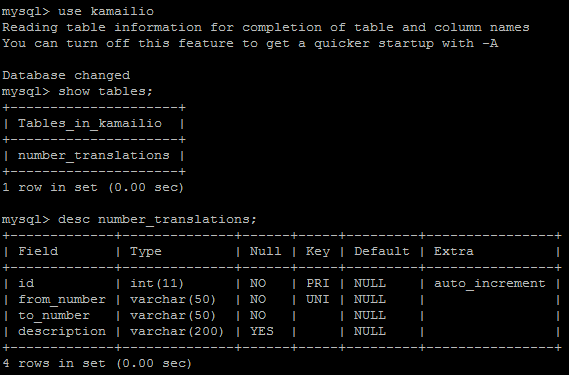
DcTranslate

*En frontend til styring af nummeroversættelser mod Danish Crown mobiler.*



**Fag:** Akademiuddannelse i Informationsteknologi – Grundlæggende Programmering

**Navn:** Klaus Vestfall Mark

**Undervisere:** Per Ole Kjeldsen og Kristian Toustrup Langborg-Hansen

**Uddannelsessted:** SmartLearning

**Dato:** 30. maj 2016

**Tegn:** 10.525

Indhold

[Indledning 3](#_Toc452364519)

[Løsningen 4](#_Toc452364520)

[Udfordringer 5](#_Toc452364521)

[Væsentlige klasser og beskrivelse af deres funktion 6](#_Toc452364522)

[NumberTranslation 6](#_Toc452364523)

[INumberTranslationRepository 6](#_Toc452364524)

[NumberTranslationRepositorySqliteDemo 6](#_Toc452364525)

[NumberTranslationRepositoryMysql 6](#_Toc452364526)

[NotifyBase 6](#_Toc452364527)

[DelegateCommand 6](#_Toc452364528)

[MainWindowViewModel 6](#_Toc452364529)

[ButtonProperties 6](#_Toc452364530)

[DoubleClickProperty 6](#_Toc452364531)

[IViewFunctions 6](#_Toc452364532)

[ViewFunctions 6](#_Toc452364533)

[ViewModelLocator 6](#_Toc452364534)

[E164ValidationRule 7](#_Toc452364535)

[FromNumberValidationRule 7](#_Toc452364536)

[Perspektivering 8](#_Toc452364537)

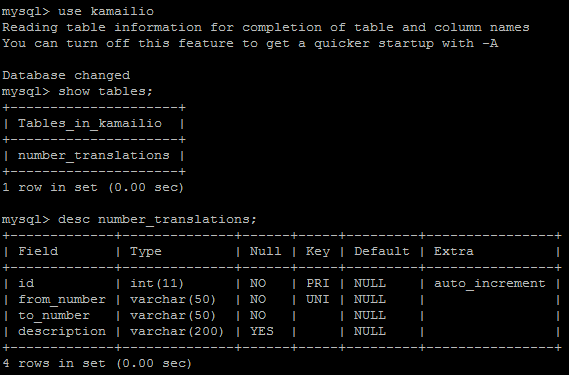
[Konklusion 9](#_Toc452364538)

# Indledning

I forbindelse med at Telia har vundet telefoniudbuddet i Danish Crown er der dukket en række tekniske udfordringer op. En af udfordringerne er at man i dag fra en telefon i Danish Crown kan ringe 18XXXX for at ringe til brugeren med lokalnummeret XXXX’s mobil. Bagved ligger bare et almindeligt mobilnummer, altså skal 18XXXX slås op i en database for at finde det bagvedliggende mobilnummer og derefter ringe det op. Dette kan Telia ikke lave og derfor vil vi selv løse opgaven.

Vi har lavet backend’en som er en SIP gateway der ud fra et opslag i en mysql database gør som beskrevet, nu mangler vi bare en frontend som vores service desk kan vedligeholde databasen med.

Opgaven er altså at lave en frontend til følgende simple database/tabel:



Opgaven løses dog med Sqlite med samme data struktur, dette er for at i også kan få adgang til nogle data og se at det virker.

Backenden fungere altså på den måde at får den et kald til 181234 slår den dette nummer op i from\_number i tabellen og retter opkaldet til at være til det der står i to\_number feltet. Description er blot til logning og til at brugeren af frontenden lettere kan knytte nummer og bruger sammen.

Det er meningen at første version af frontenden skal være hurtigt færdig (deadline for skift til Telia er 8. juni). Skal tjene som POC applikation for ledelsen og projektet i sin helhed. Frontenden skal kunne vise, tilføje, fjerne og rette nummeroversættelser. Der skal kunne søges i felterne så man let kan finde og arbejde med rækkerne i databasen. Applikationen skal tage højde for at databasen kan blive stor, så den skal kun hente de data der vises, og databasen skal forespørges så lidt som muligt.

I første omgang er udseende ikke vigtigt, det er vigtigere at applikationen laves så lagopdelt som muligt da det nok i sidste ende skal være en ASP.NET MVC løsning.

# Løsningen

Applikationen laves i WPF og overholder MVVM patteren uden code behind.

Der laves en klasse der skal repræsentere en linje i databasen og et repository der skal fungere som et lag mellem databasen og applikationen. Altså en abstraktion i forhold til valget af database. Dette er særligt vigtigt, da vi overvejer om man faktisk burde kommunikere med databasen igennem et WEB baseret API liggende på SIP gatewayen, og dermed ikke give nogen direkte adgang til mysql databasen.

Applikationen skal ikke bare hente alle data fra databasen og arbejde med dem lokalt, den skal så vidt muligt kun hente de data den skal bruge. Det vil gøre applikationen mere responsiv på langsomme forbindelser men sikrer også at man arbejder med de nyeste data. Samtidig vil databasens størrelse ikke påvirke applikationens performance.

Der vises et antal rækker pr. side og med mulighed for at skifte mellem siderne, både antallet af rækker pr. side og siden kan vælges af brugeren.

Man skal kunne højreklikke på en række og rette eller slette den. Det er også fra denne menu man tilføjer en ny række.

# Udfordringer

Største udfordring har været at lave søgningen samtidig med at der kun vises x rækker pr side.

Det var også en udfordring ikke at lave flere ens kald efter hinanden til databasen når man bruger MVVM og binding. Retter man en property kan man let overse at man faktisk afvikler noget ekstra kode. Det er faktisk en af de lidt større problemstillinger når man bruger MVVM.

Også det at have data i databasen og data på skærmen til at være opdaterede viste sig at give udfordringer. For at løse dette implementeres INotifyPropertyChanged på DTO objektet, for så vedligeholder WPF det der bliver vist. Der er mange forskellige meninger om, hvorvidt det er okay at implementere INotifyPropertyChanged på model-objekter, da det forstyrre deres hensigt.

# Væsentlige klasser og beskrivelse af deres funktion

## NumberTranslation

Klasse der repræsentere en række i databasen / en nummeroversættelse. Implementere INotifyPropertyChanged så brugere af klassen får underretninger hvis data ændre sig.

## INumberTranslationRepository

Interface der skal skjule den reelle implementation af det gældende repository.

## NumberTranslationRepositorySqliteDemo

En Sqlite version af et repository. Klassen er ment som brug til demo.

## NumberTranslationRepositoryMysql

En Mysql version af et repository. Dette er det repository der skal bruges i produktion og snakke direkte med den mysql database der styrer kaldsoversættelserne.

## NotifyBase

En klasse jeg har skrevet over noget tid. Den skal hjælpe med at implementere INotifyPropertyChanged så klasser der nedarver fra den bliver mere simple og lettere læse.

## DelegateCommand

En implementation i ICommand. Findes også i en generisk version. Bruges f.eks. til at knytte klik på en knap til et kald i ViewModel.

## MainWindowViewModel

ViewModel for MainWindow. Eksponere en række properties for viewet, som viewet derefter binder til. I sin constructor initialisere den det repository der bliver brugt. Resten af tiden tilgås det valgte repository via et interface. Det er denne klasse der står for det meste af applikationslogikken.

## ButtonProperties

En attached property der bruges til at få en knap til at sætte DislogResult på dens vindue til true. Et godt eksempel på at MVVM er besværligt, samme funktion kunne laves i click handleren for knappen i code behind i en linje.

## DoubleClickProperty

Knytter et dobbeltklip event til en ICommand. Klassen skal bruges så man i XAML kan binde en command til dobbeltklik.

## IViewFunctions

Interface der skjuler View specifikke funktioner som ViewModel’en er nødt til at kalde. Den er her for at man er i stand til at skille View og ViewModel ad i flere assemblies.

## ViewFunctions

Implementation af IViewFunctions. Giver ViewModel nogle muligheder for at interagere med Viewet uden at kende det view den arbejder med. Her beder ViewModel’en for eksempel om at Viewet skal editere en NumberTranslation og give resultatet retur.

## ViewModelLocator

Bliver brugt til at et Window kan lokalisere sin ViewModel. I denne applikation bliver den kun brugt af MainWindow der i XAML binder din DataContext til MainWindowViewModel på denne klasse. Klassen initialiseres i App.xaml, og er på den måde point of entry for alt det bagvedliggende. Der bruges Lazy<T> så ViewModel’en først konstrueres når den skal bruges. Dette giver nok mest mening hvis ViewModelLocator skal lokalisere mange ViewModels. Lazy<T> garantere også at der kun er en instans af klassen og at flere tråde ikke kan påvirke dette.

## E164ValidationRule

En regl der kan bruges når man binder til en property på en ViewModel der skal overholde E164 (altså f.eks. +4588888888)

## FromNumberValidationRule

En regl der kan bruges når man binder til en property på en ViewModel der skal overholde Danish Crowns standart for mobil dæknumre (altså 18XXXX)

# Perspektivering

Jeg i en længere periode haft utroligt travlt på jobbet hvilket har resulteret i at jeg har haft begrænset tid til denne opgave. Produktet er lavet på omkring en arbejdsdag. Jeg har ikke lavet tests og jeg er allerede bekendt med en del fejl. Alligevel synes jeg at det er en ok applikation der kan opfylde mine krav og hjælpe mig med at demonstrere vores samlede løsning over for ledelsen. Nogle gange handler programmering også om at få et produkt færdigt. Det vigtigste er at man ikke glemmer at komme retur og rette ens fejl og mangler på et senere tidspunkt.

Særligt vil jeg gerne lave om i MainWindowViewModel. Den er blevet for stor og krøllet. En del logik må kunne tages ud i andre klasser, for metoderne i klassen har ikke en ret høj sammenhørighed. Klasssen burde også kun have et formål, altså at eksponere properties for Viewet resten af logikken har ikke noget at lave der.

NumberTranslationRepositorySqliteDemo.GetNumberTranslations laver to kald til databasen. Første kald for at se hvor mange rækker kaldet ville have returnere og andet kald for at bede om de rækker der ligger inden for den side der bliver vist. Dette er for at kunne regne ud hvor mange sider kaldet er på så viewet kan give dette valg til brugeren. Dog til mit forsvar er Sqlite lokal og vildt hurtig så det er ikke noget problem her. I Mysql versionen skal der bruges en anden måde at håndtere dette på, men jeg mener også at Mysql har funktioner til at håndtere dette.

Generelt for begge implementationer af repositoriet gælder at de har flere ansvar og flere grunde til at blive ændret. Der er kode gentagelser der kan undgås. Der burde laves en klasse det bygger SQL statements. Som det er nu, med kun et repository, går det nok, men hvis der var flere ville det ikke være let at ændre et tabelnavn, navnet på en kolonne eller lignende. Dette må anses for at være dårlig stil.

Resten er sådan set ok. Særligt synes jeg min NotifyBase klasse er fin, dog måske lidt ”gold plated”. ButtonProperties er jeg også glad for, da den løser et problem jeg tidligere har været nødt til at lave i code behind.

Generelt kunne jeg godt have arbejdet mere med at gøre min kode clean, men det ligger ofte i efterbehandlingen og det er jeg ikke kommet til endnu.

# Konklusion

Opgaven er løst dog er der plads til forbedringer. Man kan tilføje, fjerne og rette nummeroversættelser og man kan søge. Applikationen arbejder tæt med databasen for at overføre så lidt data som muligt mellem klient og server.

Applikationen har et tåleligt udseende og en nogenlunde fungerende brugergrænseflade. Jeg vil absolut mene at jeg kan bruge denne applikation som en første version i næste fase af projektet hvor vi skal have lavet POC på det færdige produkt, altså frontend og backend samlet set.

Der er ingen i organisationen der skuler til at der er fejl i en applikation lavet på en arbejdsdag, det skal naturligvis ændres, men min vurdering er at hele applikationen bliver ændret til noget web baseret. Den får givet vis også flere funktioner så som at afspejle de samme data i AD og hos Telia så der kun skal rettes et sted og ikke 3.