

# Slutrapport

## Sammanfattning

Kunskaper som jag har fått under arbetet är GraphQL, JSON Web Tokens [JWT] och fördjupad kunskap kring databashantering i praktiken. GraphQL användes som huvudsaklig kommunikationskanal mellan klienten och databasen vid hämtning och ändring av information. JWT användes för att dels begränsa vissa av GraphQL funktionaliteten till att bara vara tillgänglig om man var inloggad, samt för servern att hålla koll på inloggade användare. Den fördjupade kunskapen kring databashantering skedde till största del när mer komplicerade relationer skulle skapas i databasmodellen samt när befintlig data skulle migreras till den uppdaterade databasmodellen.

I produkten som har skapats är det möjligt att logga in eller skapa ett konto för att visa upp information om sig själv:

- Namn
- Kontaktuppgifter
- Tidigare erfarenheter
- Färdigheter

Till sina erfarenheter och färdigheter går det att lägga till referenser i form av länkar till videor eller texter, telefonnummer, mailadresser samt andra användare från sidan.

Det går även att söka bland befintliga användare på sidan. Det finns en bredare sökning där söktermer (fritext) matchas med användarens:

- namn eller användarnamn
- färdighet - namn eller beskrivning
- erfarenheter - namn eller beskrivning

Det finns även möjlighet för snävare sökningar där man kan hitta användare med specifika erfarenheter eller färdigheter.

Som användare kan man även visa intresse till andra användare, och även se hur många andra som är intresserade av användaren. Som användare går det att se vilka som är intresserade av en och besvara visat intresse genom svaren: att man själv är intresserad, inte intresserad eller likgiltig (standard). Det går även att se vilka man själv har visat intresse till samt svaret från användaren. Båda listorna med användare av intresse och intresserade användare sorteras i ordningen: likgiltig, intresserad, inte intresserad.

## Produktbeskrivning

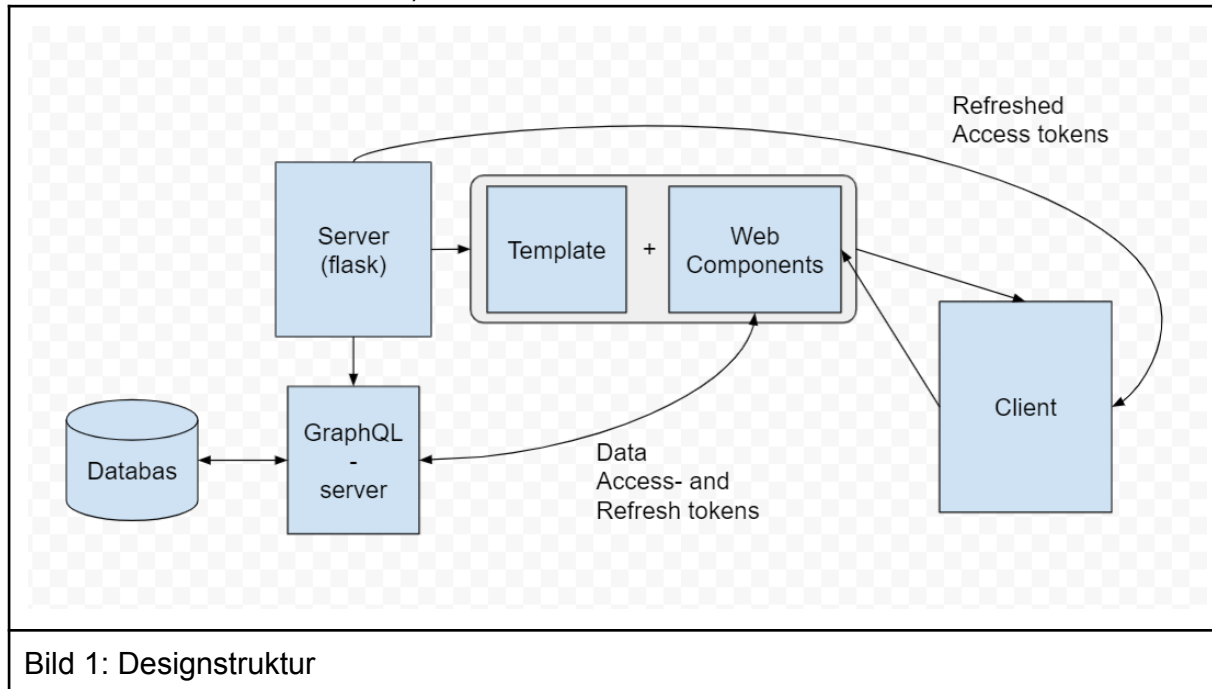
### *Designstruktur*

Servern skickar en template, dynamisk HTML, innehållande web-components som visas till klienten.

GraphQL hanterar logiken när det gäller hämtning, ändring och borttagning av information i databasen.

Web-component komponenterna hämtar information via GraphQL-servern som visas på klienten. Om klienten vill ta bort, eller ändra information skickar web-component komponenten den informationen till GraphQL-servern som hanterar klientens förfrågan och svara om den kunde hantera förfrågan eller inte.

När klienten loggar in eller skapar ett konto skickas access- och refresh tokens från graphQL till klienten som används för att göra specifika typer av förfrågan till servern (e.g. ta bort eller ändra information).



### *Databasmodell*

Staff innehåller information om själva användaren. Till en användare från staff-tabellen sparas:

- Lösenord som är krypterade
- färdigheter
- erfarenheter
- visat intresse

Till erfarenheter eller referenser sparas referenser.

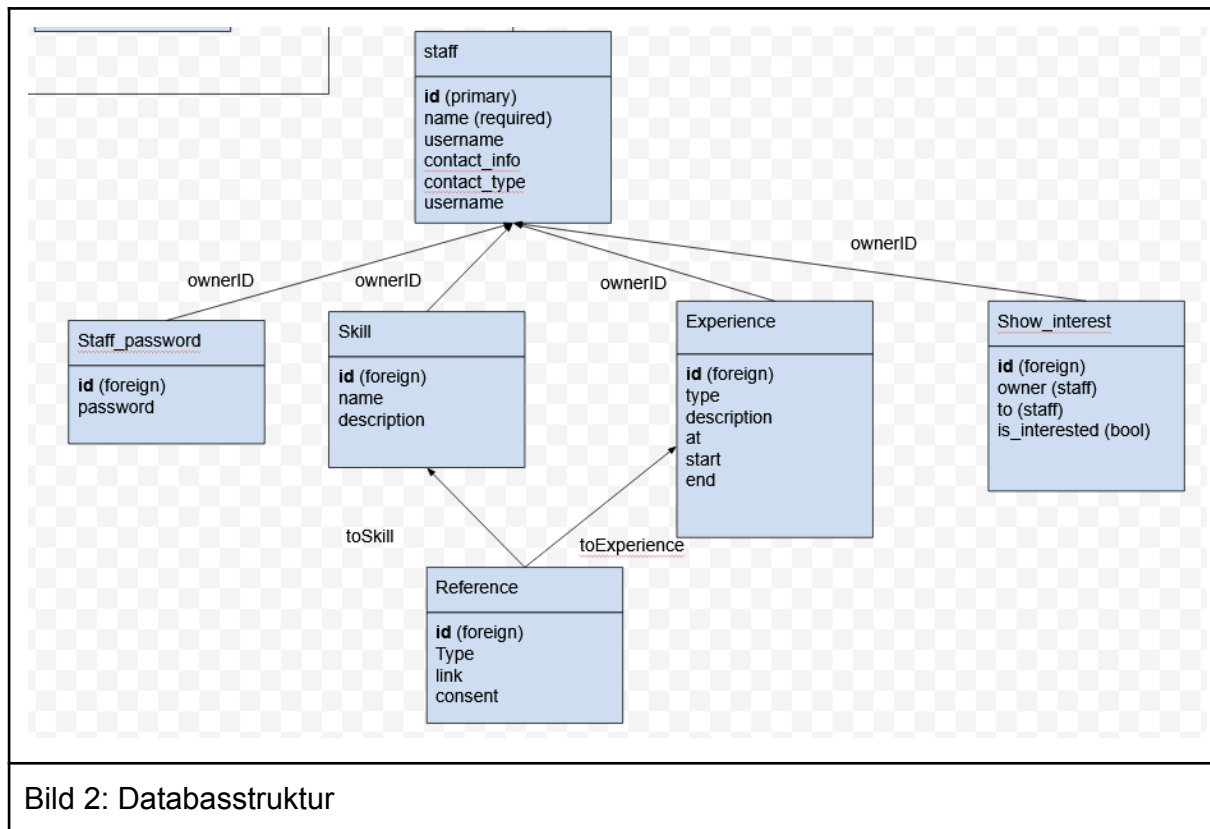


Bild 2: Databasstruktur

## Produkt

Den första tanken kring utseendet produkten blev i form av en prototyp. Tanken var att användar-sidans information ska delas upp i rutor i två spalter på sidan där användarens kontaktuppgifter står högst upp på sidan. Därefter skulle erfarenheter och färdigheter visas.

Startsidan skulle innehålla en rubrik med sidans namn och en knapp för att logga in eller skapa ett konto på sidan. På startsidan skulle det även vara möjligt att göra olika typer av sökningar bland användare.

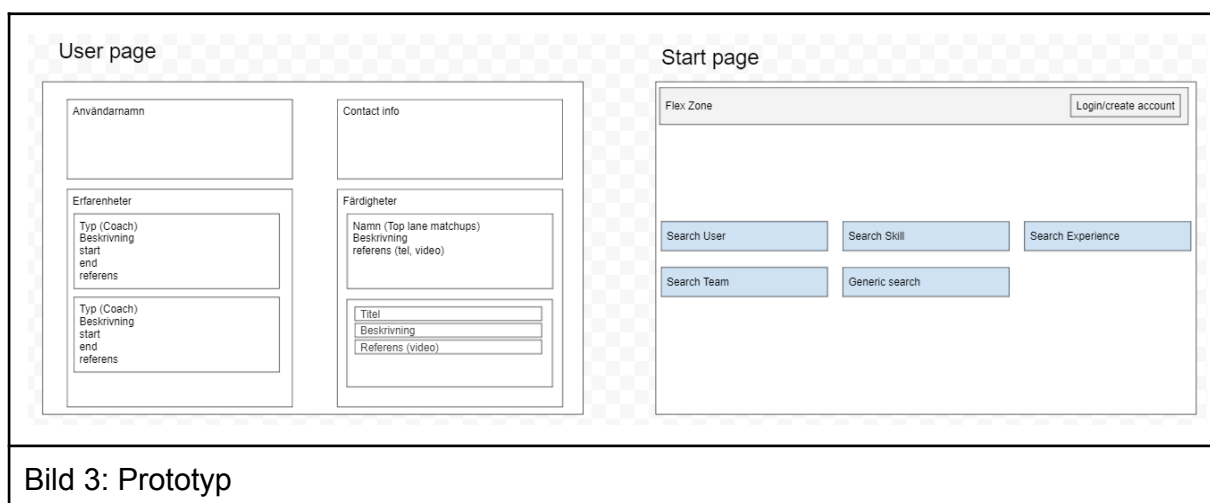


Bild 3: Prototyp

Det slutliga resultatet blev följande:

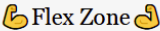
The screenshot shows the Flex Zone homepage. At the top left is the logo "Flex Zone" with a thumbs up icon. At the top right is a button "Log in | Create account". Below the logo is a "Search" section with three search boxes: "Generic Search" (placeholder: "e.g. Username or name"), "Experience Search" (placeholder: "Select Type"), and "Skill Search" (placeholder: "e.g. Pick ban or Top la"). Each search box has a "Submit" button and a "Search" button. At the bottom of the page is a cookie notice: "This site uses only functional cookies" with a thumbs up icon and a button "I understand".


Bild 4: Startside med olika typer av sökningar

The screenshot shows the Flex Zone search results page. At the top left is the logo "Flex Zone" with a thumbs up icon. At the top right is a button "pelle123" with a thumbs up icon and a notification bubble with the number "1". Below the logo is a "Search" section with three search boxes: "Generic Search" (placeholder: "e.g. Username or name"), "Experience Search" (placeholder: "Select Type"), and "Skill Search" (placeholder: "e.g. Pick ban or Top la"). Each search box has a "Submit" button and a "Search" button. The "Experience Search" dropdown menu is open, showing the following options: "coach", "analyst", "coach", "positional coach", "sports psychologist", and "other". Below the search boxes are two search results:

- [Lisa](#)  
Coach - Pepega Gaming  
Head coach, focus in team morale
- [Pelle](#)  
Assistant coach - Pepega Gaming  
Assistant coach, focus one to one relationships

Bild 5: Ett sökresultat från sökning bland erfarenheter





Lisa  
1234-5678-90 [phone]

Experiences

Coach

Pepega Gaming

Head coach, focus in team morale

2020-01-01 - 2021-01-01

References:

- External link
- External link
- User: test ?

Skills

Pick Ban

Good att analyzing och doing pick ban

References:

- External link

Bild 6: En profil-sida när man kollar på en annan användare



[Log in](#) | [Create account](#)

Create user

Name

\*

Username

\*

Password

\*

Contact Information

Select Contact Type

▼

Contact address (email, discord, phone ...)

\*

Cancel

Create User

[Back to startpage](#)

Bild 7: Skapa konto

Bild 8: Den egna profil-sidan som inloggad

Bild 9: Redigera konto som inloggad

### Alternativa lösningar

Istället för att använda web-components som ramverk för klient-sidan skulle andra ramverk kunna användas som React, Svelte eller Angular. Dessa ramverk valdes bort för att jag har utvecklat hemsidor i ramverken tidigare, men inte en som använder web-components. Fördelen med React, Svelte och Angular över web-components är att det är enklare att skicka information och funktioner mellan komponenter. Det hade resulterat i att komponenterna som bygger upp sidan skulle bli fler och mindre, vilket skulle underlätta utvecklingsprocessen, vidareutveckling och underhållning av sidan. Fördelen med web-components är att själva tekniken inte är beroende av något externt bibliotek utan finns som standard i JavaScript.

Vid utvecklingen av GraphQL servern skulle färre bibliotek användas. Ett problem som uppstod var att det var problematiskt att vidareutveckla funktionalitet när flera bibliotek användes. Det hade varit ett större arbete att definiera modeller till GraphQL som matchar databasen, vilket gjordes av ett bibliotek, men hade gjort det enklare att vidareutveckla funktionalitet senare i projektet.

Möjlighet att sortera information i GraphQL servern innan den skickas till klienten, istället för att sortera informationen i komponenten som hämtar informationen. Är ett följdproblem från tidigare stycke att det blev för problematiskt vid användning av för många olika bibliotek vid skapandet av GraphQL servern.

Sidan skulle kunna använt sig av Server Side Event [SSE] eller web sockets för att skicka information från servern till klienten vid uppdatering av databasen. Exempelvis när en användare visar intresse till en annan kan vyn uppdateras hos båda användarna. Sett i efterhand anser jag att det inte hade gjort så mycket för att öka produktens kvalitet i relation till utvecklingstid och påfrestning hos servern. Tanken med sidan är inte att sitta och vänta på notiser, så om användaren ser att någon har visat intresse direkt, eller lite senare gör ingen större skillnad.

## Produktkvalitet

Produkten är stabil vid användning, men informationen som användaren skickar till servern valideras inte gediget på servern. Den lilla validering som sker är dels hos klienten men även datatypen på informationen som sparas i databasen.

## Kunskapsanvändning

Förutom generell programmering (skriva, analysera) var följande kurser till nytta:

- *Databashantering för dataloger* var användbar vid skapandet av databasmodellen.
- *Interaktionsprogrammering och dynamiska webben* var användbar vid skapandet av applikationsarkitekturen och programmets designstruktur.

Andra informationskällor som har varit till användning är:

- Har tidigare undervisat i webbutveckling och webbserverprogrammering med Flask på gymnasie-nivå.
- Använt React vid utveckling av examensarbete
- Lyssnat på ett seminarium där olika designmönster med web-components diskuterades

## Framgångar

### *Tekniska framgångar*

Moment som gick speciellt bra vid utvecklingen av produkten:

- Att följa liknande struktur vid:
  - skapandet av komponenter med web-components
  - hantering av information som skickas med GraphQL
- Implementering av JWT för både hantering av inloggning och skydda vissa anrop till GraphQL servern.
- Sökningen av användare genom GraphQL
- Skapandet av relationer i databasen
  - Referenser till antingen erfarenheter eller färdigheter
  - Visat intresse som binder ihop två olika användare

### *Projektarbetet*

Specifika glädjeämnen vid utvecklingen av projektet var när:

- GraphQL kom igång och det gick att börja söka information för första gången.
- jag förstod hur man kunde använda mutations i GraphQL för att manipulera information i databasen.
- JWT fungerade bra för att logga in samt skydda anrop till GraphQL.



## Problem

### Tekniska problem

Det uppstod svårigheter att skicka information och funktioner mellan komponenter vid användning av web-components. Problemet löstes genom att slå ihop de komponenter som behövde kommunicera med varandra och i framtiden fundera kring vilka delar av sidan som behöver kommunicera och inte.

Det uppstod även problem när referens-komponenten utvecklades, specifikt vid redigering av referens eller färdighet eller erfarenhet. Själva problemet grundade sig i hur själva processen som användaren skulle ta sig genom för att enkelt redigera erfarenheten/färdigheten med referenser. Jag ville även undvika att för många requests skulle skickas till servern i onödan. Problemet löste sig när jag kom på och bestämde mig för att dela upp komponenterna så användaren redigerade en i taget. Alltså fick användaren bestämma om erfarenheten/färdigheten ska redigeras eller om en referens ska redigeras (se bild 8).

Det uppstod även problem när ett bibliotek för GraphQL förbestämda attribut skrevs över (id) vilket skapade stor förvirring under utvecklandet av GraphQL servern. Det löstes först genom att en användares användarnamn användes som identifierare, eftersom id inte matchade. Sedan när källan till problemet visade sig ändrades databasmodellen så det inte blev konflikt mellan den skriva koden och biblioteket.

Det uppstod problem när data skulle sorteras på GraphQL servern innan den skickas till klienten. Som tidigare benämnt mynnade sig problemet dels i användning av för många bibliotek, men även att undersökningstiden sprang iväg och kunde inte rättfärdigas när den alternativa lösningen tog 10 minuter att implementera. Den alternativa lösningen var att sortera informationen i komponenten hos klienten.

### *Projektarbetsproblem*

Det har inte uppstått några direkta problem i projektet. Det har flutit på som planerat, förutom att allt tog längre tid än specificerat i projektspecifikationen (som fick feedback att arbetet skulle ta för kort tid för kursen).

### Arbetsplan

Nästan alla mål från projektspecifikationen har mötts, även målen för vidareutveckling. Den betydande avvikelser från planen var att det inte implementerades någon funktionalitet att skapa ett konto eller logga in som ett lag. Alla användare behandlas som "vanliga" användare. Det är därför användarna benämns som "staff" i databasmodellen, för tanken var att det skulle även finnas "team" (se bild 2).

## Sluttidrapport

Del	Tid [h]
Setup flask	5
Design databas (analyst/coach)	5
Få GraphQL att fungera	14
Första prototyp	5
komponenter för användarvyn	6
Undersöka graphql	6
Komponenter för att skapa färdigheter	8
Komponenter för att skapa efrarenheter	6
Uppdatera databasmodell och berörda komponenter (staff)	5
Uppdatera databasmodell och berörda komponenter (erfarenheter)	5
Rita ikon för tabb i webbläsaren	3
Skapat skapa användare komponent	5
Skapat mer dummy-data i databasen	5
skapat en route och sida för skapa användare med återanvändning av komponenter	5
Söka bland noder i GraphQL	7
Hämta data från GraphQL för att visa i komponenter	8
Planera startsidan (prototyp)	5
Lagt till tabell för lösenord i databasen	8
Uppdaterat skapa användare komponent	2
Undersöka mutations i GraphQL	15
Information sparas i databas vid skapande av konto	7
använda användares username istället för id i url på användarsidan	5
gå till graphql mutations istället för routes med form för att spara information från användare	8
information skickas från klient (genom graphql mutations) när den skapar användare, skills och experiences som visas för användaren och sparas i databas	13

fixat till utseende på logga in/skapa konto komponent	4
Feedback vid skapandet av användare om username redan finns	5
ta bort användare	10
ta bort Färdighet/erfarenhet (problem med bibliotek när jag tidigare skrev över standard-attribut)	15
Ändra erfarenheter (samma för färdigheter)	10
Lägg till edit skill (samma flöde som edit experience)	5
GraphQL Mutation för att uppdatera lösenord	5
vy för att uppdatera lösenord	6
Vy för att uppdatera användare	7
Logga in/skapa konto uppdaterad vy	5
undersöka bibliotek för inloggning med graphql	5
implementerat så vissa graphql mutations går endast att använda om man är inloggad	5
skydda routes (måste vara inloggad)	5
komponenter har olika utseende om man är inloggad eller inte	6
Fixat med access tokens och refresh tokens	4
Skapat search komponent med generell sökning	10
lagt till Reference i databas	2
fixat graphql mutations för Reference	2
edit/delete referenser (svårt att hitta ett bra flöde för att göra det på ett bra sätt)	20
utveckla search (färdigheter och erfarenheter)	2
Cookie bottom bar	4
Föreslår användare vid skapandet/redigering av referens som refererar till användare	16
validera email-address	1.5
påbörjan till responsivitet	2
Skapat möjlighet att spara om man är intresserad av en annan användare	2

skapat koppling i databas så en användare kan se vilka som är intresserade av en (interestees) och vilka man är intresserad av (interests)	5
skapat graphql Mutation för att skapa, ta bort och uppdatera intressen	2
vy för att visa/ge intresse	15
Visar hur många som är intresserad av en	4
undersöka sortering av graphql	10
Skapa en error-sida	2
para information om man väljer att ens kontakt-typ är "annan"	2
mobil-anpassade knappar (inte hover med css längre)	5
Undersökt sätt att versionhantera/uppdatera databas utan att behöva ta bort befintliga användare	6
användare svara om någon ger dem som referens	8
snygga till skapa/uppdatera användare formulär	8
småfix	4
Skriva rapport	4

*Total under projektet*

Projektmedlem	Tot
Jens Berntsen	393