## Program beskrivelse

Jeg har så vidt muligt undgået at lave gentagende kode ved at lave metoder dedikeret til at gøre bestemete funktioner som serveren skal bruge ofte f.eks. database write/read hver gang en metode skal skrive eller læse fra databasen bliver de metoder kaldt.

## Klasser

Jeg valgte at ligge clientGui og admin gui i klasser så adminguien kan nedarve klient guien og derved behøver jeg ikke at lave det 2 gange derudover gav det mig også mulighed for at tilføje nogle ekstra metoder til admin guien. Admin guien giver bl.a. information i infometoden. KlientGuien har også infometoden men jeg overskriver den så i adminGui klassen så den gør noget andet.

## Diagram

Serveren bruger en bagliggende sql server som indeholder en bruger tabel og en banned users tabel. Hvor bruger tabellen har UserID som er primarykey som så forbinder til Banned users 1 til 1 hvilket betyder at der kan kun være ban per user. Serveren bruger usertabellen til at oprette brugere i og godkende brugere så de kan logge på. Grunden til at jeg har valgt at forbinde dem med userID er at det så det gør det muligt at finde ud af hvilken bruger som er blevet banned noget andet er at det gør sådan man ikke kan slette brugeren hvis han f.eks. er blevet banned det sikrer at der ikke er data i databasen som ikke hører til noget.



## Unittest

Unit test laver jeg på en lidt anderledes måde, i stedet for at importere metoder og så teste på den måde har jeg ”emuleret” en klient som forbinder til serveren. Grunden til dette er at det ville være svært at teste serverens metoder uden at den kørte. Derfor forbinder unittesten til serveren og logger på en bruger hvorefter den så begyder at sende commands til serveren ligesom en normal bruger ville derefter sender serveren så tilbage til unittesten om commanden lykkedes. Unittesten tester bl.a. for om den kan skrive og læse til sql serveren.

## SQL tabeller

CREATE TABLE [Users]

(

[UserID] int NOT NULL ,

[Username] nvarchar(50) NOT NULL ,

[Password] nvarchar(50) NOT NULL ,

[Is\_admin] int NOT NULL ,

CONSTRAINT [PK\_users] PRIMARY KEY CLUSTERED ([UserID] ASC)

);

GO

CREATE TABLE [Banned\_Users]

(

[UserID] int NOT NULL ,

[Ban\_Date] datetime NOT NULL ,

[Ban\_Reason] nvarchar(100) NOT NULL ,

[Banned\_until] datetime NOT NULL ,

CONSTRAINT [FK\_18] FOREIGN KEY ([UserID]) REFERENCES [Users]([UserID])

);

GO

CREATE NONCLUSTERED INDEX [fkIdx\_18] ON [Banned\_Users]

(

[UserID] ASC

)

GO

## Stored procedure

Dette er en stored procedure som jeg kalder i min kode det er til at godkende bruger login hvor den retunere 1 eller 0 afhængig af om den bliver godkendt eller ikke.

CREATE PROCEDURE check\_login @user\_username nvarchar(30), @user\_password nvarchar(60)

AS

SELECT

(CASE WHEN password = @user\_password THEN 1 ELSE 0 END) AS is\_equal

FROM

Chatapp.dbo.Users

WHERE

Username = @user\_username