller surfplatta, vilket skulle skapa en mer flexibel lösning för fritidsledarna.

Att anpassa portalen till mobilanvändning innebär dock betydligt mer än enbart justering av stilreglerna (CSS). Det krävs även tekniska förändringar, särskilt i hur komponenterna fungerar på mindre skärmar. Ett konkret exempel är listkomponenterna i användarhanteringen, där varje användare presenteras med information såsom poäng, grupper och profilinformation. Det blir särskilt utmanande att bestämma vad eller hur detta ska visas på en mindre skärm, antingen om det finns en expanderbar profilvy, eller om viss information ska ignoreras. Den nuvarande lösningen förutsätter att det finns tillräckligt mycket horisontellt utrymme.

Att anpassa portalen för mobilanvändning skulle inte bara förbättra användarupplevelsen, utan även göra verktyget mer flexibelt i olika miljöer. Det skulle underlätta snabb åtkomst till poängutdelning och notifieringshantering även när fritidsledarna inte har tillgång till en dator. Därför bör detta betraktas som ett prioriterat förbättringsområde för framtida vidareutveckling.

6.8 Hållbar Utveckling

Projektet har krävt särskild hänsyn till etiska aspekter och hållbar utveckling, särskild med tanke på att systemet hanterar information om barn och ungdomar. Ett av de mest centrala målen har därför varit att säkerställa att följa GDPR och skydda användarnas personliga integritet.

Systemet är utformat för att spara en minimal mängd data per användare, vilket betyder att ingen information efterfrågas om den inte är nödvändig för portalens funktionalitet. Exempelvis efterfrågas varken namn, kön eller e-postadress, och användare identifieras genom unika användarnamn som inte kan kopplas direkt till en individ.

Behörighetssystemet har varit en central del i både i portalens säkerhet men även utifrån ett hållbarhetsperspektiv eftersom det skapar transparens och ansvarstagande av administratörerna. Dessutom bidrar hela projektet till social hållbarhet genom att främja social tillvaro till barn och ungdomar som kommer ha en bra anledning att vara på fritidsgården.

Även om projektet inte har en direkt miljöpåverkan i form av fysiska resurser, finns det aspekter som bidrar till minskad miljöbelastning. Portalen är byggd för att vara resurssnål både i datalagring och i drift. Genom att endast behandla nödvändig data och undvika externa beroenden, kan detta bidra till att minska den totala energiåtgången för systemet.

Sammanfattningsvis stödjer projektet hållbar utveckling genom att främja integritet, säkerhet och resurseffektivitet med särskilt fokus på barns rätt till trygg digital närvaro.

7

Slutsats

Detta projekt har resulterat i en webbportal som stödjer Reningsborgs fritidsgård i dess arbete med närvarohantering, poängsystem och riktad kommunikation till deras deltagare. Genom att kombinera ett administrativt gränssnitt med användarhantering och ett rollbaserat behörighetssystem har portalen lagt grunden för ett användarvänlig och skalbart verktyg, anpassat för Reningsborgs behov.

Under utvecklingen har projektet utforskat hur en säker och modern webbapplikation, vilket inkluderar användargränssnittet, backend och databas, är uppbyggd. Med hjälp av en iterativ arbetsmetod har funktioner och struktur successivt anpassats från nya eller förändrade mål, krav och insikter som uppstod under utvecklingen. Denna flexibilitet möjliggjorde att systemet kunde förbättras och förändras till en mer dynamisk portal.

Trots att vissa funktioner, såsom fullständig profilhantering och statistik inte hann implementeras, har portalen nått sitt huvudsakliga syfte. Resultatet är ett första steg i att digitalisera fritidsgårdens deltagarhantering och kommunikation.

Litteraturförteckning

[1] Myndigheten för ungdoms- och civilsamhällesfrågor (MUCF), "Unga med låg utbildning – en riskgrupp för socialt utanförskap", 2021. [Online]. Tillgänglig: https://www.mucf.se/sites/default/files/2022/07/MUCF_Rapport_UtanfÃűrskap_och_unga_TANP.pdf (hämtad: 2025-05-13)

[2] Edutopia, "How to Use Gameplay to Enhance Classroom Learning", 2021. [Online]. Tillgänglig: https://www.edutopia.org/article/how-use-gameplay-enhance-classroom-learning (hämtad: 2025-02-05)

[3] Reningsborg, Reningsborg – Socialt arbete och second hand"2025. [Online]. Tillgänglig https://www.reningsborg.se (hämtad: 2025-02-05)

[4] Council of the European Union, The general data protection regulation (GDPR)", 2025. [Online]. Tillgänglig: https://www.consilium.europa.eu/en/policies/data-protection-regulation (hämtad: 2025-02-05)

[5] Asana, Förstå iterativa processer, med exempel", 2025. [Online]. Tillgänglig: https://asana.com/sv/resources/iterative-process (hämtad: 2025-05-13)

[6] blog by pixartprinting, Figma: vad är det och hur fungerar det?", 2023. [Online]. Tillgänglig: https://www.pixartprinting.se/blog/figma-vad-det-ar/?srsltid=AfmBOopCeY8yrP_peb2JhPrhXWHQbXAUwlnb5eURJcTFODQ0cc-xFSOG (hämtad: 2025-04-10)

[7] Mozilla web docs, "HTML: HyperText Markup Language", 2025. [Online]. Tillgänglig: https://developer.mozilla.org/en-US/docs/Web/HTML (hämtad: 2025-04-10)

[8] W3 Schools, Åhat is HTML?", 2025. [Online]. Tillgänglig: https://www.w3schools.com/whatis/whatis_html.asp (hämtad: 2025-04-10)

[9] Mozilla web docs, "CSS: Cascading Style Sheets", 2025. [Online]. Tillgänglig: https://developer.mozilla.org/en-US/docs/Web/CSS (hämtad: 2025-04-10)

[10] Bootstrap, Build fast, responsive sites with Bootstrap", 2025. [Online]. Tillgänglig: https://getbootstrap.com (hämtad: 2025-05-13)

[11] Digitala Lyftet, "Vad är JavaScript och vad använder man det till?", 2023. [Online] Tillgängligt: https://digitalalyftet.se/

vad-ar-javascript-och-vad-anvander-man-det-till/#:~:text=JavaScript%20Ãđr%20ett%20skriptsprÃěk%20som,och%20video%2C%20och%20mycket%20mer. (hämtad: 2025-04-18)

[12] Typescript, TypeScript: JavaScript with syntax for types", 2025. [Online]. Tillgänglig: https://www.typescriptlang.org (hämtad: 2025-02-05)

[13] React, React - A JavaScript library for building user interfaces", 2025. [Online]. Tillgänglig: https://react.dev (hämtad: 2025-02-05)

[14] Node.js, "Introduction Node.js", 2025. [Online] Tillgängligt: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs (hämtad: 2025-04-18)

[15] NPM, Äbout npm", 2025. [Online] Tillgängligt: https://docs.npmjs.com/about-npm/ (hämtad 2025-06-06)

[16] Express.js, Fast, unopinionated, minimalist web framework for Node.js", 2025. [Online] Tillgängligt: https://expressjs.com (hämtad: 2025-04-18)

[17] Microsoft, "Vad är en SQL-databas?", 2025. [Online] Tillgänglig: https://azure.microsoft.com/sv-se/resources/cloud-computing-dictionary/what-is-sql-database (hämtad: 2025-04-18)

[18] Microsoft, "Vad är PostgreSQL?", 2025. [Online] Tillgänglig: https://azure.microsoft.com/sv-se/resources/cloud-computing-dictionary/what-is-postgresql (hämtad: 2025-04-18)

[19] Kinsta, "Vad är Docker?", 2023. [Online]. Tillgänglig: https://kinsta.com/se/kunskapsbas/vad-ar-docker/ (hämtad: 2025-04-18)

[20] Google, Firebase Cloud Messaging", 2025. [Online]. Tillgänglig: https://firebase.google.com/docs/cloud-messaging (hämtad: 2025-02-05)

[21] Google, Firebase Authentication", 2025. [Online]. Tillgänglig: https://firebase.google.com/docs/auth (hämtad: 2025-02-05)

[22] Google. "Introduction to the Admin Auth API", 2025 [Online]. Tillgänglig: https://firebase.google.com/docs/auth/admin (hämtad: 2025-03-12)

[23] Google. Äuthenticate with Firebase using Password-Based Accounts using Javascript", 2025 [Online]. Tillgänglig: https://firebase.google.com/docs/auth/web/password-auth (hämtad: 2025-03-12)

[24] Google. "Verify ID Tokens", 2025 [Online]. Tillgänglig: https://firebase.google.com/docs/auth/admin/verify-id-tokens (hämtad: 2025-03-12)

[25] Google. Manage Session Cookies", 2025 [Online]. Tillgänglig: https://firebase.google.com/docs/cloud-messaging (hämtad: 2025-03-12)

[26] U. Malm. (2025) Trygghet i Frölunda och Tynnered 2022, Bostads AB Poseidon, Göteborg, Västra Götaland, Sverige, 2022 [Online]. Tillgänglig https://frolundatynnered.se/wp-content/uploads/2023/06/Trygghet-i-Frolunda-och-Tynnered-2022-l.pdf (hämtad: 2025-02-05)

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

CHALMERS