

Tur-Retur KørselsLog Rapport

Navn - Jonas Barigo Østergaard

Uddannelse - Datatekniker med speciale i Programmering

Instruktør - Simon Nicolas El Hanafi

Opret bruger

Navn:

Date:

Nr. plade:

Cancel OK

Rediger bruger

Navn:

Date:

Nr. plade:

Cancel OK

Slet bruger

Navn:

Cancel OK

Opret KørselsLog

Navn:

Date:

Nr. Plade:

Opgave:

Cancel OK

Opret en bruger:

Indtast et navn og en nummerplade. Klik på OK for at oprette en record og få det vist i nedenstående liste. Ved klik på Cancel revideres feltene.

	navn	dato	nr_plade
#	Jon	15-03-2022	AA12345
	Olga	14-03-2022	BB12321
	KatjaKaj	15-03-2022	CC45654
	BenteBent	15-03-2022	DD65456
	Henrik	15-03-2022	EE78987
	Stef	15-03-2022	FF98765

Rediger en bruger:

Vælg en bruger fra drop down listen. Rediger nummerplade feltet. Klik på OK for at gemme den opdaterede record, og få det vist i nedenstående liste. Ved klik på Cancel sættes feltene tilbage.

	navn	dato	nr_plade
#	Jon	15-03-2022	AA12345
	Olga	14-03-2022	BB12321
	KatjaKaj	15-03-2022	CC45654
	BenteBent	15-03-2022	DD65456
	Henrik	15-03-2022	EE78987
	Stef	15-03-2022	FF98765

Slet en bruger:

Vælg en bruger fra drop down listen. Klik på OK for at slette den bruger (record) fra den nedenstående list. Ved klik på Cancel resettes listen.

	navn	dato	nr_plade
#	Jon	15-03-2022	AA12345
	Olga	14-03-2022	BB12321
	KatjaKaj	15-03-2022	CC45654
	BenteBent	15-03-2022	DD65456
	Henrik	15-03-2022	EE78987
	Stef	15-03-2022	FF98765

Opret en Kørsel:

Vælg en bruger fra drop down listen. Indtast en kort opgave beskrivelse. Klik på OK for at gemme den opdaterede record og få det vist i nedenstående liste. Ved klik på Cancel revideres feltene.

	navn	dato	nr_plade	opgave_beskrivelse
#	Henrik	16-03-2022	GG10115	Kort 12 km
	Olga	17-03-2022	BB12321	Kort 22 km

Indhold

Indledning	3
Dokumentation.....	4
Konklusion	23
Logbog	24

Indledning

I denne case er jeg blevet bedt om, at lave et windows forms program, der skal kobles til en sql database. Her skal programmet kunne indsætte, redigere, slette og vise data fra databasen.

Databasen skal indeholde data der beskriver brugere, dato, nummerplade og beskrivelse af en opgave, som brugeren har udført.

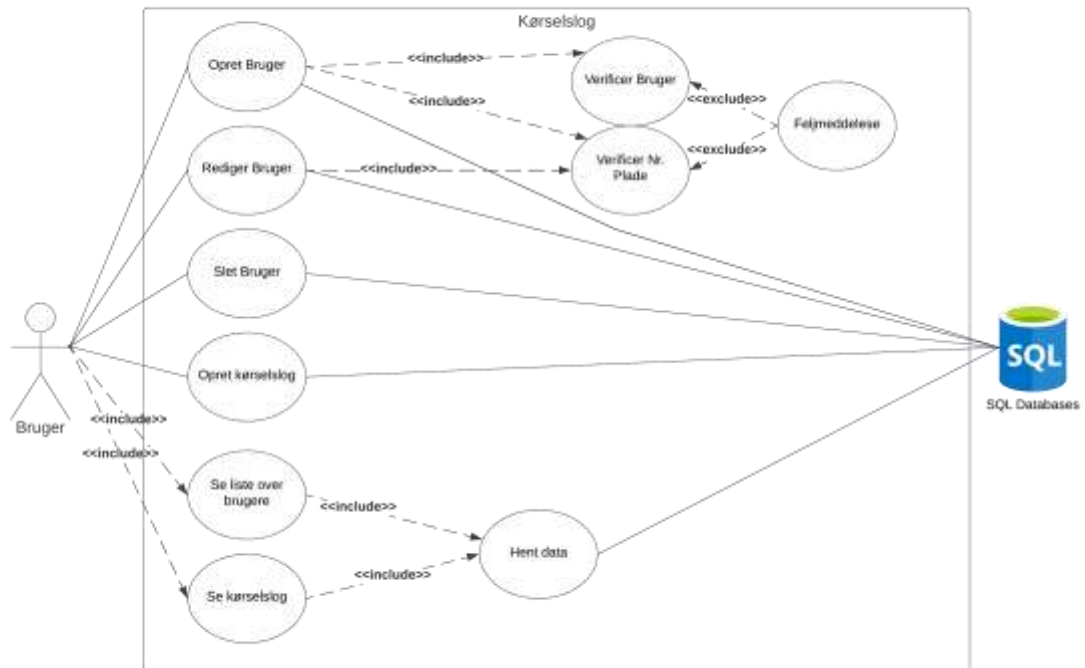
Til opgaven har jeg anvendt OOAD metoder for at analysere og specificere programmet, herunder user case, flowchart, klasse diagram, e/r diagram.

Derudover har jeg anvendt sql scripts, til at oprette min database og tabeller, og objekt orienteret programmering i c#, til at udvikle min brugergrænseflade.

Derudover har jeg også udarbejdet et Work Break Down (WBD) skema for at få afgrænset de enkelte underopgaver til mit program og på denne måde få mere struktur for hvornår og i hvilken rækkefølge jeg skal løse de enkelte opgaver.

Dokumentation

Use Case - Kørselslog



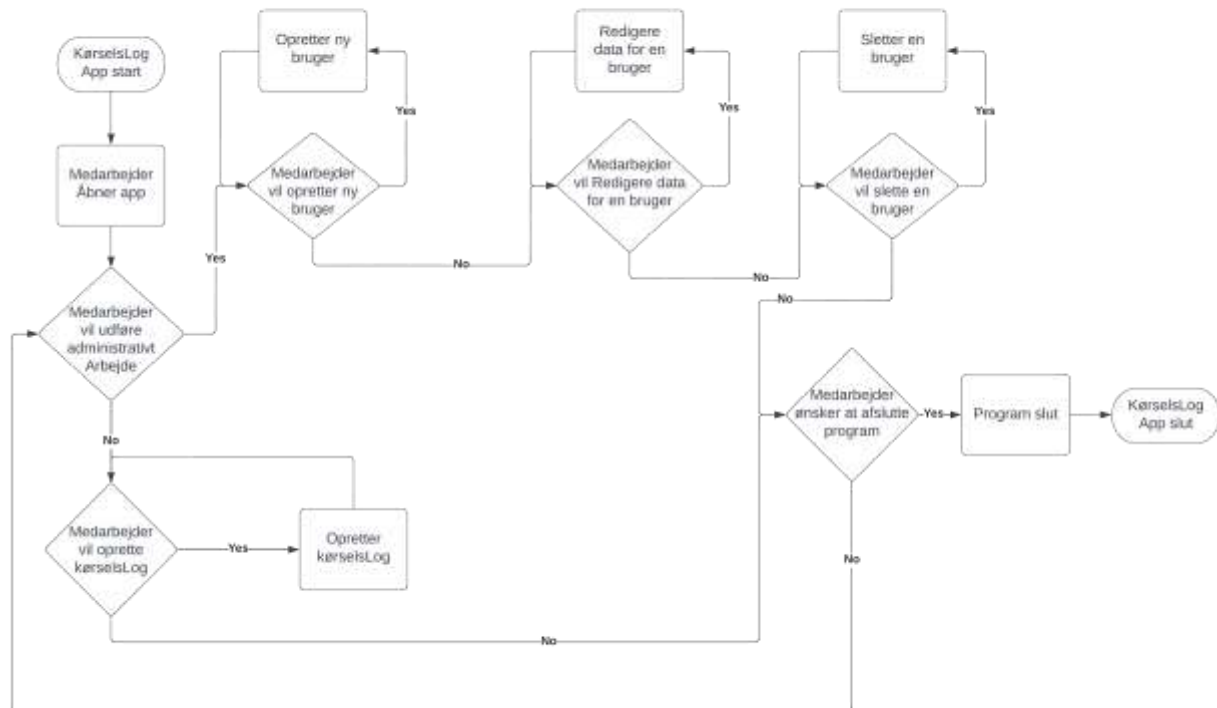
(Use case)

I denne use case er der en bruger. Denne bruger har valgt mellem at udføre administrative opgaver eller oprette en kørsels log. For de administrative opgaver har brugeren valgt mellem at oprette en bruger, redigere en bruger eller slette en bruger. Desuden kan brugeren også se en liste med oprettede brugere. Hvis brugeren ikke vil udføre administrative opgaver, er der kun et valg tilbage, nemlig at oprette en kørselslog.

Når data bliver oprettet eller opdateret, bliver de desuden verificeret af systemet automatisk, altså skal brugeren ikke foretage sig noget. Her bliver data for brugeren verificeret. Hvis verifikationen ikke går igennem kommer der en fejlmeddelelse, dette er dog kun hvis data ikke er korrekt, altså vil der ikke altid automatisk komme en fejlmeddelelse.

Brugeren har desuden ikke et valg om at få vist data, dette er noget der sker automatisk uden input fra brugeren. Data bliver desuden også automatisk hentet fra serveren.

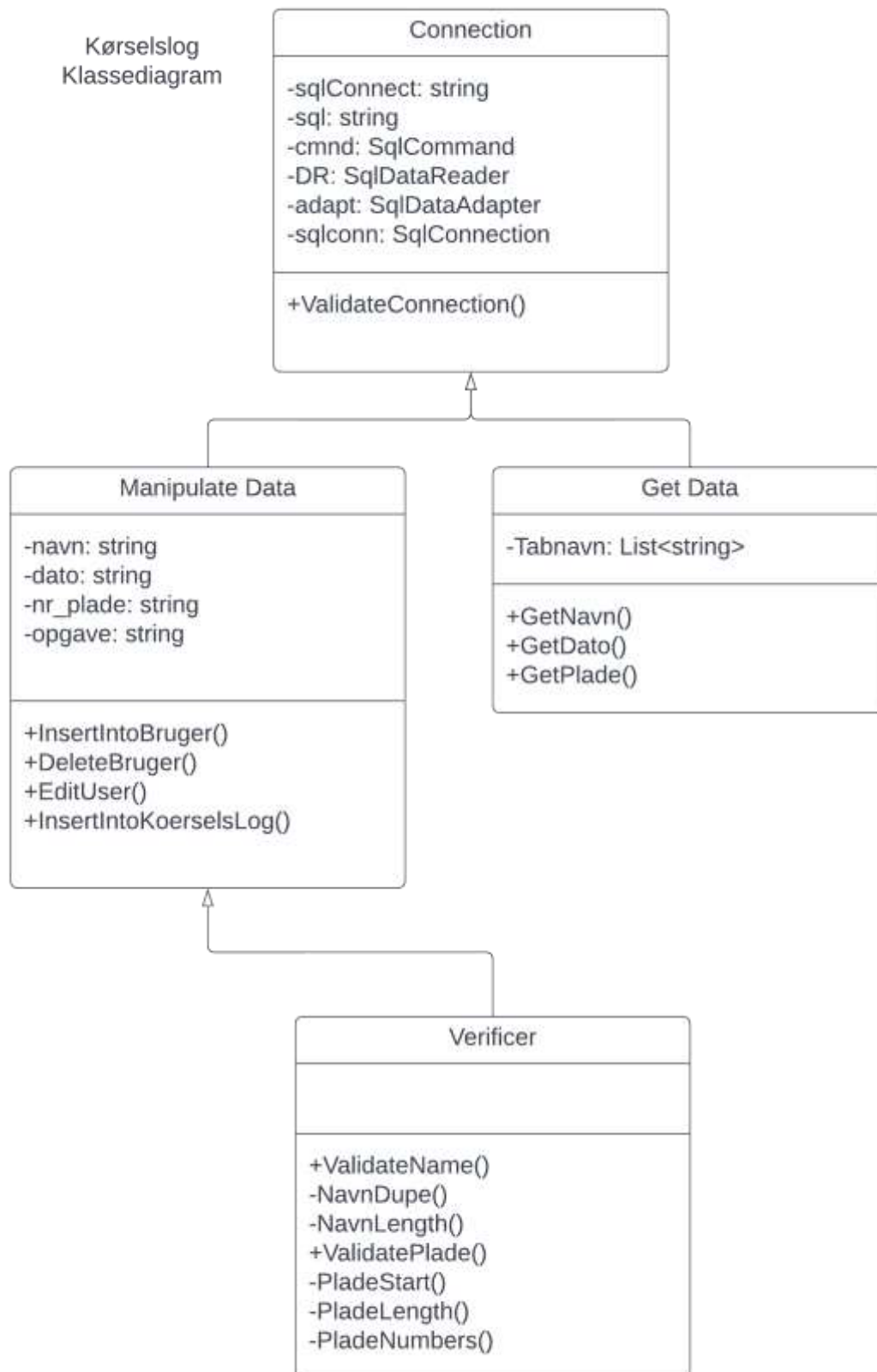
KørselsLog - Flowchart



(flowchart)

Dette flowchart tager udgangspunkt i start af Kørselslog appen, efter medarbejderen har åbnet appen, har medarbejderen et valg mellem at udføre administrativt arbejde eller om brugeren vil oprette en kørselslog. Hvis medarbejderen kun vil oprette en kørselslog, kan medarbejderen gøre dette. Medarbejderen kan desuden blive ved med at oprette kørselslogs indtil medarbejderen er tilfreds.

Hvis medarbejderen derimod vil udføre administrativt arbejde. Har medarbejderen valget mellem at oprette en ny bruger, redigere data for en bruger eller slette en bruger. Medarbejderen kan desuden gøre dette indtil medarbejderen er tilfreds. Når medarbejderen er færdig med de administrative opgaver, kan medarbejderen vælge at lukke programmet, eller starte forfra ved valget om at udføre administrativ arbejde, eller oprette en kørselslog.



(Klassediagram)

I dette klassediagram kan man se klasserne Connection, Manipulate Data, Get Data og Verificer.

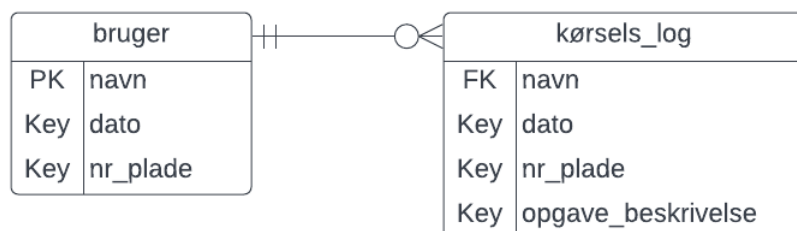
Connection er en klasse der tager sig af forbindelsen mellem programmet og SQL databasen, samt at initialisere commandoer der bruges i sammenhæng mellem SQL databasen og C# programmet.

Manipulate Data klassen tager sig af at indsætte, opdatere og slette data i SQL databasen. Her arver den egenskaber fra connection klassen og benytter forbindelsen idet klassens metoder bliver anvendt.

Get data klassen arver også fra connection klassen. Get data klassens formål er at hente data fra SQL databasen og vise dem i C# programmet.

Til sidst er der Verificer Klassen. Verificer klassen arver fra Manipulate data. Verificer klassen har til formål at verificere om de data der bliver oprettet og opdateret lever op til de krav jeg har sat. Inden de bliver indsat I SQL databasen.

KørselsLog - ERD



(ERD)

I dette Entity Relationship Diagram, kan det ses at jeg har to tabeller i min database. Den første tabel 'bruger' har en primary key (PK) som har navnet 'navn'. Denne value skal være unik da det er en primary key. Denne primary key er en unik identifikation og kan derfor ikke have værdier der er den samme. Dette betyder at hvis der bliver oprettet en bruger ved navn Jonas i min tabel, kan der ikke oprettes flere brugere med navnet Jonas. Desuden bruges primary key'en som et reference punkt af andre tabeller.

I den anden tabel 'kørsels_log' har vi næsten de samme values som i bruger. Her er navn dog en Foreign key, og der er en ny value kaldet 'opgave_beskrivelse'. Navn har en reference til tabellen bruger og valuen navn.

Relationen mellem de to tabeller er som følgende. For hver brugere kan der eksistere 0 eller flere kørsels_log, men for hver kørsels_log kan der kun eksistere 1 og kun 1 bruger.



(DB diagram)

Her ses et database diagram. Tabellen bruger har 3 values. Navn, dato og nr_plade. Navn er en Primary Key og har derfor et nøgle ikon ved siden af sig. Tabellen koersels_log har 4 values. Navn, dato, nr_plade og opgave_beskrivelse. Navn er her en foreign key or har en reference til primary key'en navn i tabellen bruger.

```
IF NOT EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'KLP') -- ser om databasen KLP ikke eksistere
begin
    Create Database KLP; -- opretter databasen 'KLP'
end
GO
USE KLP --vælger databasen KLP

IF NOT EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'dbo' AND TABLE_NAME = 'bruger') -- Ser om tabellen 'bruger' ikke eksistere
begin
    create table bruger -- opretter tabellen bruger
    (-- sætter values til tabellen bruger
    navn varchar(255) NOT NULL PRIMARY KEY, --sætter navn til at være primary key
    dato varchar(255),
    nr_plade varchar(255)
    )
end

IF NOT EXISTS (SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'dbo' AND TABLE_NAME = 'koersels_log') -- Ser om tabellen 'koersels_log' eksistere
begin
    create table koersels_log --opretter tabel koersels_log
    (--sætter values til tabellen koersels_log
    navn varchar(255) foreign key references bruger(navn), --sætter navn til at være foreign key
    dato varchar(255),
    nr_plade varchar(255),
    opgave_beskrivelse text
    )
end
```

(sql script)

Her ses mit SQL script. Dette script opretter en database med to tabeller. Databasen har navnet KLP og tabellerne har navnene 'bruger' og 'koersels_log'.

Tabellen 'bruger' bliver oprettet med følgende elementer:

- Navn som er af typen varchar. Desuden er navn en primary key og sættes derfor også som værende NOT NULL, altså må der ikke placeres værdier af typen null ind i dette element.
- Dato som er af typen varchar
- Nr_plade som er af typen varchar.

Tabellen 'koersels_log' bliver oprettet med følgende elementer:

- Navn som er af typen varchar. Navn er her en foreign key med reference til primary key'en i tabellen bruger. Dermed referere navn i koersels_log til navn i bruger.
- Dato som er af typen varchar
- Nr_plade som er af typen varchar
- Opgave_beskrivelse som er af typen text.

Windows Forms Application

```
public partial class Form1 : Form
{
    public MyTable tab = new MyTable(); // initialisere klassen MyTable
    public List<string> NavnList = new List<string>(); //opretter en string hvori brugerne bliver gemt.

    const string brugerText = "Opret en bruger:\nIndtast et navn og en nummerplade. Klik på OK for at oprette en record og få de
    const string redigerText = "Rediger en bruger:\nVælg en bruger fra drop down listen. Rediger nummerplade teksten. Klik på OK
    const string sletText = "Slet en bruger:\nVælg en bruger fra drop down listen. Klik Ok for at slette den bruger (record) fra
    const string registrerText = "Registrer en kørsel:\nVælg en bruger fra drop down listen. Indtast en kort opgave tekst. Klik p
    1 reference
    public Form1()
    {
        InitializeComponent();
    }
}
```

Her ses starten af mit program. Her initialisere jeg min klasse MyTable, jeg opretter en list af strings og jeg opretter diverse strings, der skal vises i mit program.

```
private void Form1_Load(object sender, EventArgs e) //hvad der sker når form1 bliver loadet.
{
    if (tab.ValidateConnection()) //validere forbindelse
    {
        // TODO: This line of code loads data into the 'kLPDataSet.bruger' table. You can move, or remove it, as needed.
        // udfylder tabel med data fra sql databasen
        this.brugerTableAdapter.Fill(this.kLPDataSet.bruger);
        // TODO: This line of code loads data into the 'kLPDataSet2.koersels_log' table. You can move, or remove it, as needed.
        this.koersels_logTableAdapter1.Fill(this.kLPDataSet2.koersels_log);
    }
}
```

Her i form1_load bestemmes hvad der sker, når appen først bliver loadet. Først ser jeg om tab.ValidateConnection er true og hvis den er det køres to kommandoer der hver udfylder deres egen tabel, med data fra databasen. ValidateConnection ser desuden om der er forbindelse til databasen og returnere true hvis der er. På nedenstående billeder ses de tabeller der bliver kaldt og vist i appen.

	navn	dato	nr_plade	opgave_beskrivelse
►	Bente Bent	31-03-2022	BB22334	Kørt 12 km
	Anders And	31-03-2022	AA11112	Kørt 4 km
	Katja Kaj	31-03-2022	CC33445	kørt 74 km

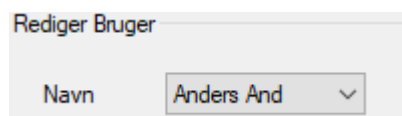
	navn	dato	nr_plade
►	Anders And	31-03-2022	AA11112
	Bente Bent	31-03-2022	BB22334
	Katja Kaj	31-03-2022	CC33445
	Ole	04-04-2022	KJ76291

```
7 references
public bool ValidateConnection() //metode der tester om der er forbindelse til serveren.
{
    try //prøver om følgende kan lade sig gøre
    {
        sqlconn.Open(); //åbner forbindelse til sql serveren
        sqlconn.Close(); //lukker forbindelse til sql serveren
        return true;
    }
    catch (SQLException) //hvis der ikke er forbindelse til serveren og man derved får en sql exception,
                        //vil følgende blive kørt
    {
        MessageBox.Show("Connection Error"); //popup med meddelelse
        return false;
        //throw; //skal det her stå her?
    }
}
```

Her ses ValidateConnection metoden der blev kaldet. Når man kalder den vil den prøve at køre koden sqlconn.Open(); sqlconn.Close(). Disse kommandoer åbner og lukker for forbindelsen til sql databasen. Hvis koden ikke kan køres vil og vi får en fejlmeddelelse vil metoden fange fejlen og køre noget kodet i stedet for. Her bliver MessageBox.Show("Connection Error") kørt. Som skrevet tidligere returnere metoden enten true eller false, afhængig af om der opstår en fejl eller ej.

```
//henter navne fra databasen og sætte dem ind i de 3 comboboxe der er oprettet.
NavnList = tab.GetNavn();
foreach (string i in NavnList)
{
    navn_pick1.Items.Add(i);
    navn_pick2.Items.Add(i);
    navn_pick3.Items.Add(i);
}
```

Dernæst I load, bliver listen NavnList sat lig med metoden GetNavn. GetNavn returnere en liste med navne som svare til værdien navn i sql databasens tabel bruger. I foreach loopet tilføjer jeg hver værdi i navnlist til 3 comboboxe jeg har oprettet I min app. Disse comboboxe fungerer som en dropdown menu hvor man nu kan vælge et navn. Combobox1 med navnet navn_pick1 ses under dette.



Rediger Bruger

Navn Anders And ▼

```
public List<string> GetNavn() //metode der henter navne fra tabellen bruger og gemmer dem i en liste.
{
    sql = "Select navn from bruger";
    sqlconn.Open();
    {
        cmd = new SqlCommand(sql, sqlconn);
        DR = cmd.ExecuteReader();

        while (DR.Read())
        {
            TabNavn.Add(" " + DR.GetValue(0));
        }
        DR.Close();
        cmd.Dispose();
        sqlconn.Close();

        return TabNavn;
    }
}
```

Her ses metoden GetNavn. Jeg starter med at sætte stringen sql = "Select navn from bruger". Dette anvendes som et sql query. Derefter åbner jeg forbindelsen til sql databasen med sqlconn.Open(). Initialisere sqlcommand med værdierne sql, som er et query, og sqlconn, der er en string som fortæller hvor og hvordan der skal forbindes til databasen, hvorefter den sættes lig med cmd.

```
// string der skal bruges til forbindelse mellem databasen og c#
private const string sqlConnect = @"Data Source=DummyServerOne;Initial Catalog=KLP;User ID=jonas;Password=Password!";
```

```
private SqlCommand cmd;
private SqlDataReader DR;
private SqlDataAdapter adapt = new SqlDataAdapter();
private SqlConnection sqlconn = new SqlConnection(sqlConnect);
```

SqlCommand sørger for at jeg kan køre min kode mod en sql database.

Cmd.ExecuteReader bliver sat lig med DR. Som er af typen sqldatareader. Sqldatareader sørger for at jeg kan læse fra sql databasen.

Dernæst startes et while loop der har til formål rent faktisk at læse dataten fra det query jeg har angivet. Inde i while loopet taget den valuen på index 0 altså det første element i rækken af data og tilføjer den til en tom string, der i øvrigt tilføjes til listen TabNavn. Når der ikke er mere at læse, vil while loopet stoppe hvorefter jeg lukker for datareader (DR), sqlconn og disposes sqlCommand (cmd).

Til sidst returneres listen TabNavn.

```

}
richTextBox.Text = brugerText;           // indsætter oprettede strings i win form
richTextBox1.Text = redigerText;
richTextBox2.Text = sletText;
richTextBox3.Text = registrerText;
Dato.Text = DateTime.Now.ToString("d");   //indsætter dagens dato ind i text box
Dato3.Text = DateTime.Now.ToString("d");

```

Tilbage I form1_load sætter koden nogle tekstbokse lig med de string der blev vist ved initialiseringen. Derudover henter den også computerens lokale dato og sætter dato og dato3 lige med dette. I DateTime.Now.ToString(); afgør hvad der står inde i parenteser hvordan datoens format skal se ud. Jeg har valgt at benytte "d" der viser dd/mm/yyyy.

```

private void button1_Click(object sender, EventArgs e) //bestemmer hvad der sker
                                                        //når man trykker på knappen cancel (opret bruger)
{
    navn_input.Text = "<text>"; //ændre textbox til at være <text> når der klikkes på cancel knappen
    nr_plade_input.Text = "<text>";
}

```

Her ses hvad der sker, ved klik på button1. Denne knap er en cancel knap og sætter de tilhørende input felter til at være default værdien "<text>".

```

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e) //bestemmer hvad der sker
                                                                        //når man ændre på valgt bruger (i rediger bruger)
{
    if (tab.ValidateConnection())
    {
        Dato2.Text = tab.GetDato(navn_pick1.Text); //henter dato der stemmer overens med datoen
                                                    //tilknyttet en bestemt bruger
    }
}

```

Her ses hvad der sker, når indexet af combobox1 blive ændret. Combobox1 indeholder, som vist tidligere, navnene på de brugere der findes i databasen.

Først tjekkes forbindelsen til serveren. Hvis der er forbindelse sættes feltet for datoen til at være den dato tilknyttet oprettelse af brugeren.


```

public string GetDato(string name) //metode der henter dato fra tabellen bruger
{
    if (!string.IsNullOrEmpty(name))
    {
        sql = "select dato from bruger where navn =" + name + " "; //da der ikke kan være duplicate navne,
                                                                    //vil der kunne være en data entry og
                                                                    //derfor kun et element der kan hentes

        sqlconn.Open();
        {
            cmd = new SqlCommand(sql, sqlconn); //query og connection
            DR = cmd.ExecuteReader(); //eksekverer datareader, der læser data ud fra ens query
            while (DR.Read()) //sørgere for at gå igennem alt dataen.
            {
                holdme = "" + DR.GetValue(0); // er det her en acceptabel måde at gøre det på
                                                // eller skal man bruge .ToString?
                //sætter en string lig med den værdi der er queriet.
            }
        }
        DR.Close(); //lukker datareader
        cmd.Dispose(); //lukker sqlcommand
        sqlconn.Close(); //lukker sqlconnection
        return holdme;
    }
    return "";
}

```

GetDato metoden ser sådan her ud. Hvis vi ser bort fra if statementet, da dette er overflødigt, fordi navnet i min dropdown ikke kan ændres til en tom string. Jeg starter igen med et nyt query, hvor jeg vælger dato fra bruger hvor navn = input. Input er her det valgte navn i dropdown menuen.

Dernæst åbner jeg forbindelse til sql databasen. Og ligesom i getNavn læser jeg fra sql databasen med datareader.

Til sidst lukker jeg forbindelsen til sql serveren og returnere stringen holdme der indeholder den læste data.

```

private void button3_Click(object sender, EventArgs e) // hvad der sker når man trykker OK (rediger bruger)
{
    if (tab.ValidateConnection())
    {
        if (!string.IsNullOrEmpty("" + navn_pick1.SelectedValue))
        {
            MessageBox.Show("Input blev ikke indsat i databasen\nIntet navn valgt.");
        }
    }
}

```

Her ses hvad der sker når knappen button3 bliver klikket. (ok knap under rediger bruger)

Først tjekkes forbindelsen. Hvis der er forbindelse tjekkes der for om der er valgt et navn. I multiboxen navn_pick1. Hvis der ikke er det vil en messagebox poppe op med en fejlbesked.

```

else if (tab.ValidatePlade(nr_plade_input2.Text)) // validere om nr_plade lever op til de satte krav
{
    tab.EditUser(navn_pick1.Text, nr_plade_input2.Text); //ændre nummerpladen i tabellen bruger
                                                         //udfra de inputs der er i winform
}

```

Hvis der er valgt en gyldig value, vil vi gå videre til else if statementet. Her bliver input feltet for nr_plade valideret via metoden ValidatePlade. Hvis nr_plade er gyldig bliver metoden EditUser kaldt med navnet og nr_pladen der skal ændres.

```
public bool ValidatePlade(string plade) // metode der validere om input lever op til kriterier for nummerplader
{
    if (!PladeLength(plade)) //kalder funktioner - hvis der returneres false sker følgende
    {
        MessageBox.Show("Input Fejl:\nPlade opfylder ikke længde-krav");
        return false;
    }
    else if (!PladeStart(plade))
    {
        MessageBox.Show("Input Fejl:\nDe to første tegn I nummerpladen skal være tal");
        return false;
    }
    else if (!PladeNumbers(plade))
    {
        MessageBox.Show("Input Fejl:\nPlade skal indeholde fem tal efter bogstaverne");
        return false;
    }
    return true;
}
```

Her ses metoden ValidatePlade. Metoden tager en string som input. Metoden tjekker først om input har en vis længde. Dette gør den med metoden PladeLength. Dernæst tjekker den om de to første tegn i input er bogstaver. Dette gøres med metoden PladeStart. Til sidst kaldes metoden PladeNumbers. Her tjekkes input for om resten af input er tal. Desuden kommer der en fejlbesked hvis input ikke lever op til kravene.

```
private bool PladeLength(string plade) //metode der tjekker om inputs længde er 7
{
    if (plade.Length == 7)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Her ses metoden PladeLength. Metoden returnere true hvis inputtet er 7 tegn langt og returnere false hvis inputtet er alt andet.

```
private bool PladeStart(string plade) //metode der tjekker om de to første tegn i input er bogstaver
{
    if (char.IsLetter(plade[0]) && char.IsLetter(plade[1]))
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Her ses metoden PladeStart. PladeStart tjekker for om tegnene på position 0 og 1 i mit input er et bogstav. Hvis de begge er bogstaver returneres true ellers returneres false.


```
private bool PladeNumbers(string plade) //metode der tjekker om input indeholder bogstaver hvor der skal være tal.
{
    for (int i = 2; i < plade.Length; i++)
    {
        if (Char.IsLetter(plade[i]))
        {
            return false;
        }
    }
    return true;
}
```

Her ses metoden PladeNumbers. Metoden tjekker om hvert tegn i input er et bogstav, udover de to første tegn. Dette gøres via et for loop. Hvor int i bruges som index af input, derfor sættes den til at være 2. Når index er 2 vil index altså se på position 3 da index starter på 0. Derudover besluttet det at for loopet vil køre så længe i er mindre end input's længde. Vi ved at længden på input kun kan være 7, index går fra 0-6 og skal se på i = 6 som det sidste. Derfor skal for loopet stoppes når i er lig med eller større end pladens længde. Altså skal i < plade.length for at loopet skal køre.

Inde i loopet tjekkes der for om hver char er et bogstav i input ved index i. Hvis en af disse chars viser sig at være et bogstav, returneres false. Hvis ingen af chars i input er bogstaver returneres true.

```
else
{
    MessageBox.Show("Input blev ikke indsat I databasen"); //popup med meddelelse
}
nr_plade_input2.Text = "<text>"; //ændre input til <text>
this.brugerTableAdapter.Fill(this.kLPDataSet.brugere); //opdaterer den viste tabel med brugere
```

Hvis vi går helt tilbage og kigger på hvad der sker når man trykker på button3, er der et else statement til sidst. Hvis vi går ind i else statementet kommer der en dialog boks op, der fortæller at inputtet ikke blev indtastet i databasen. Derudover, efter else statemnetet, ændres inputboksen tilbage til <text> og tabellen med data for brugere bliver opdateret.

```
private void button5_Click(object sender, EventArgs e) //bestemmer hvad der skal ske, når man trykker ok (slet bruger)
{
    if (!string.IsNullOrEmpty("" + navn_pick2.SelectedValue))
    {
        MessageBox.Show("Input blev ikke indsat i databasen\nIntet navn valgt.");
    }
}
```

Her ses hvad der sker når man trykker på knappen ok under slet bruger. Her vil der først blive tjekket om et navn er blevet valgt i comboboxen navn_pick2. Hvis der ikke er valgt noget navn, vil man få en fejlmeddelelse.

```
else if (tab.ValidateConnection())
{
    tab.DeleteBruger(navn_pick2.Text); //sletter data i tabellen bruger udfra valgt navn
    this.brugerTableAdapter.Fill(this.kLPDataSet.bruger); //opdaterer den viste tabel med brugere
}
```

Her ses hvad der sker hvis der er valgt et navn. Først ser vi om der er en forbindelse til serveren. Hvis der er det, kaldes metoden DeleteBruger og den viste tabel bliver efterfulgt opdateret.

```
public void DeleteBruger(string name) //metode der sletter en række i tabellen bruger.
{
    sql = "delete bruger where navn='" + name + "'";
    sqlconn.Open();
    {
        adapt.DeleteCommand = new SqlCommand(sql, sqlconn);
        adapt.DeleteCommand.ExecuteNonQuery();
    }
    adapt.Dispose();
    sqlconn.Close();
}
```

Her ses metode DeleteBruger, først har jeg et sql query der finder navnet på den valgte bruger i databasen. Derefter bruges adapt.deletecommand og executenonquery til at slette rækken hvori det valgte navn findes. Til sidst lukkes og disposes de sql kommandoer der er blevet anvendt

```
navn_pick1.Items.Remove(navn_pick2.Text); // sletter navn fra dropdown
navn_pick2.Items.Remove(navn_pick2.Text);
//navn_pick3.Items.Remove(navn_pick2.Text); //hvorfor virker den her ikke?
```

Tilbage til ok knappen. Efter tabellen bliver opdateret, slettes navnet også fra comboboxene. Her bruges kommandoen Items.Remove. Som man kan se udfra mine kommentarer, havde jeg lavet en dum fejl her og kunne ikke få navn_pick2 til at blive slettet fra navn_pick3. Dette er selvfølgelig fordi jeg prøvede at slette navn_pick3 efter at jeg havde slettet navn_pick2 og derved prøvet at slette en tom string.

```
navn_pick1.Items.Remove(navn_pick2.Text);
navn_pick3.Items.Remove(navn_pick2.Text);
navn_pick2.Items.Remove(navn_pick2.Text);
```

Her ses hvordan det endte med at virke.

```
private void ok_Click(object sender, EventArgs e) //bestemmer hvad der sker når der trykkes OK (opret bruger)
{
    if (tab.ValidateConnection())
    {
        if (tab.ValidateName(navn_input.Text) && tab.ValidatePlade(nr_plade_input.Text)) //validere 2 inputs
        {
            tab.InsertIntoBruger(navn_input.Text, nr_plade_input.Text); //indsætter data ind i tabellen bruger
            navn_pick1.Items.Add(navn_input.Text); //opdatere dropdown liste, med det indtastede navn
            navn_pick2.Items.Add(navn_input.Text);
            navn_pick3.Items.Add(navn_input.Text);
        }
    }
}
```

Her ses hvad der sker når der trykkes på OK knappen under opret bruger. Først valideres forbindelsen. Hvis der er forbindelse valideres navn og nr_plade. Disse valideres ud fra metoderne ValidateName og ValidatePlade. Hvis disse returnere true kaldes metoden InsertIntoBruger der indsætter inputs ind i databasens tabel bruger. Derudover tilføjes det input man har valgt under navn, til de dropdown menuer der findes i appen.

```
public bool ValidateName(string name) //metode der validere om kriterierne for navn er opfyldt
{
    if (!NavnDupe(name))
    {
        MessageBox.Show("Input Fejl:\nNavn eksistere i forvejen");
        return false;
    }
    else if (!NavnLength(name))
    {
        MessageBox.Show("Input Fejl:\nNavn er for kort");
        return false;
    }
    else
    {
        return true;
    }
}
```

Her ses ValidateName metoden. Metoden kalder to andre metoder og hvis de returnere false popper en fejlmeddelelse op og ValidateName metoden returnere false. Hvis begge metoderne returnere true, returnere validateName metoden også true. Den første af de to metoder 'NavnDupe' tjekker om input allerede findes i databasen. Og den anden metode, 'NavnLength', tjekker om længden af navnet er tilstrækkelig.

```
private bool NavnDupe(string name) //metode der tjekker om input allerede findes i tabellen bruger
{
    sql = "select navn from bruger";

    sqlconn.Open();
    {
        cmd = new SqlCommand(sql, sqlconn);
        DR = cmd.ExecuteReader();
    }
}
```

Her ses første del af metoden NavnDupe. Først laves et sql query og forbindelsen til databasen åbnes. SqlCommand initialiseres med sql queriet og forbindelses stringen og DataReader eksekveres.

```

while (DR.Read())
{
    s = "" + DR.GetValue(0);
    if (s == name)
    {
        DR.Close();
        cmd.Dispose();
        sqlconn.Close();
        return false;
    }
}

```

Her ses hvad der sker mens datareader læser fra databasen. En string 's' sættes lig med en tom string + værdi 0, altså den første værdi i tabellen som datareader læser. Dette gøres da værdien der kommer fra getvalue ikke er en string, men jeg gerne vil arbejde med den som en string. Derefter tjekkes værdien for om den er det samme som det input jeg har, altså name. Hvis s og input er det samme, lukkes forbindelsen og false bliver returneret.

```

    }
    DR.Close();
    cmd.Dispose();
    sqlconn.Close();
}
return true;
}

```

Her ses hvad der sker hvis s ikke er det samme som input. Her vil forbindelserne lukkes og true bliver returneret.

```

1 reference
private bool NavnLength(string name) //tjekker om input er mindre end 3 tegn langt,
                                     //hvis det er returneres false, hvis ikke returneres true
{
    if (name.Length < 3)
    {
        return false;
    }
    else
    {
        return true;
    }
}

```

Her ses den anden metode, NavnLength. Her tjekkes der for om inputs længde er mindre end 3, hvis den er det returneres false, ellers returneres true. Man kunne udvide denne til også at have et maksimum i stedet for kun at have et minimum.

```

else
{
    MessageBox.Show("Input blev ikke indsat I databasen"); //popup meddelelse
}

```

Tilbage til ok knappen. Hvis ValidateName eller ValidatePlade, som i øvrigt blev gennemgået tidligere, returnere false, går vi ned i dette else statement og udskriver en fejlbesked.


```
//opdatere diverse elementer
navn_input.Text = "<text>";
nr_plade_input.Text = "<text>";
this.brugerTableAdapter.Fill(this.kLPDataSet.bruger);
```

Til sidst opdatere vi disse elementer, således at standard input er <text> for både navn input og nr_plade input. Derudover opdateres tabellen også til at afspejle evt. ny data, denne kunne godt blive rykket til kun at blive eksekveret hvis validateName og ValidatePlade begge returnere true.

```
private void comboBox1_SelectedIndexChanged_1(object sender, EventArgs e) //bestemmer hvad der sker
//når man ændre valgt bruger (koerselslog)
{
    if (tab.ValidateConnection())
    {
        Nr_plade_get.Text = tab.GetPlade(navn_pick3.Text); //henter nummerplade der svarer til valgte bruger
    }
}
```

Her ses hvad der sker når man ændre på index af combobox1, med andre ord hvad der sker når man ændre valgt navn under oprettelse af kørselsLog. Her tjekkes der for forbindelse, hvis der er forbindelse sættes Nr_plade input lig med metoden GetPlade.

```
1 reference
public string GetPlade(string name) //metode der henter nr_plade fra tabellen bruger.
{
    if (!string.IsNullOrEmpty(name))
    {
        sql = "select nr_plade from bruger where navn='" + name + "'";
        sqlconn.Open();
        {
            cmd = new SqlCommand(sql, sqlconn);
            DR = cmd.ExecuteReader();
            while (DR.Read())
            {
                holdme = "" + DR.GetValue(0);
            }
        }
        DR.Close();
        cmd.Dispose();
        sqlconn.Close();
        return holdme;
    }
    return "";
}
```

Her ses metoden GetPlade. Først tjekkes der for om en string er gyldig, hvis den er det sættes sql stringen lig med et query der finder nr_plade i tabellen bruger ud fra et navn i tabellen bruger. Altså findes en brugers nummerplade.

Når DataReader læser fra databasen, sætter den stringen 'holdme' lig med den læste data. Da der kun er et dataelement, fordi alle navne er unikke, kan dette gøres. Stringen holdme bliver returneret. Desuden bliver en tom string returneret hvis input er null eller tom

```
private void button1_Click_1(object sender, EventArgs e) // bestemmer hvad der sker når man trykker OK (koerselslog)
{
    if (tab.ValidateConnection())
    {
        if (!string.IsNullOrEmpty(navn_pick3.Text) && !string.IsNullOrEmpty(opgave_text.Text))
        {
            tab.InsertIntoKoerselsLog(navn_pick3.Text, Dato3.Text, Nr_plade_get.Text, opgave_text.Text);
            //indsætter data ind i databasen
            this.koersels_logTableAdapter1.Fill(this.kLPDataSet2.koersels_log); //opdaterer viste tabel
            opgave_text.Text = "<text>"; //ændre input til at være <text>
        }
    }
}
```

Her ses hvad der sker når OK knappen trykkes under KørselsLog. Først tjekkes forbindelsen, hvis der er forbindelse tjekkes der for om der er valgt et navn i comboboxen og om der er blevet skrevet en opgave beskrivelse. Hvis der er det, kaldes metoden InsertIntoKoerselsLog. Denne metode indsætter data ind i databasens tabel koersels_log. Efter dataen er blevet sat ind, opdateres en tabel I appen for at vise den indtastede data og opgave_text input bliver sat til <text>.

```
public void InsertIntoKoerselsLog(string name, string dato, string plade, string opgave)
//metode der indsætter input ind i tabellen koersels_log
{
    sql = "insert into koersels_log (navn, dato, nr_plade, opgave_beskrivelse) " +
        "values('" + name + "', '" + dato + "', '" + plade + "', '" + opgave + "')";
    sqlconn.Open();
    {
        cmdnd = new SqlCommand(sql, sqlconn);
        adapt.InsertCommand = new SqlCommand(sql, sqlconn);
        adapt.InsertCommand.ExecuteNonQuery();
    }
    cmdnd.Dispose();
    adapt.Dispose();
    sqlconn.Close();
}
```

Her ses metoden InsertIntoKoerselsLog. Denne metode fungerer på samme måde som metoden InsertIntoBruger, her er der blot et ekstra element der bliver sat ind i databasen.

Konklusion

I denne opgaver har jeg arbejdet med OOAD, SQL og OOP. Jeg har brugt disse til at oprette diagrammer over den case jeg har fået, oprette en database udfra de UML diagrammer jeg har lavet og til sidst programmeret en windows forms application, til at varetager en kørselslog. I mit program har jeg derfor skulle forbinde windows forms application'en til min SQL database, hvilket jeg fandt lidt udfordrende til at starte med, men endte op med at finde det spændende. Jeg blev nødt til at ændre i portene og rettighederne for min database på min windows server i den virtuelle maskine jeg anvendte.

For at forbinde C# og SQL Databasen brugte jeg `System.Data.SqlClient`, da jeg her kunne bruge kommandoer egnet til at forbinde til sql databasen, samt manipulere databasen via inputs fra C#. Desuden har jeg anvendt CRUD princippet. Dette kommer til udtryk i min grænseflade, hvor der er muligheder for at oprette data, læse data, opdatere data og slette data fra min sql database.

Logbog

08-03-2022

I dag startede jeg med at læse og forstå opgaven, samt start af de første diagrammer. Først lavede jeg use case og derefter flowchart. Derefter oprettede jeg et E/R diagram, dette nåede jeg ikke at blive færdig med i dag.

09-03-2022

I dag startede jeg med at lave resten af E/R diagrammet, derefter lavede et database diagram og til sidst lavede jeg databasen via et script. I databasen oprettede jeg desuden to tabeller vi det same script, som jeg oprettede databasen med.

10-03-2022

I dag startede jeg med at lave et klasse diagram over databasen, hvorefter jeg startede på mit windows forms program. Da jeg ikke har beskæftiget mig med windows forms før, satte jeg mig til at finde oplysninger på nettet om hvordan man gør. Derudover satte jeg windows forms op således at det lignede mockup'en vi fik udleveret i opgaven.

11-03-2022

I dag fortsatte jeg med at opsøge viden omkring windows forms og hvordan man forbinder til en database i C#. Derudover begyndte jeg for alvor at skrive programmet og teste den viden jeg havde fundet af. Her fik jeg C# til at hente mine tabeller's data og jeg lavede funktioner der tjekkede mine data for at se om de levede op til nogle krav jeg selv fandt på.

14-03-2022

I dag fik jeg lavet en metode der oprettede en bruger i sql databasens tabel. Her indtaster man et navn og en nummerplade trykker på OK knappen og dataen bliver så indsat i databasen.

15-03-2022

I dag fik jeg lavet en metode til at redigere en eksisterende bruger, i databasen. Derudover fik jeg også lavet en metode til at slette en bruger. Til sidst anvendte jeg også et datagridview for at få vist de tabeller jeg har oprettet og deres indtastede data.

16-03-2022

I dag fik jeg oprettet en metode til at indsætte data for kørselsloggen. Derudover fik jeg også vist den i et datagridview. Derudover lavede jeg også metoder til at verificere om der er forbindelse til serveren, samt om nr. pladen er det rigtige format og om brugerens navn allerede eksistere i databasen.

17-03-2022

I dag itererede jeg på nogle af de metoder jeg havde lavet og andre dele af programmet. For at se om jeg kunne gøre dem bedre eller lave dem på en smartere måde.

18-03-2022

I dag fortsatte jeg med iterationen af koden. Her ændrede jeg min opdatering af navn i drop down menuen. Førhen slettede jeg en liste med alle navnene i og skrev dem alle sammen ind i listen igen via et loop. Dette ændrede jeg til at det navn som jeg skriver ind i tabellen bliver tilføjet til min liste uden at hente det fra databasen.

Derefter gik jeg i gang med at lave rapporten.