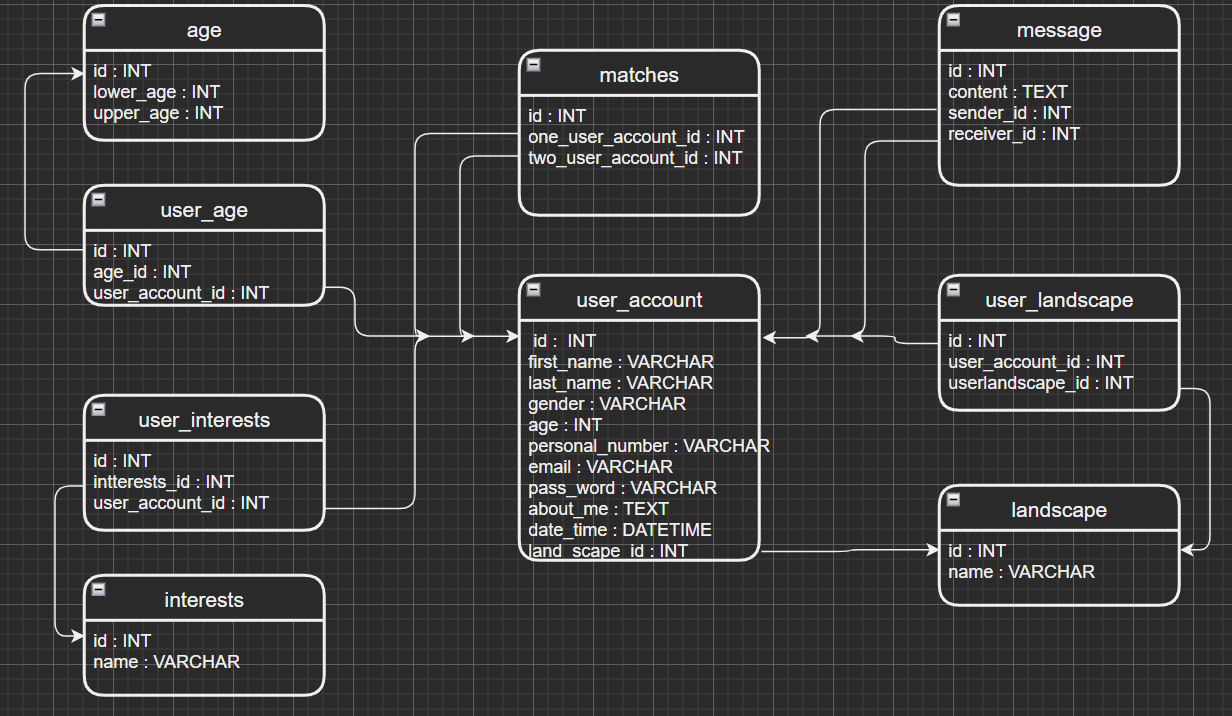
**2gether**

****

**Databasdesign**

Databasen 2gether används till ett datingprogram. Som innehåller funktioner som de flesta har idag, fast mycket är för utveckling.

2gether

* lagrar alla användare som skaffar ett konto med data, se attributen. Användaren kan uppdatera fälten email, password, about\_me på sitt konto.
* Håller age span, interests och landscapes för alternativ inför en match på tabellerna age, interests och landscape.
* Kopplar ihop och lagrar alternativ id:n med användarens id:n på user\_age, user\_interests och user\_landscape när man gör en match.
* Lagrar två användares idn vid samma val och egenskap på table matches när en match uppstår.
* Lagrar chatt mellan två användare på message table. Hämtar ut datan varje gång en användare vill använda sig av funktionen.

**Djupdyk**

Ett problem som dök upp var att först bestämma hur användarna skulle matchas. Antingen beroende på valen dem gör eller på egenskaperna dem har. Man skulle kunna matchas med val till val, egenskap till egenskap eller egenskap till val. Den queryn fick bli en blandning tillslut, både val till egenskap och val till val. Då användaren inte har intressen som egenskap så matchas användaren endast mot en annan användares val och inte mot en egenskap. Eller så skulle man kunna vända på det helt och matcha en egenskap mot en egenskap beroende på hur man ser på det.

Om man väljer att intresse är en egenskap, tex att man är en hemmamysare. Då matchas min egenskap (hemmamysare) mot en användare som har samma intresse (hemmamysare). Om man väljer att intresse är ett val av vad man söker, tex jag söker en festprisse – då baseras valet på att man letar efter samma sorts person.

Detta blev krångligt att få en helhet över i början av projektet. Och jag diskuterade denna frågan med andra och undrade över hur dom ville ha det. Jag kom fram till att svaret handlar om hur en person känner för tillfället, idag vill jag träffa en hemmamysare och imorgon en festprisse. Därför fick intresse bli baserad på val.

Age och landscape är egenskaper man som användare redan har, och användaren väljer då vilken ålder samt vilket landskap man är intresserad av och matchas med en annan användares egenskap.

**Reflektion**

Eftersom det stora problemet var att få ihop just en matchning på det bästa sättet så kom jag fram till att det berodde på att jag fick utveckla min databas utefter mitt program då jag redan hade kodat funktionerna.

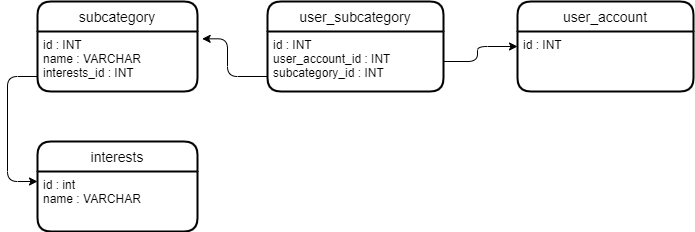
Så något jag lärt mig att vara noga över är att se över databasen först och utgår från den före jag sätter i gång med koden nästa gång. Och lägga mer tid till planeringen och bestämma hur jag vill att saker och ting ska hänga ihop.

**Vidareutveckling**

Att kunna matchas med flera val från de fyra kategorier som finns nu. Intressen, ålder, landskap och även gender. Underkategorier till intressen tex hade Jag löst så här:

- En till tabell som heter subcategory med id och namn, där man tex sätter in matlagning mm. Där står även ett interests\_id tex matlagning tillhör intresse; hemmamysarn på id 5.

- och istället för user\_interests så gör jag en ny tabell som heter user\_subcategory, som är en kopplingstabell där user\_account\_id och subcategory\_id står.



Mitt program är tydligt uppdelat från ui, genom Logic till databas så jag anser att det inte är svårt att utveckla. Det finns ett tydligt mönster att följa med interfaces som möjliggör low coupling. Och jag bedömmer även att min databas är lätt att förstå och utveckla.