Delfinen

Vi har i vores hold - bestående af Matti, Mathias og Vibeke - lavet opgaven Delfinen. Vi startet med medlem klassen, da den skulle bruges til, at lave næsten alt andet i projektet. Vi har brugt interface til, at tilføje information til databasen. Vi skulle lave et program som kan, tilføje medlemmer til database med information om aktivitet alder og om de har deltaget i konkurrencer, hvor der, ud fra alder og aktive status, skal betales den tilsvarende kontingent. Træneren ønsker, at se de medlemmers resultater, så der er derfor også lavet en database, hvor det kan blive gemt. Træneren vil se top 5 for hver svømmedisciplin. Kasserer vil kunne se hvilke medlemmer der er i restance og administrere det. Vi sørger derudover for at koden er læsevenlig, genbrugbar og robust.

Vores kode afgrænsninger:

Træner, kassér og formand har forskellige ting som de kan gøre i deres respektive user interfaces. Vi fik heller ikke implementeret en lettere måde at søge efter medlemmer på end navnet, også selvom det burde have været et cpr-nummer eller lignende unikt nummer, men da almindelige svømmeklubber ikke har adgang til den slags, tænkte vi ikke på at bruge det selv i programmet. Så en tydelig afgrænsning er det, at to med helt samme navn vil blive blandet sammen.

Status for koden:

Vi har lavet et program, hvor formanden kan tilføje medlemmer til databasen med information om aktivitet, alder og om de har deltaget i konkurrencer, hvor der derudfra skal betales den tilsvarende kontingent. Han kan derudover også rette, slette og se medlemmer af svømmeklubben.
Kasserer kan se hvilke medlemmer der er i restance og opdatere denne restance når den er betalt. Træneren kan i sin UI se medlemmernes resultater, ved hjælp fra database. Træneren kan se top 5 for hver svømmedisciplin. Han kan derudover tilføje rekorder, se liste over rekorder for medlemmer og se liste over stævne for medlemmer. UI er lavet så man vælger om man er træner, formand eller kassér. Man kan ud fra valgt person køre forskellige metoder.

Use cases og rækkefølge:

Obs! Det skal siges, at i programmet bliver brugeren først bedt om at registrere sig som formand, kassér eller træner, inden de får adgang til relevante funktioner. Derefter får de en hovedmenu med valg.
Vores use cases blev prioriteret i følgende rækkefølge;

1. Opret medlem
    - Systemet prompter brugeren, her formanden, til at indtaste oplysningerne til det nye medlem de vil lægge ind i systemet. Formanden skal selv indtaste navn og alder, kontingentet bliver regnet ud af systemet, medlemmets status som aktiv/passiv skal vælges og disciplinen medlemmet vil udøve, hvis de er aktiv, skal vælges ud fra tal for at undgå fejltastning. Herefter bliver alle dataene gemt i databasen som et nyt medlem.
2. Ret medlem
    - Systemet prompter brugeren, her formanden, til at indtaste navnet på det medlem de vil ændre informationer til. Herefter kommer en liste med ting de kan ændre. Alle ændringer bliver gemt i databasen løbende.
3. Slet medlem
    - Systemet prompter brugeren, her formanden, til at indtaste navnet på det medlem de vil slette fra systemet. Hvis medlemmet eksisterer i databasen, vil det blive slettet.
4. Se medlemsliste
    - Systemet udskriver listen direkte for brugeren. Denne funktion er tilgængelig for alle tre admin typer.
5. Administrér restance
    - Denne funktion er delt op i to dele. En liste med medlemmer og deres restance og en funktion til at ændre medlemmers restance, hvis de pludselig kommer i gæld eller har betalt deres gæld. Her vil brugeren, her kasséren, blive bedt om at indtaste navnet på det medlem der skal have redigeret deres restance fra positiv til negativ eller omvendt.
6. Tilføj resultatet til medlem
    - Her vil brugeren, her træneren, blive bedt om at indtaste de nødvendige oplysninger for at oprette et resultat til et eksisterende medlem. Informationerne vil blive gemt i databasen.
7. Tilføj færdigt stævne til medlem

    - Her vil brugeren, her træneren, blive bedt om at indtaste de nødvendige oplysninger for at oprette et stævne til et eksisterende medlem. Informationerne vil blive gemt i databasen.
8. Se top 5 i hver disciplin for konkurrencesvømmere
    - Her vil brugeren, her træneren, blive spurgt hvilken disciplin og fra hvilket hold de vil se en top 5 liste. Programmet henter informationer fra databasen og udskriver en liste med oplysninger.
9. Se samtlige resultater for alle svømmere
    - Her vil programmet udskrive en liste over alle resultater nedskrevet i databasen for alle aktive medlemmer.
10. Se samtlige stævner for alle konkurrencesvømmere
    - Her vil programmet udskrive en liste over alle gennemførte stævner nedskrevet i databasen for alle medlemmer.

Obs! Det kan også ses på billedet i bunden af dokumentet.
Denne rækkefølge blev bestemt, da vi skulle bruge eksisterende medlemmer for at kunne udarbejde de andre metoder og kategorier. Så første metode blev Opret Medlem. Der blev så løbende arbejdet på resten af medlemsmetoderne samtidig med, at andre i gruppen begyndte på listerne for konkurrencer og restance, så prioriteringen blev ikke fulgt til sidste detalje.

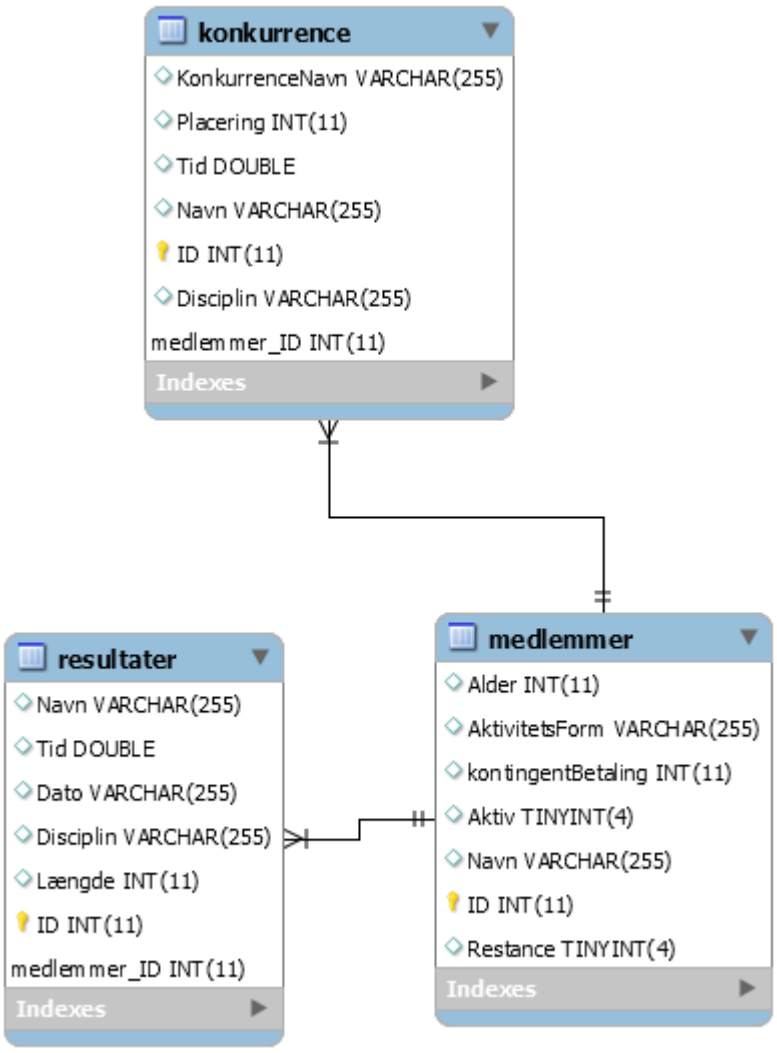
Tabeller og testdata:
Vi har tre tabeller i databasen. Medlemmer, konkurrence og resultater.
For medlemmer gemmer vi oplysninger for navn, alder, årlig kontingentbetaling, deres valgte aktivitetsform til konkurrencer og om de er i restance. De får også givet et automatisk ID. Kontingentbetalingen bliver automatisk udregnet og sat ind i databasen. Den bliver udregnet ved hjælp af medlemmets alder og deres status som aktiv eller passivt medlem. Aktivitetsformen bliver selv skrevet ind, brugeren bliver bare promptet til at vælge hvilket disciplin det er, så vi undgår at det forkerte bliver skrevet ind.

For konkurrence gemmer vi oplysninger om stævnets navn, medlemmets placering i konkurrencen, deres tid og et autogenereret ID. Navnet vil også blive gemt, da du ikke kan oprette et konkurrence resultat uden at vælge det medlem du vil oprette det til.

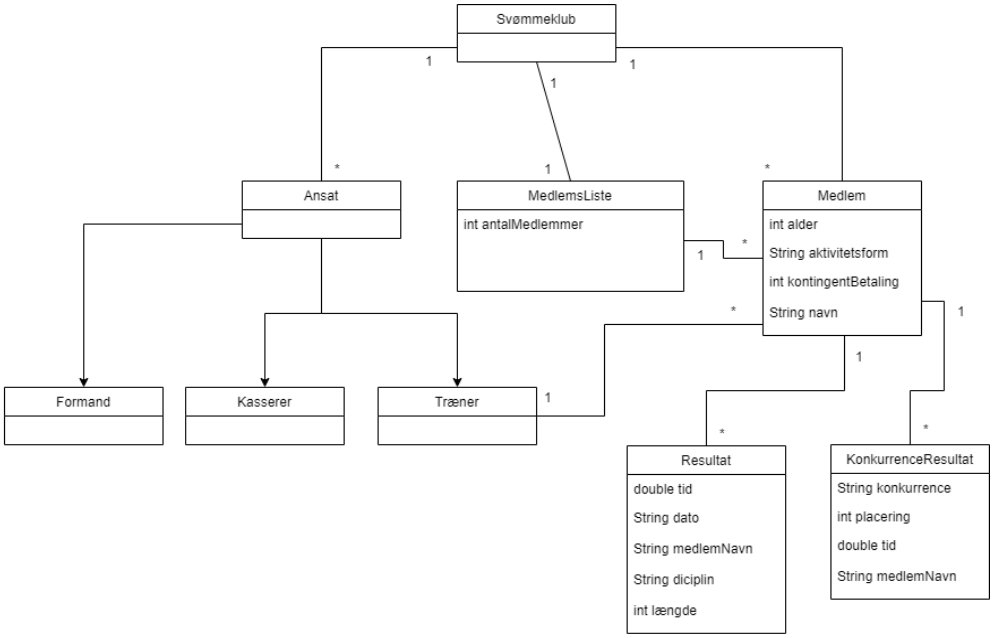
For resultat gemmer vi oplysninger om medlemmets nye rekord inden for en disciplin. Her er det tiden, datoen, disciplinen, længden af banen de har svømmet, et autogenereret ID og navnet igen, da du heller ikke kan oprette en ny rekord tid uden at sætte det til et medlem.

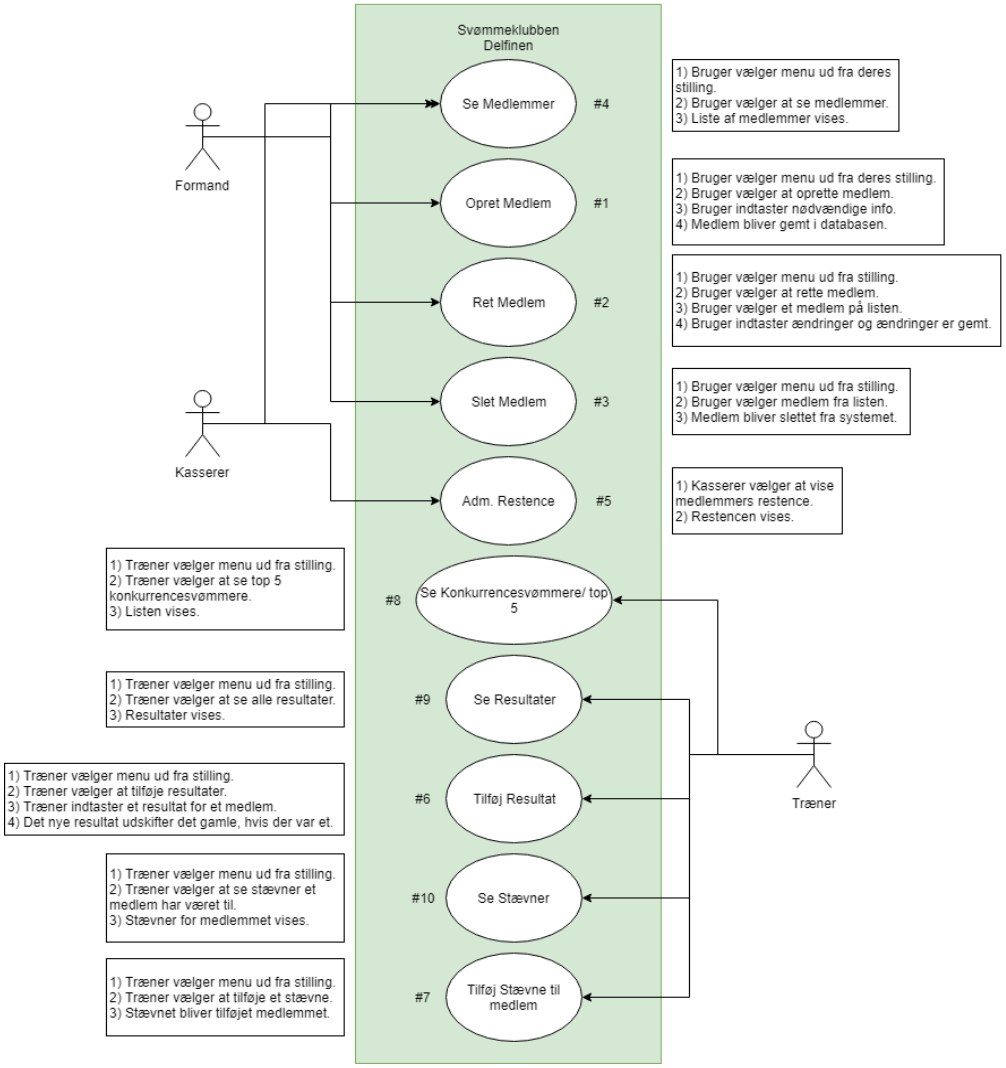
Sammenhængen mellem vores tre tabeller er så navnet på medlemmet. Det skal så også siges, at vores testdata mens vi skrev koden, ikke indeholder et efternavn, da vi bare hurtigt skrev nogle tilfældige navne ind. Der er også forskel på store og små bogstaver i disciplinerne, da vi først senere fik programmet til selv at skrive disciplinerne ind i stedet for at bede brugeren om det. Så der er nogle små afvigelser i vores testdata.

ER-Diagram:

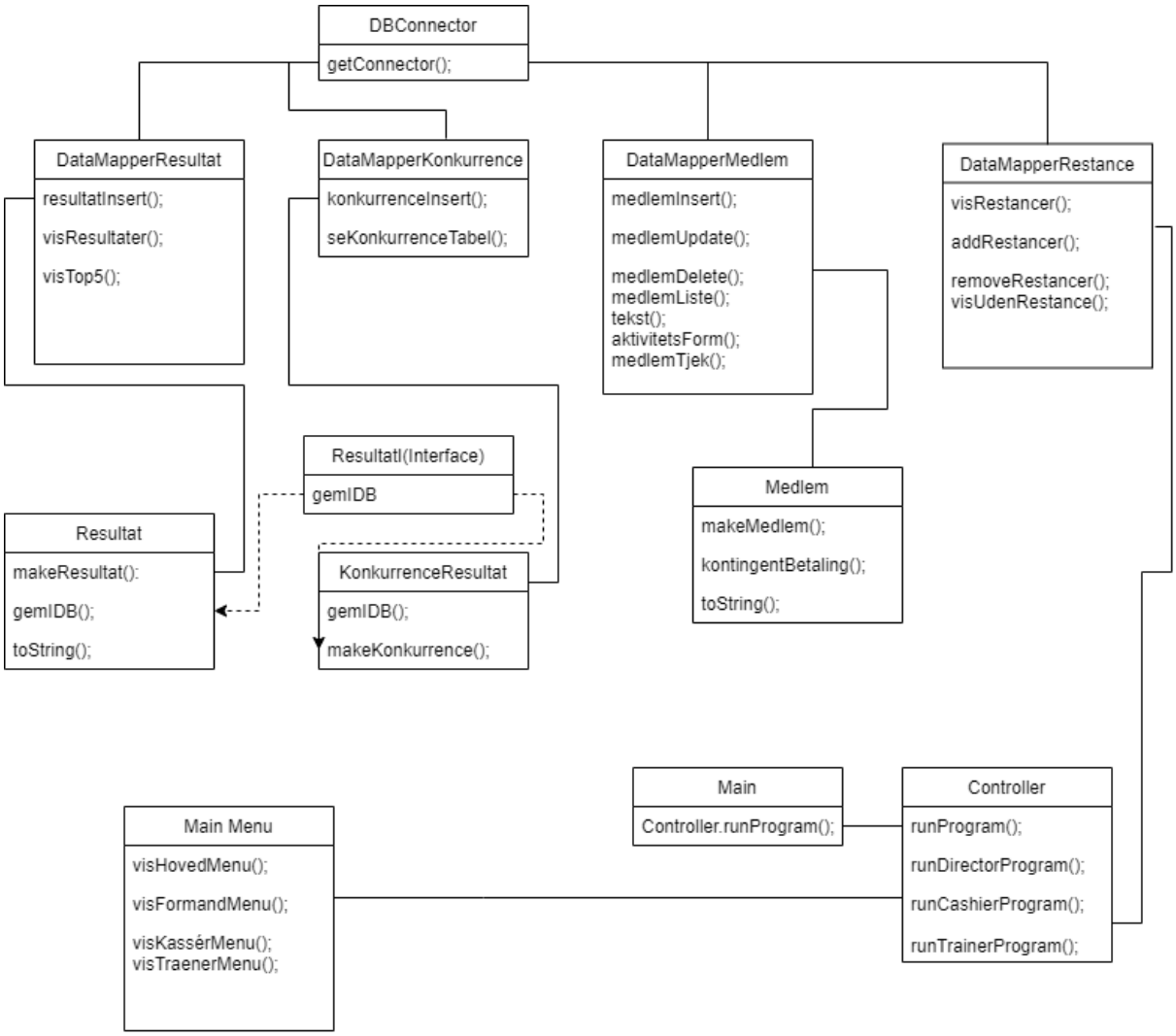


Domænemodel og Use Case diagram:


Svømmeklub
1
1
1
*
1
*
Ansat
MedlemsListe
int antalMedlemmer
Medlem
int alder
String aktivitetsform
int kontingentBetaling
String navn
1
*
1
*
1
Formand
Kasserer
Træner
1
*
*
Resultat
double tid
String dato
String medlemNavn
String diciplin
int længde
KonkurrenceResultat
String konkurrence
int placering
double tid
String medlemNavn


Svømmeklubben Delfinen
Formand
Kasserer
Træner
Se Medlemmer
#4
1) Bruger vælger menu ud fra deres stilling.
2) Bruger vælger at se medlemmer.
3) Liste af medlemmer vises.
Opret Medlem
#1
1) Bruger vælger menu ud fra deres stilling.
2) Bruger vælger at oprette medlem.
3) Bruger indtaster nødvændige info.
4) Medlem bliver gemt i databasen.
Ret Medlem
#2
1) Bruger vælger menu ud fra stilling.
2) Bruger vælger at rette medlem.
3) Bruger vælger et medlem på listen.
4) Bruger indtaster ændringer og ændringer er gemt.
Slet Medlem
#3
1) Bruger vælger menu ud fra stilling.
2) Bruger vælger medlem fra listen.
3) Medlem bliver slettet fra systemet.
Adm. Restence
#5
1) Kasserer vælger at vise medlemmers restence.
2) Restencen vises.
#8
Se Konkurrencesvømmere/ top 5
1) Træner vælger menu ud fra stilling.
2) Træner vælger at se top 5 konkurrencesvømmere.
3) Listen vises.
#9
Se Resultater
1) Træner vælger menu ud fra stilling.
2) Træner vælger at se alle resultater.
3) Resultater vises.
#6
Tilføj Resultat
1) Træner vælger menu ud fra stilling.
2) Træner vælger at tilføje resultater.
3) Træner indtaster et resultat for et medlem.
4) Det nye resultat udskifter det gamle, hvis der var et.
#10
Se Stævner
1) Træner vælger menu ud fra stilling.
2) Træner vælger at se stævner et medlem har været til.
3) Stævner for medlemmet vises.
#7
Tilføj Stævne til medlem
1) Træner vælger menu ud fra stilling.
2) Træner vælger at tilføje et stævne.
3) Stævnet bliver tilføjet medlemmet.

UML Diagram:


DBConnector
getConnector();
DataMapperResultat
resultatInsert();
visResultater();
visTop5();
DataMapperKonkurrence
konkurrenceInsert();
seKonkurrenceTabel();
DataMapperMedlem
medlemInsert();
medlemUpdate();
medlemDelete();
medlemListe();
tekst();
aktivitetsForm();
medlemTjek();
DataMapperRestance
visRestancer();
addRestancer();
removeRestancer();
visUdenRestance();
ResultatI(Interface)
gemIDB
Medlem
makeMedlem();
kontingentBetaling();
toString();
Resultat
makeResultat():
gemIDB();
toString();
KonkurrenceResultat
gemIDB();
makeKonkurrence();
Main
Controller.runProgram();
Controller
runProgram();
runDirectorProgram();
runCashierProgram();
runTrainerProgram();
Main Menu
visHovedMenu();
visFormandMenu();
visKassérMenu();
visTraenerMenu();