Programmering og udvikling af små systemer samt databaser

Godkendelsesopgave 3

Victoria Skjøren

Dette APi´et bruker programvaren npm, som importerer pakken express og jsonwebtoken, og kjører i runtime miljøet node.js. Inneholder følgende filer:
endpoints.js, script.js, usersEndpoints.js, interestsEndpoints.js, matchesEndpoints.js, private.pem og .gitignore.

Fil: endpoints.js

I denne filen har jeg samlet alle endpoints til programmet, her ligger også funksjonen isAuthorized.

```
JS endpoints.js > ...
1
2   const express = require ("express")
3   const app = express()
4   const port = 3001
5   const fs = require("fs")
6   const jwt = require("jsonwebtoken")
7   const {getUsers, deleteUser, createUser, getImage, getCreditCardInfo} = require ("./usersEndpoints")
8   const {getInterests, deleteInterest, createInterest} = require("./interestsEndpoints")
9   const {getMatches, deleteMatch, newMatch} = require("./matchesEndpoint")
10
11
12
13  app.get("/users", isAuthorized, getUsers )
14  app.delete("/users/:userID", isAuthorized, deleteUser )
15  app.post("/users", isAuthorized, createUser )
16  app.get("/users/:userID/images", isAuthorized, getImage )
17  app.get("/users/:userID/creditcard", isAuthorized, getCreditCardInfo )
18
19  app.get("/users/:userID/interests", isAuthorized, getInterests )
20  app.delete("/users/:userID/interests/:interest", isAuthorized, deleteInterest )
21  app.post("/users/:userID/interests", isAuthorized, createInterest )
22
23  app.get("/users/:userID/matches", isAuthorized, getMatches )
24  app.delete("/users/:userID/matches/:match", isAuthorized, deleteMatch )
25  app.post("/users/:userID/matches", isAuthorized, newMatch )
```

Jeg startet med å lage de tre CRUD-endpoints for User, Interest og Match, i tillegg til getImage og getCreditCard. Jeg har valgt å legge de tre hoved endpoints i tre ulike filer for å få en bedre oversikt. Disse har jeg kalt usersEndpoints, interestsEndpoints og matchesEndpoints.

```
app.get("/jwt", (req,res) => {
    let privateKey = fs.readFileSync("./private.pem", "utf8");
    let token = jwt.sign({"body": "stuff"}, privateKey, {algorithm: "HS256"});
    res.send(token);
})


function isAuthorized ( req, res, next ){
    if (typeof req.headers.authorization !== "undefined"){
        let token = req.headers.authorization.split(" ")[1];
        let privateKey  = fs.readFileSync("./private.pem", "utf8");

        jwt.verify(token, privateKey, {algorithm: "HS256"}, (err, decoded) =>{
            if (err){
                res.status(401).json({error : "Not authorized"})
            }
            console.log(decoded);
            next();
        })
    } else {
        return res.status(401).json({error : "Not authorized"})
    }
}


app.listen(port,
    () => console.log(`Listening on port ${port} `));
```

Login-funksjonen isAuthorized lagde jeg ved å følge denne tutorial: https://tutorialedge.net/nodejs/nodejs-jwt-authentication-tuto rial/.
Funksjonen blir kalt på før get/delete/post funksjonen ved hvert endpoint for at brukeren skal verifiseres før de får tilgang til programmet. Endpointet /jwt lager en private key, som brukeren kan ta i bruk.

Fil: script.js

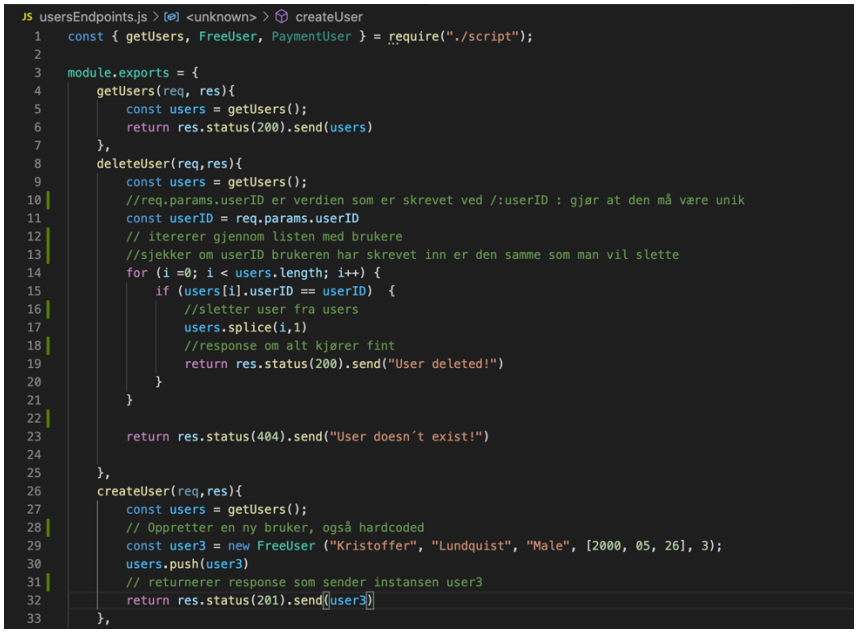
Filen script.js fungerer litt som en database med hardcodet informasjon om users. Ettersom alle verdier er hardcodet lagres ikke endringene i filen. Her ligger klassen User, PaymentUser og FreeUser som ved nedarving arver metodene og parameterene fra User. I tillegg ligger klassene Interests, Match, Images og CreditCard her. Jeg har også lagt til noen metoder som ikke blir kalt på i dette programmet, men som kan være nyttig om man skal videreutvikle i en senere tid.

```
module.exports = {
    getUsers() {
        return users;
    },
    getInterests(){
        return interests;
    },

    getMatches(){
        return matches;
    },
    getImage(){
        return images;
    },
    getCreditCardInfo(){
        return creditCards;

    },


    FreeUser: FreeUser,
    PaymentUser: PaymentUser,
    CreditCard: CreditCard,
    Images: Images,
    Interests: Interests,
    Match: Match
}
```

For å kunne kalle på de ulike funksjonene i endpoints.js oppretter og eksporterer jeg funksjonene og returnerer kun arrays med informasjonen jeg vil ha tak i. I tillegg eksporterer jeg klassene for å kunne bruke de også.

Fil: usersEndpoints.js

Denne filen eksporterer følgende funksjoner: getUsers, deleteUser, createUser, getImage og get creditCardInfo som alle returnerer en respone. FIlen importerer verdier fra «databasen» script.js.



```
JS usersEndpoints.js > [@] <unknown> > createUser
1   const { getUsers, FreeUser, PaymentUser } = require("./script");
2
3   module.exports = {
4       getUsers(req, res){
5           const users = getUsers();
6           return res.status(200).send(users)
7       },
8       deleteUser(req,res){
9           const users = getUsers();
10          //req.params.userID er verdien som er skrevet ved /:userID : gjør at den må være unik
11          const userID = req.params.userID
12          // itererer gjennom listen med brukere
13          //sjekker om userID brukeren har skrevet inn er den samme som man vil slette
14          for (i =0; i < users.length; i++) {
15              if (users[i].userID == userID)  {
16                  //sletter user fra users
17                  users.splice(i,1)
18                  //response om alt kjører fint
19                  return res.status(200).send("User deleted!")
20              }
21          }
22
23          return res.status(404).send("User doesn´t exist!")
24
25      },
26      createUser(req,res){
27          const users = getUsers();
28          // Oppretter en ny bruker, også hardcoded
29          const user3 = new FreeUser ("Kristoffer", "Lundquist", "Male", [2000, 05, 26], 3);
30          users.push(user3)
31          // returnerer response som sender instansen user3
32          return res.status(201).send(user3)
33      },
```

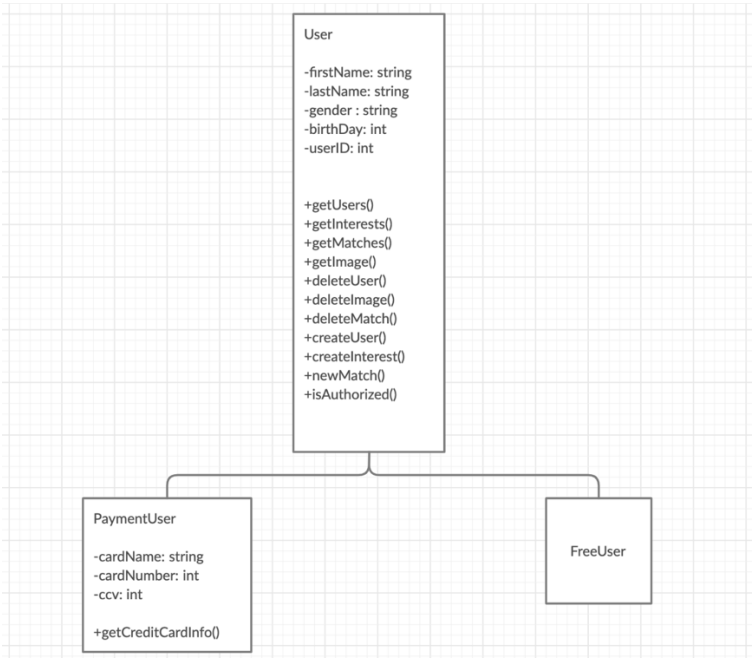
Fil: interestsEndpoints.js og matchesEndpoint.js

Disse filene er svært like bare referer til ulike verdier og derfor responderer også med ulike verdier. Det er i disse filene interesser eller matches faktisk blir lagd eller slettet, eller sender en liste over de aktuelle verdiene.

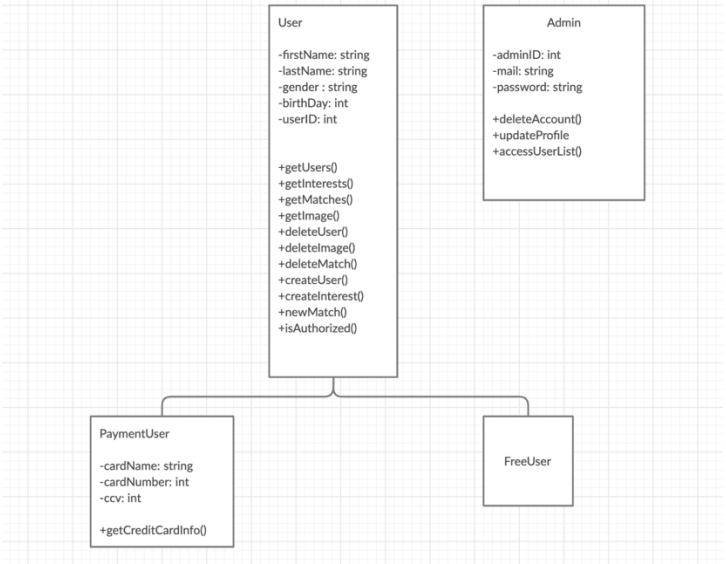
```
JS InterestsEndpoints.js > [@] <unknown> > deleteInterest
1
2   const { getUsers, getInterests, Interests  } = require("./script");
3
4   module.exports = {
5       getInterests(req, res){
6           const users = getUsers();
7           const userID = req.params.userID
8           for (i =0; i < users.length; i++) {
9               //sjekker om man har riktig user
10              if (users[i].userID == userID)  {
11                  //returnerer alle brukerens eksisterende interesser
12                  return res.status(200).send(users[i].interests);
13              }
14          }
15          return res.status(404).send("User doesn´t exist!")
```

```
JS matchesEndpoint.js > [@] <unknown> > deleteMatch
1
2   const { getUsers, getMatches, Match} = require("./script");
3
4   module.exports = {
5       getMatches(req, res){
6           const users = getUsers();
7           const userID = req.params.userID
8           for (i =0; i < users.length; i++) {
9               //sjekker om man har riktig user
10              if (users[i].userID == userID)  {
11                  //returnerer alle brukerens eksisterende mathcer
12                  return res.status(200).send(users[i].matches);
13              }
14          }
15          return res.status(404).send("User doesn´t exist!")
```

UML class chart over dette programmet



Mulig UML chart over et fremtidig program med en admin



Link til github med commits:
https://github.com/victoriaskjoren/godkjennelsesoppgave-3