Microservices, Kong Arthur

# Overblikk:

Eksempelet består av 4 komponenter, der 2 av dem er (så og si) det som utgjør dette rammeverket:

* api: et eksempel API, som håndterer autorisering og returnerer bl.a. brukere, adresser og grupper, som ligger i databasen
* auth: komponenten som tar seg av autentisering. Her er det også mulig å implementere autorisering i samme modul, men slik den er satt opp nå tar den seg kun av autorisering av en bruker. Her er det også lagt inn støtte for å registrere nye brukere og logge inn (hente ut jwt for) en bruker
* proxy: dette er en nginx server som tar seg av alle requests inn til systemet. Det fine med denne er at den (her) er konfigurert til å gjøre et subrequest til auth-serveren dersom en klient forsøker å aksessere api-et
* webapp: en eksempel webapp, som kan brukes for å illustrere bruken av systemet

# Proxy:

Proxyen er en enkel nginx-server med følgende endepunkt:

* /auth: brukes for å autentisere en bruker som forsøker å aksessere et annet endepunkt (her: /api). Har også støtte for å registrere en bruker og logge inn en bruker
  + /auth/me: autentiserer en bruker, ved at brukeren sender inn header Authorization med et Bearer token
  + /auth/login: logger inne en bruker, som sender inn brukernavn/email og password og returnerer et Bearer token
  + /auth/register: lager en ny bruker med navn, brukernavn, passord, email, osv. Bearer token blir returnert
* /api: brukes for å aksessere data som ligger lagret i databasen (users, groups, addresses)
  + /api/addresses: henter ut data om adresser i Oppland
  + /api/users: henter ut data om alle brukere som er registrert (brukes også for autentisering i auth-endepunktet)
  + /api/groups: henter ut data om alle brukergrupper
* /: brukes for å aksessere serveren som webappen ligger på
  + /addresses: visualiserer alle adresser i en tabell (krever innlogging og gyldig Bearer token for alle api-kall fra webapp)

Det som er fint med en proxy-server er at absolutt all trafikk til alle endepunkt i systemet går gjennom denne. Her kan man konfigurere autentisering til de endepunktene man ønsker, vha. auth\_request til en service som tar seg av autentisering. Når denne returnerer 2.. -status er brukeren gitt tilgang.

Auth-servicen legger til en header «x-user» som brukes for å autorisere brukeren i api-et. Her kan det være fornuftig å legge både autentisering og autorisering sammen med API-et, i og med at autoriseringen i API-et er avhengig av at headeren x-user blir satt i auth-servicen. Dette er kun for å vise litt hvordan man kan bruke en autoriserings-service sammen med nginx

# API:

Api-et er en enkel express server som aksepterere følgende endepunkt:

* /api/addresses: henter ut data om adresser i Oppland
* /api/users: henter ut data om alle brukere som er registrert (brukes også for autentisering i auth-endepunktet)
* /api/groups: henter ut data om alle brukergrupper

Her har jeg eksperimentert litt med både promises og middleware. En middelware som heter getGroups (kalles i routes/users.js) finner headeren «x-user», henter ut alle gruppene som brukeren er registrert med og setter en property «groups» i request-objektet. Her kan det oppstå problemer dersom x-user ikke er satt i auth-servicen og det kan være hensiktsmessig igjen å bake autentiseringen inn i api-servicen. På den annen side så ønsker man jo alltid autentisering før autorisering, men man er altså avhengig av at autentiserings-servicen er konfigurert i nginx ved endepunkt som krever autorisering.

# Auth:

En express-server som utelukkende håndterer autentisering. Den her settes en header x-user, med id til brukeren som er blitt autentisert og som kan brukes i API-et. Følgende endepunkt er implementert:

* /auth/me: autentiserer en bruker, ved at brukeren sender inn header Authorization med et Bearer token
* /auth/login: logger inne en bruker, som sender inn brukernavn/email og password og returnerer et Bearer token
* /auth/register: lager en ny bruker med navn, brukernavn, passord, email, osv. Bearer token blir returnert

Det kan være tilfeller der man ønsker å skille autentiseringen av en bruker fra resten av servicene, f.eks. dersom man ønsker å kontrollere aksesseringen til fulle systemer med flere microservices som kjører. Man kan også velge å gjennomføre autorisering i samme service, men kan være litt vanskeligere å kontrollere hvert enkelt endepunkt. Slik det er satt opp nå trenger man bare å legge til et funksjonskall i hvert endepunkt i API-et, som sjekker om brukeren er med i ei gruppe som har tilgang til endepunktet. Dersom man ønsker forskjellig autentisering i forskjellige endepunkt kan dette enkelt legges til i nginx, som en auth\_request (en subrequest som utføres før det opprinnelige kallet.

Jeg har lært utrolig mye ved å jobbe med dette prosjektet og det er på ingen måte noe jeg ser på som en fasit, men kan gi en pekepin på hvilke muligheter det er når det kommer til microservices, nginx og express middleware.