Slutrapport

Sammanfattning

<Beskriv kortfattat resultatet av ert arbete: kunskaper ni fått, produkten ni skapat.>

Kunskaper jag fått:

- GraphQL
- JWT

Produkten som har skapats:

- Möjlighet att logga in för att skapa, ändra och visa upp sina:
    - namn, kontaktuppgifter
    - Tidigare erfarenheter
        - Med referenser
    - Sina färdigheter
        - Med referenser
- Möjlighet att söka bland befintliga användare
    - Bred sökning
        - Sök efter termer i namn, användarnamn, erfarenheter, färdigheter
    - Snäv sökning
        - Bland bestämda termer i erfarenheter
            - Samma termer som när användare “skapar” erfarenheten
        - Fri sökning bland färdigheter
- Möjlighet att visa intresse till andra användare
    - Om en användare har fått intresse kan användaren svara:
        - Intresserad
        - Inte intresserad
        - likgiltig (default)

      Och svaret visas till den användare som skickade intresset
    - Om man visar intresse, se hur många andra som är intresserade av samma användare
-

Produktbeskrivning

<Beskriv produkten ni skapat. Bifoga gärna skärmdumpar, beskriv designstruktur osv.>

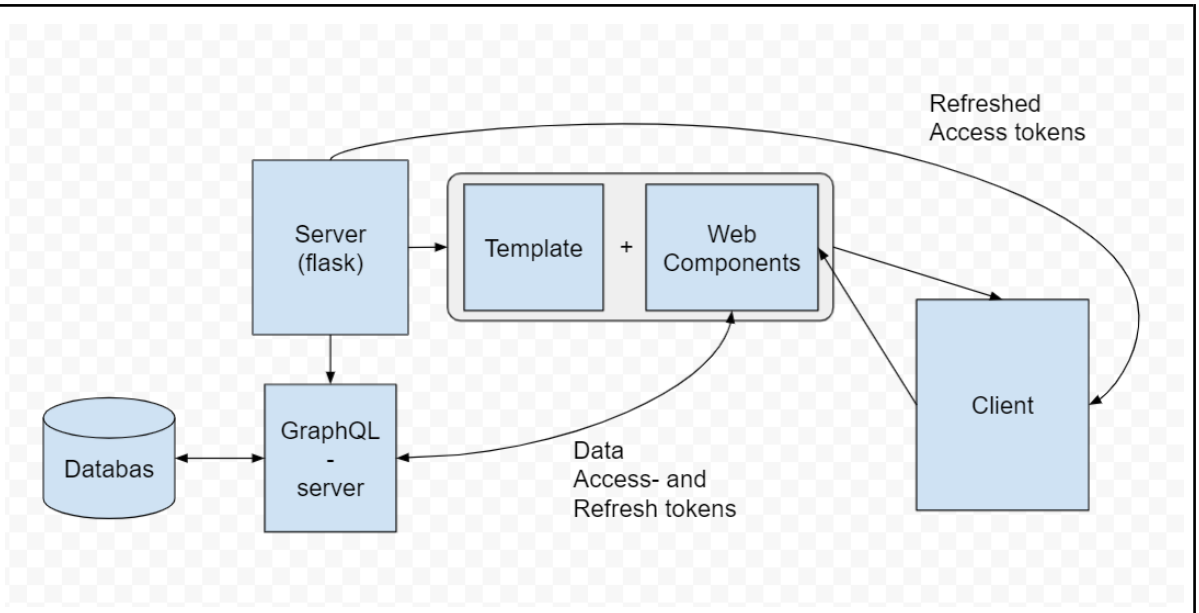

Bild 1: Designstruktur

- Server skickar template(html) och web-components till klienten (själva sidan)
- Web komponenten kan hämta och skicka information till graphQL servern
- Klienten kan be web-komponenten hämta/skicka information (logga in, söka, skapa, ändra, ta bort)
- GraphQL servern hanterar hämtning/ändring av information i databasen.
- När klienten loggar in skickas även tokens för att hantera inloggningen som kan refreshas av servern.

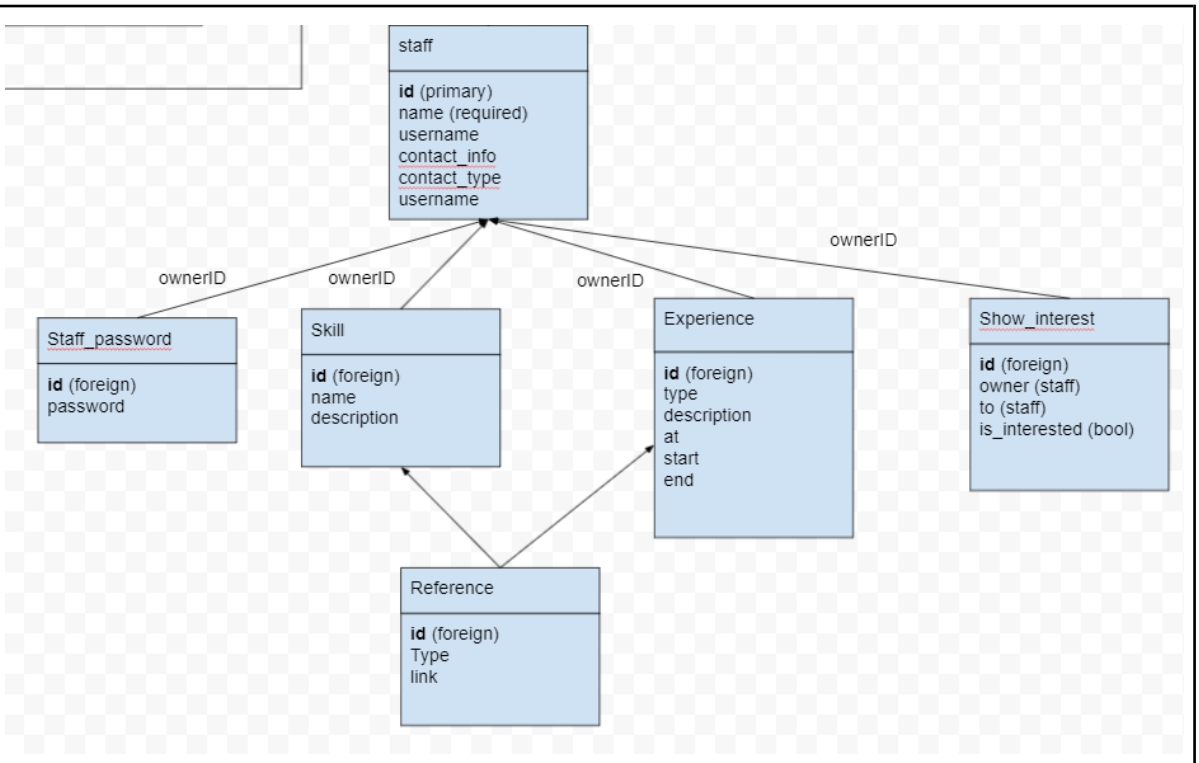

Bild 2: Databasstruktur

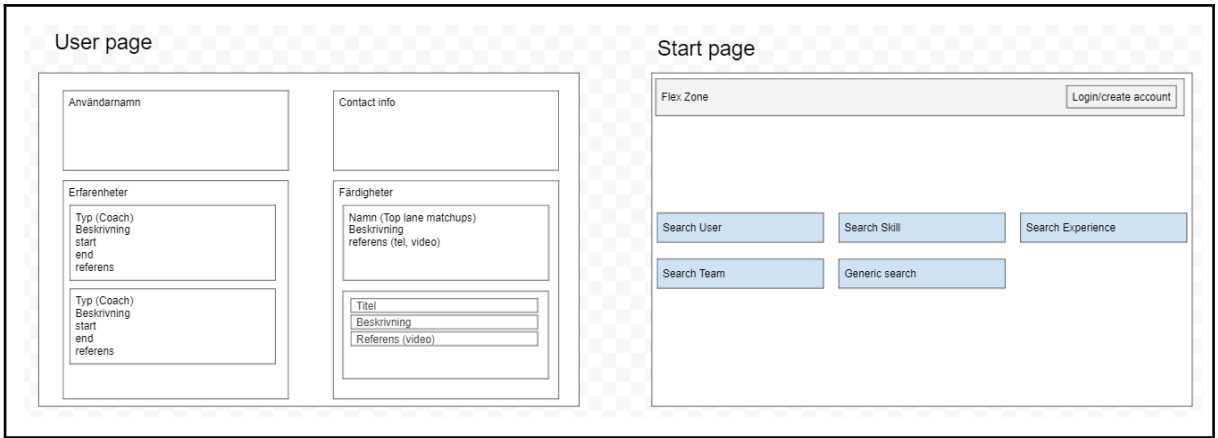

Bild 3: Prototyp

<Kommentera eventuella alternativa lösningar. Varför valdes de bort under arbetets gång? Sett i efterhand?>
Alternativa lösningar:

- Använda annat bibliotek/ramverk för frontend
    - Blir enklare att dela upp i ännu mindre komponenter
    - Valdes bort för att fördjupa kunskap kring nya bibliotek/ramverk (web components)
- Använda färre befintliga bibliotek för GraphQL, försöka implementera lösningar där bibliotek inte nödvändigtvis inte behövs.
    - Exempelvis: graphene_sqlalchemy
        - Definiera egna object-types som matchar databasen
- Använda sig av Server event[SSE] eller Web sockets för att skicka information till klienten om någon har visat intresse av ens profil
    - Valdes bort för att tid/snabb respons inte är så viktig på sidan.
    - Ingen annan del av sidan som måste använda sig av SSE/web sockets för att skicka information.

Produktkvalitet

<Är slutprodukten i er mening stabil eller finns kända brister?>

Kunskapsanvändning

<Kommentera använd kunskap från föregående kurser. Vad har ni speciellt haft nytta av för moment?>

- Databashantering (Databashantering för dataloger)
- Applikationsarkitektur (Interaktionsprogrammering och dynamiska webben)

<Kommentera eventuella andra informationskällor>

- Undervisat i webbserverprogrammering
    - Flask

- Använde React i examensarbete
- Gått på seminarium om web components

Framgångar

Tekniska framgångar

<Kommentera vad som har gått bra>

- Följa liknande arbetsstruktur vid skapandet av
  - Komponenter
  - Hantering av information med GraphQL
    - Queries
    - Mutations
- Implementering av inloggningen
  - Smidigt att lägga till JWT för att skydda routes och queries i GraphQL
- Sökningen genom GraphQL
  - Enkelt att göra snäva och breda sökningar
- Skapa relationer i databasen som fungerar bra med GraphQL
  - Referenser → Erfarenheter eller Färdigheter
  - Intresse binder ihop två användare(staff)

Projektarbetet

<Kommentera glädjeämnen i projektarbetet>

- När GraphQL kom igång och det gick att börja söka information
- När jag förstod hur man kunde använda mutations i GraphQL för att manipulera information i databasen.
- När JWT fungerade bra för att skydda routes och queries till GraphQL

Problem

Tekniska problem

<Kommentera ev. problem och hur ni löst dessa>

- Svårigheter att “skicka” metoder mellan komponenter som behöver kommunicera.
  - E.g. Skill och CreateSkill komponent
    - När en färdighet skapas i CreateSkill komponenten ska det kommuniceras till Skill komponenten så den nya färdigheten kan visas på skärmen (inte behöva skicka event mellan server och klient)
  - Komponenterna lades ihop, vilket även underlättade utvecklingen av att ändra på befintliga färdigheter (mycket kod kunde återanvändas).
  - Vissa komponenter får bli mer självständiga (referens med färdighet/erfarenhet)
    - Referens-komponenten behandlar skapandet, redigeringen och borttagning som inte påverkar/beror på skapandet/redigering av erfarenhet/färdighet
- Problem med hantering av id i databas och dess hämtade objekt
  - Började till en början använda användarens användarnamn (som är unika) som identifierare

- Visade sig senare när det uppstod problem vid hantering av färdigheter och erfarenheter att jag skrev över reserverade attribut (.id) som används av biblioteket. Löstes genom gedigen googling, där jag tillslut hittade någon som hade stött på liknande problem som jag hade.

Projektarbetsproblem

<Kommentera ev. problem och hur ni löst dessa. >

Arbetsplan

<Har ni nått ert mål?>
<Kommentera betydande avvikelser från plan>

Sluttidrapport

<Rapportera totalt antal timmar nedlagda på kursen>

Total under projektet

Projektmedlem	**Tot**