# Godkendelsesopgave 3

Uddannelse: HA (it)

Course: Programmering og udvikling af små systemer samt databaser

Underviser: Henrik Thorn

Afleveringsdato: 18-10-2020

Udarbejdet af: Tobias Emil Birch. Studienummer: 146147

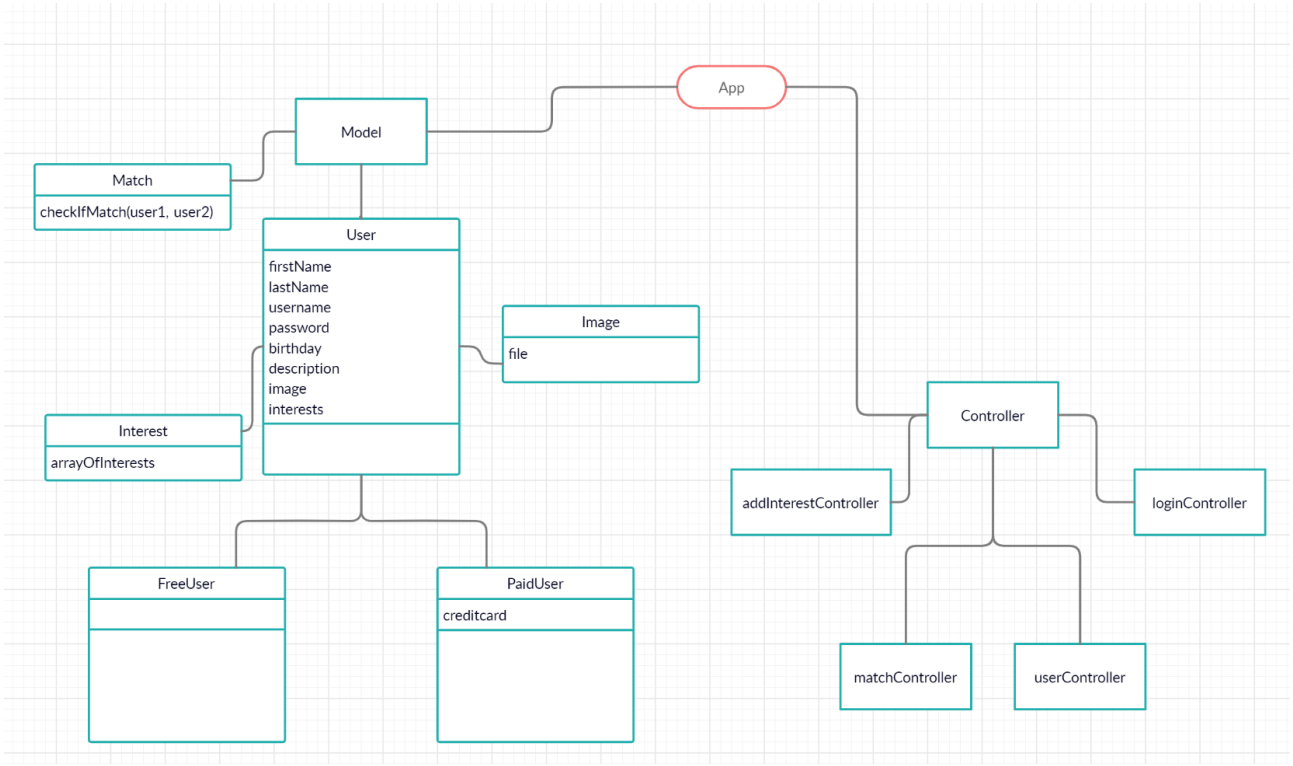
Antal anslag: 5046 - Antal sider: 3 (forside ekskluderet)

# 1. Beskrivelse af opgaven

I denne opgave har jeg udarbejdet en prototype til en backend til den datingapp, som vi skal udvikle til vores eksamensprojekt. Der er ikke udarbejdet nogen frontend, og det er altså derfor udelukkende design og implementering af API'et som er udarbejdet.

# 2. UML-klassediagram og overvejelser bag

For at danne et overblik over opgaven lavede jeg et UML-klasse diagram til at starte med. Dette hjalp med at se hvilke klasser, der skulle være og hvilke attributter de forskellige klasser hver især skulle have. Jeg har dog ikke haft så stor vægt på at udarbejde alle klasserne til en grad, hvor de er klar til den endelige app, da fokus generelt har været mere på API og prøve at få en forståelse for, hvordan dette skal fungere. Mit UML-klassediagram kan ses her:




*Figur 1: UML-klassediagram for Dating-App.*

Som nævnt ovenfor har min arbejdstid ikke ligget på de enkelte klasser, og har nogle klasser på nuværende tidspunkt ikke rigtig nogen funktion. Dette gælder fx for *Image*-klassen, der bare er en klasse som har en attribut, som i min prototype bare er en string, som jeg har kaldt ”image.png”. Til eksamensopgaven vil dette selvfølgelig skulle være en fil, som brugeren fx selv uploader ved oprettelse af profil.

Med hensyn til *Interest*-klassen har jeg været lidt i tvivl om, hvordan jeg helt skulle forstå den. Skulle det være en klasse der repræsenterer én enkelt interesse, eller skulle det være en klasse, som repræsenterer alle de interesser en bruger har? Jeg valgte at gå med den sidste løsning. Dvs. *Interest* klassen har en attribut, som er et array med forskellige interesser. Dvs. når man opretter en ny bruger, så har den en attribut, der hedder *interests*, som er et objekt af typen *Interest*. Derudover har jeg også nogle overvejelser omkring min Match klasse, som jeg egentlig ikke er helt afklaret med på nuværende tidspunkt. Den måde jeg lige nu intenderer at bruge klassen på, er ved at oprette et Match objekt, når en bruger liker en anden. Match-klassen indeholder nemlig en metode, som kan tjekke om to brugere har nogen fællesinteresser. Jeg har så i tankerne senere hen i programmet, at afhængig af metodens resultat skal en bruger så blive tilføjet til en ”likedList”, hvis de har fællesinteresser. Dette er ikke en del af den nuværende prototype.

På mit UML-klassediagram har jeg ikke inkluderet min server nogen steder, da jeg var i tvivl, om den hørte til der. Jeg besluttede at udelade den, da serveren teknisk set ikke er en klasse.

## 3. MVC

Jeg har prøvet at opbygge mit projekt efter MVC-modellen efter bedste evne på nuværende tidspunkt. Det er gjort ud fra den korte introduktion, vi har fået til MVC i øvelsestimerne og noget lidt research på nettet. Formålet med MVC, er at adskille ens dele i programmet. Dvs. modellen indeholder ens data, controlleren søger for at brugerinteraktion bliver håndteret og viewet er interfacet til brugeren. I denne opgave skulle vi udelade viewet, og jeg har lavet en model- og controllerdel. Jeg har lageret mine klasser i modeldelen, og de forskellige brugerinteraktioner i controllerdelen.
Controllerne anvender data fra nogle brugere, som jeg har hard-coded i programmet. Fordelene ved at hard-code data er, at det er nemt at teste, da ens data vil være det samme hver gang, og

man nemt kan gå ind og ændre det til nogle andre informationer, hvis man ønsker det. Derudover kan man også gå udenom behovet for at have en interface, når man skal teste.

Jeg har fire controllere i min app på nuværende tidspunkt. Den ene controller, *loginController*, anvender JSON Web Token, som er til for at validere en bruger. Den bruges på nuværende kun til, at når en bruger succesfuldt udfylder sine loginoplysninger, så bliver der oprettet en token, som er repræsentativ for denne bruger. Jeg har oprettet nogle services som kræver denne token endnu, da jeg ikke føler, at jeg er helt indforstået med, hvordan denne token helt præcis skal bruges. Dette vil jeg prøve at undersøge nærmere, når der skal arbejdes yderlige på projektet.

## 4. Test

I løbet af udarbejdelsen af programmet har jeg løbende brugt *console.log()* funktionen til at teste om min kode kørte som intenderet. Hvis der har været fejl, så har debugget og anvendt debuggeren til dette nogle af gangene.

Til at teste mit API har jeg brugt Postman, til at lave requests til min server. Jeg har brugt Postman, da vi ikke har skulle lave nogen frontend. Med Postman kan man nemt lave requests, som man ellers ville skulle bruge en frontend til.

Min testning har ikke været helt systematiseret og derfor er der sandsynligvis chance for fejl, som jeg ikke har opdaget.

## 5. Afsluttende bemærkninger og overvejelser

Jeg har til tider følt, at det har været lidt kontraintuitivt ikke at have nogen frontend, da jeg har haft svært ved at se den hele sammenhæng i appen pga. dette. Derfor har det givet nogle enkelte bump på vejen gennem udarbejdelsen af programmet. Jeg håber, at min forståelse bliver bedre, når der bliver koblet en frontend og database på, så det hele begynder at gå op i en højere helhed.