RoomReservation

Bachelor projekt i Softwareteknologi

Af
Jakob Skriver

Danmarks Tekniske Universitet
13.Juli 2015

1 Abstract

The project has resultet in the site Room Reservation, which handles reservation for a selection of DTU's databars. This is done by importing all reservations to a database from excel files given. From that database the site can search through all the relevant reservations for the databars. When a search has returned with a result, the celltable shows each reservation. When a reservation is selected in the results the week schedule updates, so that the course number and room number is set for the apropriate hours of the day.

Indhold

1 Abstract **1**

2 Introduktion **2**

3 Design **2**
3.1 Use Cases . . . 3
3.2 Detaljerede use cases . . . 4

4 Implementering **8**
4.1 Clientsiden . . . 8
4.2 Serversiden . . . 9
4.3 Detaljerede klassediagrammer for alle klasser . . . 9
4.4 Import . . . 12
4.5 Søgning . . . 12
4.6 Ugeskema . . . 13

5 Tests **14**

6 Resultater **15**

7 Diskussion **16**
7.1 Generelt . . . 16
7.2 Ekstra funktionalitet . . . 16
7.3 Perspektiv . . . 16

8 Konklusion **17**

9 Bilag **I**

Appendix **I**
Kode . . . I
Clientside . . . I
Serversiden . . . I
Model . . . XI

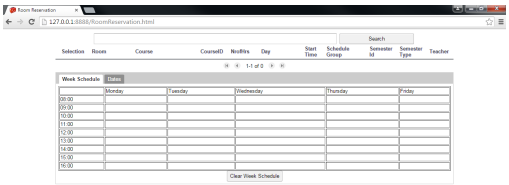
2 Introduktion

Målet med dette projekt er at designe en hjemmeside ved hjælp af Google Web Toolkit. Denne hjemmeside skal håndtere reservationer af lokaler, der modtages i excel filer. Derved skal koden importere informationen fra excel filerne til en database. Det helt grundlæggende mål med siden er at kunne vise reservationer uden at brugeren mister overblikket. Men den funktionalitet der bruges ligges der op til mange andre anvendelser af siden, som f.eks underviseren der skal finde et ledigt lokale. Strukturen af Gui'en m.m kunne estate eller tilføje til studieplanlægger funktionaliteterne på DTU's nuværende side. For at siden skal bliver ved med at være brugbar skal der gives excel filer hvert semester.

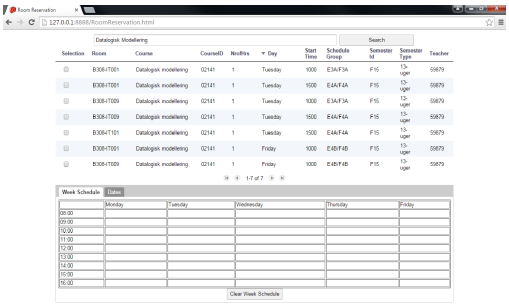
3 Design

DTU har en hjemmeside der viser en lokaleoversigt over alle kurser. Denne hjemmeside benytter sig af en lang liste af alle kurser, hvor overblikket forsvinder meget hurtigt. Dette gør siden meget tung at navigere rundt på. Hvert kursus er et link til en side med endnu en liste over alle datoer med information omkring lokale(r) og dag(e). Denne visningsmetode gør at brugeren skal arbejde meget for at få den information og specielt meget for at få et overblik over bare et kursus, samt det er en lang procedure hvis brugeren skal lave en oversigt over hele semesteret. Funktionalitet i forhold til at give lærere et overblik over hvilke lokaler er ledige eksisterer ikke i på denne hjemmeside.

Min design tankegang, bygger på at hjemmesiden skal være skalerbar uden at brugeren mister overblikket eller funktionaliteten. For at overblikket ikke forsvinder med skalaen, bruges en søge mekanisme der giver en stor flexibilitet i datainputet fra brugeren, samt det giver en minimalistisk Gui interface i forhold til den information som kan udvindes fra funktionaliteten. Resultatet af søgningerne kan bliver overvældende og derfor kan brugeren miste overblikket, med dette i tankerne bliver søgeresultaterne opdelt i sider af ti resultater. Dette gør at overblikket fastholdes selv med mange resultater fra søgningen. Det, der er med til at gøre DTU's nuværende hjemmeside tung at bruge, er at der skal stadigt arbejdes for at få et overblik over hvor på ugen og i hvilket lokale(r) kurset finder sted, når brugeren har fundet det korrekte kursus. For at gøre det nemt at få overblikket vises valgne fra søgeresultatet i et ugeskema. Dette gør, at brugeren selv kan skabe sit ugeskema eller udfra tomme celler finde ledige tidspunkter i ugen.



Figur 1: Startsiden uden søgeresultater

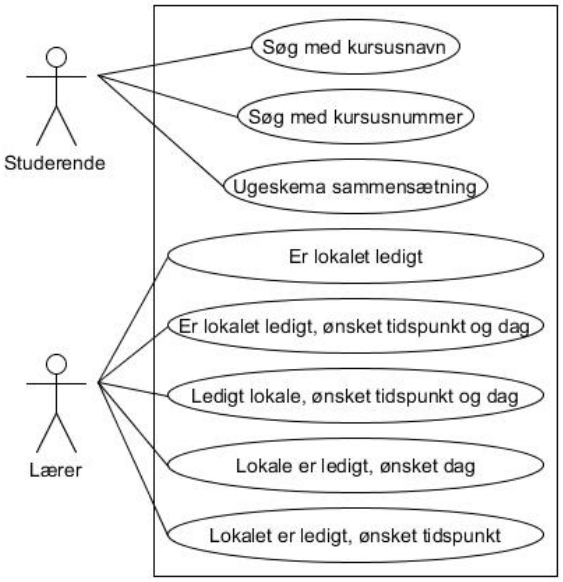


Figur 2: Startsiden med søgeresultater for Datalogisk Modellering

Hjemmesiden er centreret omkring søgefeltet, da alle funktionaliteter starter med en søgning. Derfor er søgefeltet og søgeknappen i toppen af hjemmesiden. Efter en søgning er fortaget bliver resultatet vist som en liste af reservationer under søgefeltet og søgeknappen. Under listen af søgeresultater vises et panel med tabs. I den første tab vises ugeskemaet, dette skema ændres i forhold til de valgte reservationer i søgeresultatet.

3.1 Use Cases

Selvom min hjemmeside er forholdsvis enkel, er der en del interessante use cases. Da der er datainput i form af et søgefelt, er der næsten ubegrænsede muligheder for input fra brugeren. Use casene tager udgangspunkt i de to type brugerer som hjemmesiden har. Den første bruger er den Studerende, som gerne vil vide hvilket lokale hans/hendes kurser undervises i eller skabe en ugeskemaoversigt over den sine kurser. Den anden type bruger er Læreren, der bruger hjemmesiden til at finde ud af om der er plads i det pågældende lokale. Dette kan ske med flere forskellige datainput fra brugeren, hvor det ændres i forhold til hvormeget læreren ved eller hvor specifikt læreren er omkring ønske om lokale, tid og dag. En oversigt over mine use cases kan ses nedenfor.



Figur 3: Use Case Diagram

3.2 Detaljerede use cases

USE CASE 1

Beskrivelse

En studerede vil finde lokalet for et kursus med kursusnavn eller del af kursusnavnet.

Aktør

Studerende

Forudsætning

Korrekt kursusnavn eller del af kursusnavn

Hovedscenarie

Brugeren skriver kursusnavnet eller delen af kursusnavnet i søgefeltet og derefter kan brugeren finde lokalet for kurset i søgeresultatet.

Alternative scenarier

1. Brugeren har valgt et kursus i forvejen. Brugeren trykker på clear week schedule knappen for at reset ugeskemaet. Brugeren søger nu på kurset på ny.

USE CASE 2

Beskrivelse

En studerede vil finde lokalet for et kursus med kursusnummer.

Aktør

Studerende

Forudsætning

Korrekt kursusnummer eller del af kursusnummer

Hovedscenarie

Brugeren skriver kursusnummer eller del af kursusnummer i søgefeltet og derefter kan brugeren finde lokalet for kurset i søgeresultatet.

Alternative scenarier

1. Brugeren har valgt et kursus i forvejen. Brugeren trykker på clear week schedule knappen for at reset ugeskemaet. Brugeren søger nu på kurset på ny.

USE CASE 3

Beskrivelse

En studerende vil sammenligne kurser eller skabe sit ugeskema

Aktør

Studerende

Forudsætning

Korrekt kursusnavn eller del af kursusnavn eller korrekt kursusnummer eller del af kursusnummer for hvert kursus

Hovedscenarie

Brugeren søger på enten kursunavn eller kursusnummer for et kursus og finder derefter kurset i søgeresultatet. Når det korrekte kursus er fundet vælger brugeren kursus ved at clicke i checkbox cellen i selections kolonnen. Når kurset er valgt vises kursusnummeret og lokalet i ugeskemaet i det eller de felter som svare til kursus dagen, tiden og antal timer. Dette gøres for hvert kursus brugeren.

Alternative scenarier

1. Hvis brugeren har valgt et forkert kursus eller et kursus for meget trykker brugeren på clear week schedule knappen, hvor ugeskemaet bliver reset. Brugeren søger derefter på hvert kursus og vælger det i søgeresultaterne for at vise kurset i ugeskemaet.

USE CASE 4

Beskrivelse

En lærer vil finde ud af hvornår lokalet er frit.

Aktør

Lærer

Forudsætning

Kender eller ønsker lokalet.

Hovedscenarie

Brugeren skriver lokale i søgefeltet og trykker på search. Brugeren vælger alle reservationerne i søgeresultatet. Dette giver en oversigt over hvilke tidspunkter på hvilke dage lokalet er ledigt. Brugeren kan derved se hvilke tidspunkterpå hvilke dage det pågældende lokale er ledigt.

Alternative scenarier

1. Brugeren har valgt et kursus i forvejen. Brugeren trykker på clear week schedule knappen for at reset ugeskemaet. Brugeren søger derefter på lokalet.

USE CASE 5

Beskrivelse

En lærer vil finde ud af om lokalet er reserveret den pågældende dag og tid

Aktør

Lærer

Forudsætning

Kender eller ønsker lokalet, tid og dag

Hovedscenarie

Brugeren skriver lokale mellemrum tid mellemrum dag i søgefeltet og trykker på Search. Hvis der findes reservationer for det lokale på den dag til den tid bliver de vist i søgeresultatet.

Alternative scenarier

1. Brugeren har valgt et kursus i forvejen. Brugeren trykker på clear week schedule knappen for at reset ugeskemaet. Brugeren søger derefter på lokalet, tid og dag

USE CASE 6

Beskrivelse

En lærer vil finde ud af, om der er et ledigt lokale den pågældende dag og tid

Aktør

Lærer

Forudsætning

Kender eller ønsker tid og dag

Hovedscenarie

Brugeren skriver tid mellemrum dag i søgefeltet og trykker på search. Dette giver en oversigt over alle de lokaler der er reserveret den pågældende dag og tid. Brugeren kan derved se, hvis et lokale ikke har en reservation.

Alternative scenarier

1. Brugeren har valgt et kursus i forvejen. Brugeren trykker på clear week schedule knappen for at reset ugeskemaet. Brugeren søger derefter på tid og dag.

USE CASE 7

Beskrivelse

En lærer vil finde ud af, om lokalet er ledigt den pågældende dag

Aktør

Lærer

Forudsætning

Kender eller ønsker lokalet og dag.

Hovedscenarie

Brugeren skriver lokale mellemrum dag i søgefeltet og trykker på search. Brugeren vælger alle resevationerne i søgeresultatet. Dette giver en oversigt over hvilke tidspunkter på dagen lokalet er ledigt. Brugeren kan derved se hvilke tidspunkter på dagen det pågældende lokale er ledigt.

Alternative scenarier

1. Brugeren har valgt et kursus i forvejen. Brugeren trykker på clear week schedule knappen for at reset ugeskemaet. Brugeren søger derefter på lokalet og dag.

USE CASE 8

Beskrivelse

En lærer vil finde ud af, om lokalet er frit den pågældende tid.

Aktør

Lærer

Forudsætning

Kender eller ønsker lokalet og tid.

Hovedscenarie

Brugeren skriver lokale mellemrum tid i søgefeltet og trykker på search. Brugeren vælger alle resevationerne i søgeresultatet. Dette giver en oversigt over hvilke dage lokalet er ledigt for det pågældende tidspunkt. Brugeren kan derved se hvilke dage det pågældende lokale er ledigt.

Alternative scenarier

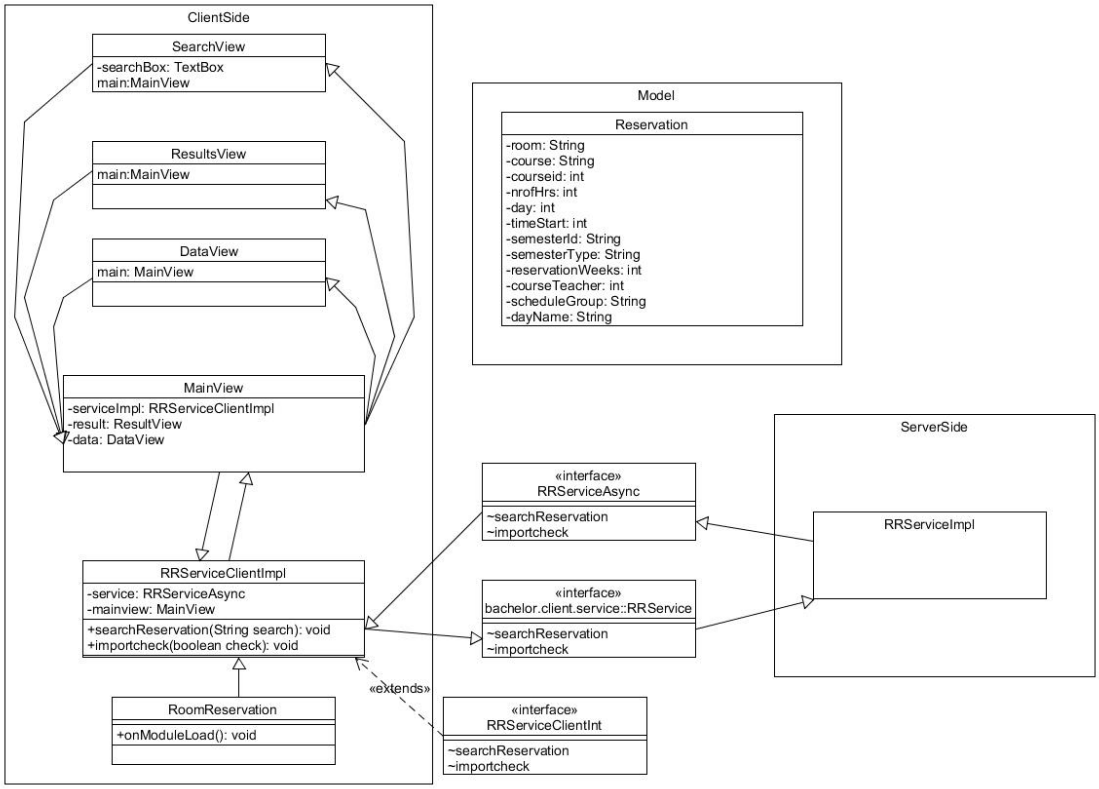
1. Brugeren har valgt et kursus i forvejen. Brugeren trykker på clear week schedule knappen for at reset ugeskemaet. Brugeren søger derefterpå lokalet og tid.

4 Implementering

Til implementeringen af designet benyttes Google Web Toolkit. GWT er et udviklingsredskab til at bygge og optimere komplekse browserbaserede applikationer. Formålet med dette web redskab er at give udvikleren mulighed for at lave høj præstations web applicationer. Uden at være en ekspert i browser specifikke udviklings redskaber, så som CSS, HTML og Javascript. En fordel ved at benytte GWT er, at koden kan køres på alle browsere.

Google Web Toolkit tager udgangspunkt i en clientside og en serverside. Kommunikationen mellem disse to sider sker igennem hvad GWT kalder Remote Procedure Calls (RPC). Når et RPC bliver kaldt af clientsiden, sendes der besked til serversiden om hvilken kode, der skal køres. RPC er en asynkron procedure, hvilket betyder, at clientsiden kan køre kode, mens den kode som RPC'et kalder på serversiden kører. RPC's sker igennem to interfaces kaldet Service og ServiceAsync, hvor Service interfaces sender kaldet og de pågældende informationer ned til serverdelen og ServiceAsync bliver kaldt når serversiden er færdig med at køre koden afhængig af RPC'et.

Sammenspillet mellem de forskellige dele og klasserne af programmet kan ses i klassediagrammet.



Figur 4: Klassediagram

4.1 Clientsiden

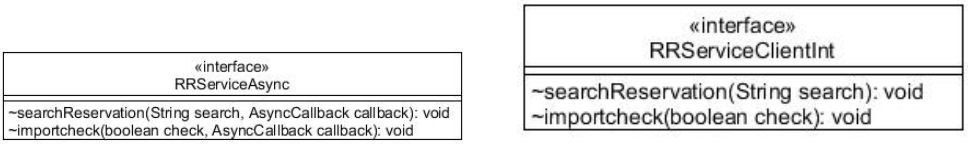
Denne del af programmet står for GUIet og for at sende information til serveren gennem RPCs, samt håndtere svaret fra serveren og hvordan informationen skal vises. Clientsiden af programmet består af 8 forskellige klasser. ServiceClientImpl er hovedklassen, den er ansvarlig for at sende RPC til serveren og håndtere svaret. Klassen sender informationen fra RPC til MainView klassen. MainView klassen giver informationen videre til ResultView der viser informationen i et Celltable. Når der sker en selektion i ResultViewets celltable, kaldes en metode der sender informationen omkring alle selektioner først til MainView. MainView sender derefter informationen til DataView, som viser de forskellige reservationer i forhold til dagen, tidspunktet og antal timer.

4.2 Serversiden

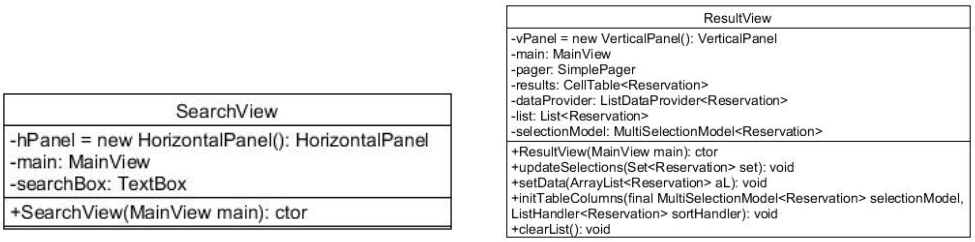
Denne del af programmet bruges til at lave de beregningstunge og komplekse interaktioner med f.eks. databaser. Alle interaktioner mellem programmet og databasen sker på serversiden. ServerImpl klassen er den eneste klasse på serversiden, da næsten alle serversidens opgaver ligger i at interagere med databasen, er implementationen nemmere i en klasse.

Interaktionen med databasen sker i flere forskellige former. Den første form er igennem INSERT kaldet for databasen, dette kald bliver brugt, når der bliver sendt et RPC om import. Import metoden læser informationen fra de to Excel filer, som er givet. Metoden udvinder den korrekte information og fjerner den unødvendige tekst før den insætter informationen i databasen. Den anden form for interaktion er gennem SELECT kaldet til databasen. Dette kald bliver eksekveret, når en søgning fra clientsiden sker, og RPC for søgningen bliver modtaget af seversiden. Før kaldet bliver lavet, kontroller metoden søgeteksten, der håndterer kaldet i forhold til hvilken sammensætning søgeteksten havde.

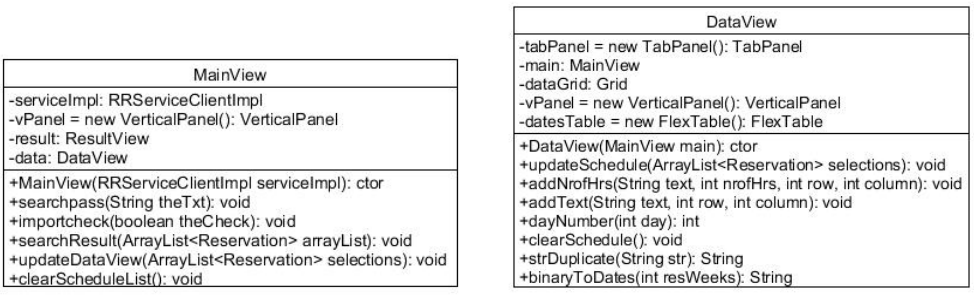
4.3 Detaljerede klassediagrammer for alle klasser



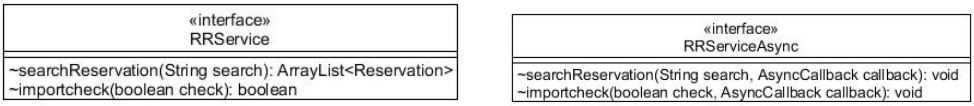
Figur 5: Fra venstre til højre: Klassediagram for ServiceClientImpl og ServiceClientInt



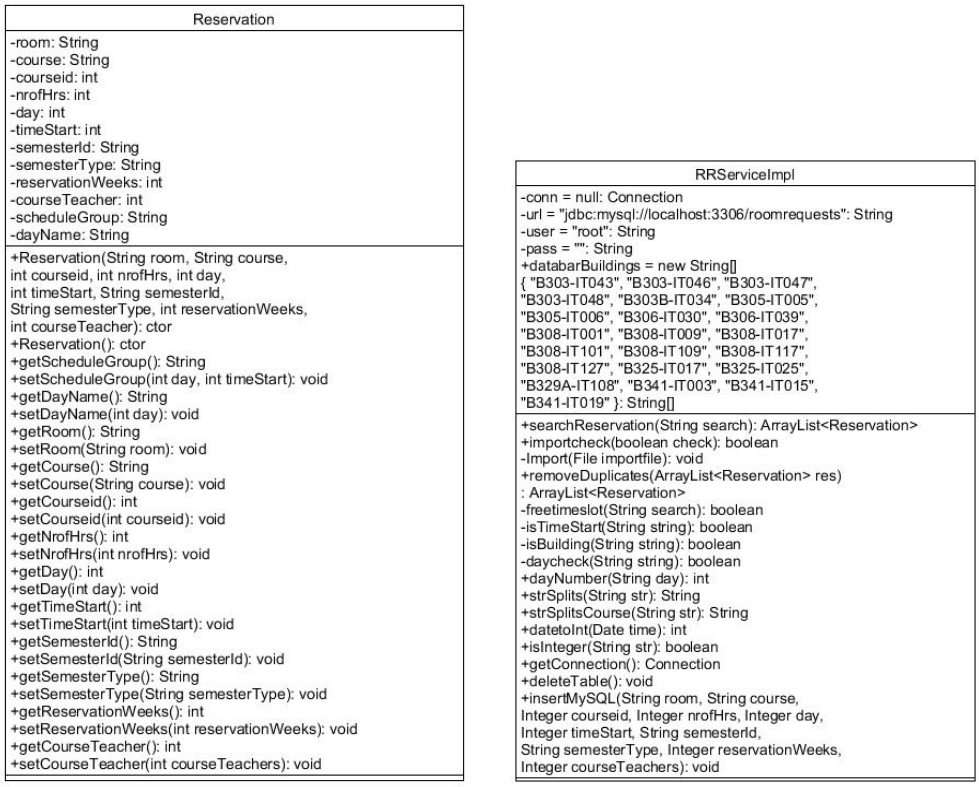
Figur 6: Fra venstre til højre: Klassediagram for SearchView og ResultView



Figur 7: Fra venstre til højre: Klassediagram for MainView og DataView



Figur 8: Fra venstre til højre: Klassediagram for Service og ServiceAsync



Figur 9: Fra venstre til højre: Klassediagram for Reservation og ServiceImpl

I denne sektion beskrives hjemmesidens klasser kort.

- MainView
  - Klassen administrerer GUI sammensætningen ved at være bindeledet mellem alle GUI klasser. Dette betyder, at al information, der er udvekslet mellem de forskellige dele af GUI Interfacet, bliver kørt igennem MainView klassen. Koden bliver enklere, idet hver GUI klasse ikke skal have en reference til hver klasse den interagerer med.
- SearchView
  - Denne del af GUI interfacet laver søgefeltet og søgeknappen, samt sender søgefelts informationen til MainView. Der sender den til ServiceClientImpl, som laver et RPC til søgemetoden på serversiden.
- ResultView
  - Denne del af GUI interfacet laver celltabellen. Når en søgning sker modtager klassen resultatet fra RPC kaldet. Klassen viser denne information i et Celltable, som sender de valgte reservationer til MainViewet hver gang, der checkes en checkbox i celltabellen.
- DataView
  - Denne del af GUI interfacet laver fanebladsvinduet, hvor det første faneblad viser et ugeskema. Ugeskemaet ændres i forhold til hvilken information, der modtages fra MainViewet, efter der er

fortaget et valg i ResultViews celltable. For at bringe ugeskemaet tilbage til starttilstanden bruges knappen Clear Week Schedule, som både fjerner teksten fra alle de pågældende dele af ugeskemaet og sender et kald til ResultViewet for at tømme listen af de valgte reservationer.

- ServiceClientImpl
  - Denne klasse håndterer RPC kaldene til serversiden. Klassen modtager kald fra MainViewet med input data fra brugeren, hvorefter den laver et RPC kald med den pågældende information. Når RPC kaldene returnerer, kalder denne klasse metoderne i forhold til hvilket form for returtype der modtages.
- ServiceClientInt
  - Dette interface bruges til at være sikker på, at ServiceClientImpl har de samme metoder, som bruges i RPC kaldene.
- Reservation
  - Dette er objektmodellen. Dette objekt bruges igennem hele programmet. Derved eksisterer denne klasse udenfor client-server strukturen. Objektet sætter to forskellige værdier, når en reservation er fundet i databasen. De to værdier er dagnavnet og skemagruppen.
- ServiceImpl
  - Klassen bruges til at kommunikere med databasen. Når der modtages et import RPC, gennemgår koden Excel filerne. Hvergang den finder en linje med en bygning og lokale, som passer til de specificerede bygninger og lokaler, bliver InsertMySQL metoden kaldt. Værdierne fra de korrekte Excel celler bliver hentet og brugt i metoden. Ved et søgnings RPC kører koden searchReservation metoden med den tekst, som er modtaget fra RPC kaldet.
- Service
  - Interfacet bruges af Google Web Toolkitet når der laves et RPC. Dette interface bliver kaldt af ServiceClientImpl.
- ServiceAsync
  - Dette interface er den anden del af kommunikationen, der finder sted i et RPC. Interfacet bliver kaldt når ServiceImpl klassen returnerer RPC kaldet.

4.4 Import

Denne funktionalitet bliver udført på serversiden. For at kunne sende informationen til databasen skal koden først hente den korrekte information fra Excel filerne. Syntaxen for en linje i Excel filerne er, som vist i tabellen.

Tabel 1: Excel datastruktur.

12879	B341-IT003	02161 Software Engineering 1 F15	2161	4	1000000	15:00
F15	13-uger	11111111001111100000000000000000000000000000000000	1	29-12-2014	34435	

For at finde den korrekte information skal der først udvælges de korrekte bygnings- og lokalesammensætninger. Koden går gennem den kolonne, som indeholder bygnings- og lokaleværdierne , og sammenligner teksten i den kolonne med et string array. Arrayet indeholder alle de lokaler, som er blevet udvalgt til at blive sendt til databasen. Hvis kolonnen finder et lokale, som findes i arrayet, indsamler koden alle kolonnerne i rækken. Informationen i kolonne to til ti og fjorten bliver hentet af koden. Den tekst, der kommer fra den tredje kolonne, bliver skåret ned, så kun den brugbare information bliver sendt videre. Kursusnummeret bliver fjernet fra starten af teksten ved hjælp af splitmetoden for en string. Hvis der findes F15 eller lignene bag kursusnavnet bliver dette også fjernet ved hjælp af splitmetoden for en string. Der bliver lavet et check om den næstsidste karakter er et tal, fordi ikke alle celler i Excel filerne har F15 tilsidst i teksten.

Syntaksen for en linje i databasen er, som vist i tabellen.

Tabel 2: Database datastruktur.

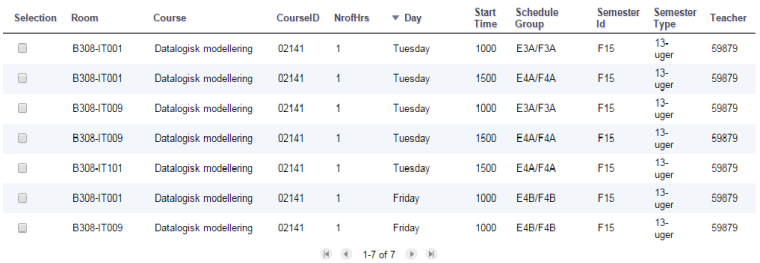
B341-IT003	Software Engineering 1	2161	4	1000000	1500	F15	13-uger	2147483647	34435

Når den korrekte information er fundet kalder import InsertMySQL metoden, som sammensætter et INSERT kald til databasen. Det kan ses fra database syntaksen og Excel syntaksen at der bliver fjernet en del af den unødvendige tekst, og der bliver udvalgt, hvilke celler der er nødvendige. Den binære string, som repræsenterer, hvilke uger reservationen er gældende for, bliver lavet om til en integer for lettere at håndtere det igennem progammet. Den sidste kolonne i databasen er underviserens id nummer. Når denne information bliver hentet fra Excel filerne, er det enten et tekst eller tal felt. For at håndtere denne problematik kontrollerer koden om cell typen er text eller integer. Når en række er valgt, bruges to forskellige kald af InsertMySQL metoden. Det ene kald spliter teksten og caster den første værdi til integer.

4.5 Søgning

Denne funktionalitet starter ved, at der trykkes på søgeknappen. Søgefeltteksten sendes til MainViewet, derefter til ServiceClientImpl, og endelig laves der et RPC kald til serversiden. Søgemetoden på serversiden kontrollerer hvilken type søgning, der skal laves i forhold til, hvordan søgeteksten er sammensat. De muligheder, der er for søgninger, kan findes i nedenstående oversigt.

- Søgnings syntaks muligheder
  - B308-IT009 eller en del af lokalet
  - Datalogisk modellering eller en del af navnet
  - 2141 eller en del af navnet
  - B308-IT009 1000 Tuesday
  - 1000 Tuesday
  - B308-IT009 1000
  - B308-IT009 Tuesday



Selection	Room	Course	CourseID	NrofHrs	Day	Start Time	Schedule Group	Semester Id	Semester Type	Teacher
	B308-IT001	Datalogisk modellering	02141	1	Tuesday	1000	E3A/F3A	F15	13-uger	59879
	B308-IT001	Datalogisk modellering	02141	1	Tuesday	1500	E4A/F4A	F15	13-uger	59879
	B308-IT009	Datalogisk modellering	02141	1	Tuesday	1000	E3A/F3A	F15	13-uger	59879
	B308-IT009	Datalogisk modellering	02141	1	Tuesday	1500	E4A/F4A	F15	13-uger	59879
	B308-IT101	Datalogisk modellering	02141	1	Tuesday	1500	E4A/F4A	F15	13-uger	59879
	B308-IT001	Datalogisk modellering	02141	1	Friday	1000	E4B/F4B	F15	13-uger	59879
	B308-IT009	Datalogisk modellering	02141	1	Friday	1000	E4B/F4B	F15	13-uger	59879

1-7 of 7

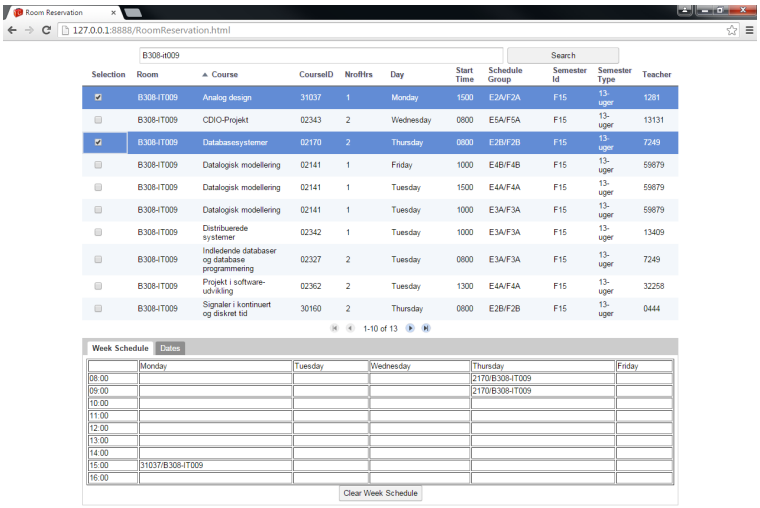
Figur 10: Søgeresultat i Celltabellen

Koden kontrollerer søgetekstens sammensætning ved brug af splitmetoden for en string med mellemrum som splitkarateren. Når splitmetoden er blevet eksekveret, checkes hvert af elementerne i splitarrayet. Sammensætningen af arrayet fastslår hvilket SELECT statement, der bliver sendt til databasen.

Resultatet fra en søgning vises i et Celltable med ti rækker pr side.

4.6 Ugeskema

Hver gang der bliver valgt en reservation i søgeresultaterne, kaldes et onSelectionChange Event. Dette event sender de valgte reservationer til DataViewet gennem MainViewet. Reservationerne bliver tilføjet til ugeskemaet i forhold til hvilken dag, hvilket tidspunkt og hvor mange timer lokalet er reserveret i. Den tekst, der bliver sat i ugeskemaet, er kursets nummer og det reserverede lokale. Der kan stå flere reservationer i hver celle af ugeskemaet. Ved hjælp af splitmetoden laves et array af alle reservationer i den celle. Hvis der findes den samme reservationsstring fjernes duplikater.



Figur 11: Data i Ugeskemaet

Denne figur viser, hvordan den sammensatte tekst ser ud, og hvorledes en reservation med flere timer bliver tilføjet. For hver reservation gennemgåes tre metoder for at sætte den korrekte tekst det rigtige antal gange i ugeskemaet. Den første metode hedder updateSelection, og den bestemmer rækketallet i forhold til hvilket tidspunkt reservationen har. Den anden metode hedder addnrofHrs og bestemmer hvormange gange, den næste metode skal kaldes. Dette betyder, at jo flere timer der er reserveret, jo flere gange kaldes den næste metode. Den tredje og sidste metode hedder addText, og den sætter teksten i forhold til, om cellen er tom eller ikke. Hvis den er tom, sættes teksten til reservationen, og hvis den ikke er tom, gennemgåes teksten for duplikater, hvorefter teksten bliver sat.

5 Tests

I dette afsnit beskrives de test cases hjemmesiden er blevet kørt igennem. Testene er baseret på Use Casene og dækker begge brugerne. Testene er fortaget i Chrome browseren, med en localhost database server fra Wamp.

Tabel 3: Funktionalitet: Netværk kommunikation
Beskrivelse: Test af netværk kommunikationen mellem app og backend. Disse test er delt foretaget som hands-on tests.

Test Case	**Beskrivelse**
A	En studerede søger med en del af et kursusnavn og vælger kurset.
B	En studerede søger med et kursusnummer og vælger kurset.
C	En studerende skaber sit ugeskema
D	En lærer vil finde ud af, hvornår et lokalet er frit.
E	En lærer vil finde ud af, om lokalet er reserveret den pågældende dag og tid
F	En lærer vil finde ud af, om der er et ledigt lokale den pågældende dag og tid
G	En lærer vil finde ud af, om lokalet er ledigt den pågældende dag
H	En lærer vil finde ud af, om lokalet er frit den pågældende tid.

Tabel 4: Forventet output.

Test Case	**Forudsætninger**	**Forventet output**
A	Korrekt del af kursusnavn	Det korrekte kursus i resultat og vist i ugeskema
B	Korrekt kursusnummer eller del af kursusnummer	Det korrekte kursus i resultat og vist i ugeskema
C	Korrekt kursusnavn eller korrekt kursusnummer for hvert kursus	Et Ugeskema med hvert kursus vist
D	Kender eller ønsker lokalet.	Et Ugeskema med alle resevationer for det lokale
E	Kender eller ønsker lokalet, tid og dag	Et resultat med mulige reservationer
F	Kender eller ønsker tid og dag	Et sorteret resultat i forhold til lokale
G	Kender eller ønsker lokalet og dag.	Et ugeskma med alle lokalets reservationer for dagen vist
H	Kender eller ønsker lokalet og tid.	Et ugeskema med alle lokalets reservationer på tidspunktet gennem ugen

6 Resultater

Resultaterne af de forskellige tests

Tabel 5: Forventet output givne forskellige forudsætninger.

Test Case	Input	Output
A	Model og valgt Datalogisk modellering	Alle 6 reservationer vises i resultat og ugeskema
B	31610 og valgt Applied Signal Processing	Reservationen for faget vist i ugeskema
C	02637, Databasesystemer, 02224 og Analog design	Alle 4 fag er vist i ugeskema
D	B308-IT009	Ugeskemaet med alle reservationer
E	B308-IT001 1500 Monday	Ingen reservationer vist derfor ledigt
F	1500 Monday	Liste over alle reservationer Mandag kl 1500
G	B308-IT017 Monday	Begge reservationer vist i ugeskema
H	B308-IT001 1000	Alle 4 reservationer i lokalet kl 1000, to samme dag

Alle testresultaterne er som forventet undtagen den sidste test, hvor der var to reservationer på samme tid samme dag i samme lokale. Dette scenarie er der ikke taget højde for i implementationen.

7 Diskussion

7.1 Generelt

Pågrund af fokus på andre aspekter af projektet er nogel funktionaliteter ikke blevet helt som de var tænkt eller mangler noget af den tiltænkte funktionalitet. Import funktionen henter alt det betydningfulde information fra excel filerne og indsættet den i databasen. Der er to værdier som hver reservation har der ikke er i databasen, det er dagnavnet og skemagruppen. Disse to Stringe bliver sat under søgningen. Den funktionalitet som var tiltænkt var at disse værdier blev givet ved informationen i excel filen og derefter sendt med til databasen, dog var det lettere i øjeblikket at sætte disse værdier senere.
I forhold til skalebarhed af databasen giver det mening ikke at indsætte disse værdier i databasen. Eftersom alt den information til at sætte disse værdier findes i databasen i forvejen og flere værdier bruger mere plads. Hvis der er flere end en underviser på kurset bliver de ikke taget med fra excel filen, da koden kun tager det første underviser id. Beslutningen omkring dette blev taget fordi mængden af arbejde for at tage alle underviserne med ville overskygge værdien der ville blive tilført til projektet.

De excel filer der importeres fra har alle undervisernes ønsker om lokaler. Dette betyder at der kan opstå flere reservationer til et lokale til samme tid på samme dag. Hjemmesiden tager ikke hånd om dette. Måden dette kunne implementeres på ville være at under importen indsætte reservationer med samme lokale, tid og dag ind i en anden tabel. Efter importen ville der komme en dialogbox af en art op, hvor brugeren skulle vælge mellem reservationerne. Den valgte reservation ville efterfølgende bliver tilføjet til den almindelige database.

7.2 Ekstra funktionalitet

I gennem en udvikling process finde man på den masse forskellige features som man gerne vil have i et program/hjemmeside. Deadlinen er en faktor man bliver nød til at tage højde for, derfor kan alt den funktionalitet man gerne vil have lavet ikke komme med i den endelige version af programmet/hjemmesiden. I dette afsnit gennemgår jeg nogle af sådan funtkionaliteter eller udvidelser.

En tilføjelse til Tabpanel funktionaliteten, som ugeskemaet ligger i, ville være at tilføje en tab med en liste over de datoer hvor kurset ikke ligger på den pågældende dag eller tid. Denne tab ville tage hånd om de specielle skema dage der bliver flytte pågrund af heligdage osv. Eftersom mængden af disse datoer aldrig vil være stor giver en liste god mening at implementere funktionaliteten med. Denne funktionalitet ville give mulighed for at der kunne være reservationer som ikke galt i hele semestret. Der ville være en grænse for hvor mange dage der kunne reserveres, før det blev for hele semestret. Et problem ville dog være at gøre brugern opmærksom på når et kursus visse med specieller datoer. Udfra hvad jeg ved om Google Web Toolkit ville jeg ændre fanebladet til Dato fanen, når et kursus med specieller datoer blev valgt.

I Celltabellen kan der være situationer hvor brugeren bliver nød til at vælge 10+ reservationer. Denne unødvendige gene kunne være forbygget med en checkbox der valgte alle søgeresultaterne. Denne checkbox ville intuitivt placeres i overskiften for kolonnen, for at brugeren let kan finde den og for ikke at ødelægge Gui designet. I forsøget på at implementere netop sådan en checkbox gik det op for mig, at alle funktionaliteter er ikke lige nemme at lave i Google Web Toolkit. Den Use case som ville få mest værdi fra denne funktionalitet er læreren der vil finde ud af om lokalet er ledigt, eftersom informationen ligger i de ikke eksisterende reservationer og derved tomme pladser i ugeskemaet.

I projektet er der blevet fokuseret på funktionaliteterne og håndteringen af database interaktionerne, derfor er GUI'et relativt simpelt. Eftersom jeg ikke har arbejdet med GUI'en så meget, ved jeg ikke hvor advanceret GWTs Gui interface kan bliver. Dog ville jeg meget gerne have lavet et GUI der så mere færdigt ud.

7.3 Perspektiv

Funktionalitetten af denne hjemmeside tager faktisk fra to forskellige DTU hjemmesider, den første er selvfølgelig Lokaleoversigts siden og den anden er studieplanlægger siden. Disse to sider har to meget specifikke formål og kan ikke andet end formålet. Min hjemmeside har en meget større mulighed for udvikling og skalerbarhed, eftersom der håndteres mange flere funktionalitets ønsker uden at GUI'en eller den underliggende struktur skal ændres.

Som start på projektet handle det meget om hvordan Google Web Toolkit fungerede og hang sammen. Når man sætter et projekt op skal der laves en del ændringer til forskellige xml filer m.m. Dette kan godt være lidt afskræmmene for nye programmører. Men når alt setup er færdigt og man har en ide om hvordan GWT strukturen fungere, er det lige så nemt at udvikle i som almindelig Java programmer. GWT har nogle restraktioner i forhold til Java, men det er ikke noget jeg er støt på. Specielt som en java eller c programmør er det nemt at arbejde med, det mest fantastiske ved dette værktøj findes i at programmøren ikke behøver at kunne kode i javascript eller html. En lille smulle CSS kodning bruges til at style ens hjemmeside.

Det system som giver de excel filer jeg har importeret fra, kan spille rigtigt godt sammen med hjemmesiden. Hvis man satte disse to sider sammen ville lærerne kunne sende reservations ænsker direkte til databasen istedet for at denne side bliver nød til at importere fra en excel fil med meget unødvendig information.

8 Konklusion

Projektet Room reservation har for mit vedkommende i høj grad været et forståelses forløb, specielt i starten af perioden. Meget af den program struktur som Google Web Toolkit tvinger programmøren til at bruge har været den største forståelses opgave. Som design koncept har jeg igennem hele projektet holdt fast i at slutresultatet skulle være skalerbart, hvilket gør at man skaber og tænker på en anden måde. Hvilket for mit vedkommende giver et mere fuldkomment program. Udfra hvad DTU har lige nu til at vise de samme informationer, gjorde jeg mig nogle tanker i starten af projektet. I forhold til at give brugeren et nemt og hurtigt overblik over den meget store mængde information der skal vises. Siden ville bliver meget bedre, hvis man slog den sammen med den side som håndterer undervisernes reservations ønsker. Dette ville give meget mindre arbejde og give hurtigere svar til underviserne. Sammen med denne sammenlægning ville funktionalitetet omkring duplikater af reservationer, være en uvurderlig feature for administrationen at have.

Figurer

1 Startsiden uden søgeresultater . . . 2
2 Startsiden med søgeresultater for Datalogisk Modellering . . . 2
3 Use Case Diagram . . . 3
4 Klassediagram . . . 8
5 Fra venstre til højre: Klassediagram for ServiceClientImpl og ServiceClientInt . . . 9
6 Fra venstre til højre: Klassediagram for SearchView og ResultView . . . 9
7 Fra venstre til højre: Klassediagram for MainView og DataView . . . 9
8 Fra venstre til højre: Klassediagram for Service og ServiceAsync . . . 10
9 Fra venstre til højre: Klassediagram for Reservation og ServiceImpl . . . 10
10 Søgeresultat i Celltabellen . . . 13
11 Data i Ugeskemaet . . . 13

Tabeller

1 Excel datastruktur. 12
2 Database datastruktur. 12
3 Funktionalitet: Netværk kommunikation Beskrivelse: Test af netværk kommunikationen mellem app og backend. Disse test er delt foretaget som hands-on tests. 14
4 Forventet output. 14
5 Forventet output givne forskellige forudsætninger. 15

9 Bilag

Appendix

Kode

Clientside

```
package bachelor.client.service;

import java.util.ArrayList;

import bachelor.client.gui.MainView;
import bachelor.client.model.Reservation;

import com.google.gwt.core.client.GWT;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.rpc.ServiceDefTarget;

public class RRServiceClientImpl implements RRServiceClientInt {
        private RRServiceAsync service;
        private MainView mainview;

        public RRServiceClientImpl(String url) {
                System.out.println(url);
                this.service = GWT.create(RRService.class);
                ServiceDefTarget endpoint = (ServiceDefTarget) this.service;
                endpoint.setServiceEntryPoint(url);

                importcheck(true);
                this.mainview = new MainView(this);
        }

        public MainView getMainView() {
                return this.mainview;
        }

        @Override
        public void searchReservation(String search) {
                this.service.searchReservation(search, new DefaultCallback());
        }

        @Override
        public void importcheck(boolean check) {
                this.service.importcheck(check, new DefaultCallback());
        }

        private class DefaultCallback implements AsyncCallback {

                @Override
                public void onFailure(Throwable caught) {
                        System.out.println("An Error has occured");
                }

                @Override
                public void onSuccess(Object result) {

                        if (result instanceof ArrayList) {
                                ArrayList<Reservation> aL = (ArrayList<
                                    Reservation>) result;
```

```
                                mainview.searchResult(aL);

                        } else if (result instanceof Boolean) {
                                Boolean importCheck = (Boolean) result;
                                //TODO
                                //mainview.
                        }

                }
        }

}


package bachelor.client.gui;

import java.util.ArrayList;

import bachelor.client.model.Reservation;
import bachelor.client.service.RRServiceClientImpl;

import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.VerticalPanel;

public class MainView extends Composite {
        private RRServiceClientImpl serviceImpl;
        private VerticalPanel vPanel = new VerticalPanel();
        private ResultView result;
        private DataView data;

        public MainView(RRServiceClientImpl serviceImpl) {
                initWidget(this.vPanel);
                this.serviceImpl = serviceImpl;
                this.vPanel.setSize("100%", "100%");
                this.vPanel.setHorizontalAlignment(HasHorizontalAlignment.
                    ALIGN_CENTER);

                SearchView search = new SearchView(this);
                this.vPanel.add(search);

                result = new ResultView(this);
                this.vPanel.add(result);

                data = new DataView(this);
                this.vPanel.add(data);
        }

        public void searchpass(String theTxt) {
                serviceImpl.searchReservation(theTxt);
        }

        public void importcheck(boolean theCheck) {
                serviceImpl.importcheck(theCheck);
        }

        public void searchResult(ArrayList<Reservation> arrayList) {
                result.setData(arrayList);
        }
```

```
        public void updateDataView(ArrayList<Reservation> selections) {
                data.updateSchedule(selections);
        }

        public void clearScheduleList() {
                result.clearList();

        }

}


package bachelor.client.gui;

import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.TextBox;

public class SearchView extends Composite {
        private HorizontalPanel hPanel = new HorizontalPanel();
        private MainView main;

        private TextBox searchBox;

        public SearchView(MainView main) {
                initWidget(hPanel);
                this.main = main;
                hPanel.setHorizontalAlignment(HasHorizontalAlignment.
                    ALIGN_CENTER);
                hPanel.setVerticalAlignment(HasVerticalAlignment.ALIGN_MIDDLE)
                    ;
                hPanel.setSize("65%", "50%");
                hPanel.setSpacing(1);
                //hPanel.setBorderWidth(1);

                searchBox = new TextBox();
                searchBox.setWidth("98%");
                hPanel.add(searchBox);

                Button searchBtn = new Button("Search");
                searchBtn.setWidth("95%");
                searchBtn.addClickHandler(new searchClickHandler());
                hPanel.add(searchBtn);
        }

        private class searchClickHandler implements ClickHandler {
                @Override
                public void onClick(ClickEvent event) {
                        String theTxt = searchBox.getText();
                        main.searchpass(theTxt);
                }
        }
}
```

```
package bachelor.client.gui;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.Set;

import bachelor.client.model.Reservation;

import com.google.gwt.cell.client.CheckboxCell;
import com.google.gwt.cell.client.NumberCell;
import com.google.gwt.core.client.GWT;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.i18n.client.NumberFormat;
import com.google.gwt.user.cellview.client.CellTable;
import com.google.gwt.user.cellview.client.Column;
import com.google.gwt.user.cellview.client.ColumnSortEvent.ListHandler;
import com.google.gwt.user.cellview.client.SimplePager;
import com.google.gwt.user.cellview.client.SimplePager.TextLocation;
import com.google.gwt.user.cellview.client.TextColumn;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.VerticalPanel;
import com.google.gwt.view.client.DefaultSelectionEventManager;
import com.google.gwt.view.client.ListDataProvider;
import com.google.gwt.view.client.MultiSelectionModel;
import com.google.gwt.view.client.ProvidesKey;
import com.google.gwt.view.client.SelectionChangeEvent;

public class ResultView extends Composite {

        private VerticalPanel vPanel = new VerticalPanel();
        private MainView main;
        private SimplePager pager;
        private CellTable<Reservation> results;
        private ListDataProvider<Reservation> dataProvider;
        private List<Reservation> list;
        private final MultiSelectionModel<Reservation> selectionModel;

        public ResultView(MainView main) {
                initWidget(vPanel);
                this.main = main;
                vPanel.setHorizontalAlignment(HasHorizontalAlignment.
                    ALIGN_CENTER);
                vPanel.setSize("80%", "70%");

                // Create Celltable
                results = new CellTable<Reservation>(KEY_PROVIDER);

                results.setAutoFooterRefreshDisabled(true);
                results.setAutoHeaderRefreshDisabled(true);

                // Pager
                SimplePager.Resources pagerResources = GWT
                                .create(SimplePager.Resources.class);
                pager = new SimplePager(TextLocation.CENTER, pagerResources,
                    false, 0,
                                true);
                pager.setDisplay(results);
```

```
        pager.setPageSize(10);

        dataProvider = new ListDataProvider<Reservation>();
        dataProvider.addDataDisplay(results);

        ListHandler<Reservation> sortHandler = new ListHandler<
            Reservation>(
                    dataProvider.getList());
        results.addColumnSortHandler(sortHandler);

        // add Selectionmodel
        selectionModel = new MultiSelectionModel<Reservation>(
            KEY_PROVIDER);
        selectionModel
                    .addSelectionChangeHandler(new
                        SelectionChangeEvent.Handler() {
                            @Override
                            public void onSelectionChange(
                                SelectionChangeEvent event) {
                                    updateSelections(
                                        selectionModel.
                                        getSelectedSet());
                            }
                    });
        results.setSelectionModel(selectionModel,
            DefaultSelectionEventManager
                    .<Reservation> createCheckboxManager());

        initTableColumns(selectionModel, sortHandler);

        vPanel.add(results);
        vPanel.add(pager);
    }

    public void updateSelections(Set<Reservation> set) {
        ArrayList<Reservation> list = new ArrayList<Reservation>();
        for (Reservation res : set) {
                list.add(res);
        }
        main.updateDataView(list);
    }

    public void setData(ArrayList<Reservation> aL) {
        list = dataProvider.getList();
        list.clear();
        for (Reservation rs : aL) {
                list.add(rs);
        }
    }

    public void initTableColumns(
                final MultiSelectionModel<Reservation> selectionModel,
                ListHandler<Reservation> sortHandler) {




        // Add Checkbox
        Column<Reservation, Boolean> checkColumn = new Column<
            Reservation, Boolean>(
```

```
                new CheckboxCell(true, false)) {
        @Override
        public Boolean getValue(Reservation object) {
                return selectionModel.isSelected(object);
        }
};
results.addColumn(checkColumn, "Selection");

// Add Text column Room
TextColumn<Reservation> roomColumn = new TextColumn<
    Reservation>() {
        @Override
        public String getValue(Reservation object) {
                return object.getRoom();
        }
};
results.addColumn(roomColumn, "Room");

// Comparator room
roomColumn.setSortable(true);
sortHandler.setComparator(roomColumn, new Comparator<
    Reservation>() {
        @Override
        public int compare(Reservation o1, Reservation o2) {
                return o1.getRoom().compareTo(o2.getRoom());
        }
});

// Add Text column Course
TextColumn<Reservation> courseColumn = new TextColumn<
    Reservation>() {
        @Override
        public String getValue(Reservation object) {
                return object.getCourse();
        }
};
results.addColumn(courseColumn, "Course");

// Comparator course
courseColumn.setSortable(true);
sortHandler.setComparator(courseColumn, new Comparator<
    Reservation>() {
        @Override
        public int compare(Reservation o1, Reservation o2) {
                return o1.getCourse().compareTo(o2.getCourse()
                    );
        }
});

// Add Int column CourseId
NumberFormat frmtCourseID = NumberFormat.getFormat("00000");
NumberCell courseIdCell = new NumberCell(frmtCourseID);
Column<Reservation, Number> courseIdColumn = new Column<
    Reservation, Number>(
                courseIdCell) {
        @Override
        public Number getValue(Reservation object) {
                return object.getCourseid();
        }
};
```

```
results.addColumn(courseIdColumn, "CourseID");

// Comparator CourseIdColumn
courseIdColumn.setSortable(true);
sortHandler.setComparator(courseIdColumn,
                new Comparator<Reservation>() {
                        @Override
                        public int compare(Reservation o1,
                            Reservation o2) {
                                return Integer.toString(o1.
                                    getCourseid()).compareTo(
                                                Integer.
                                                    toString(o2
                                                    .
                                                    getCourseid
                                                    ()));
                        }
                });

// Add Int column NrofHrs
NumberCell nrofHrsCell = new NumberCell();
Column<Reservation, Number> nrofHrsColumn = new Column<
    Reservation, Number>(
                nrofHrsCell) {
        @Override
        public Number getValue(Reservation object) {
                return object.getNrofHrs() / 2;
        }
};
results.addColumn(nrofHrsColumn, "NrofHrs");

// Add Text column day
TextColumn<Reservation> dayColumn = new TextColumn<Reservation
    >() {
        @Override
        public String getValue(Reservation object) {
                return object.getDayName();
        }
};
results.addColumn(dayColumn, "Day");

// Comparator Day
dayColumn.setSortable(true);
sortHandler.setComparator(dayColumn, new Comparator<
    Reservation>() {
        @Override
        public int compare(Reservation o1, Reservation o2) {
                return o1.getDayName().compareTo(o2.getDayName
                    ());
        }

});

// Add Int column TimeStart
NumberFormat frmtTimeStart = NumberFormat.getFormat("0000");
NumberCell timeStartCell = new NumberCell(frmtTimeStart);
Column<Reservation, Number> timeStartColumn = new Column<
    Reservation, Number>(
                timeStartCell) {
        @Override
```

```
        public Number getValue(Reservation object) {
                return object.getTimeStart();
        }
};
results.addColumn(timeStartColumn, "Start Time");

// Comparator timeStartcolumn
timeStartColumn.setSortable(true);
sortHandler.setComparator(timeStartColumn,
                new Comparator<Reservation>() {
                        @Override
                        public int compare(Reservation o1,
                            Reservation o2) {
                                return Integer.toString(o1.
                                    getTimeStart()).compareTo(
                                                Integer.
                                                    toString(o2
                                                    .
                                                    getTimeStart
                                                    ()));
                        }
                });

// Add Text column Schedule Group
TextColumn<Reservation> scheduleGroupColumn = new TextColumn<
    Reservation>() {
        @Override
        public String getValue(Reservation object) {
                return object.getScheduleGroup();
        }
};
results.addColumn(scheduleGroupColumn, "Schedule Group");

// Comparator ScheduleGroup
scheduleGroupColumn.setSortable(true);
sortHandler.setComparator(scheduleGroupColumn,
                new Comparator<Reservation>() {
                        @Override
                        public int compare(Reservation o1,
                            Reservation o2) {
                                return o1.getScheduleGroup().
                                    compareTo(
                                                o2.
                                                    getScheduleGroup
                                                    ());
                        }
                });

// Add Text column Semester Id
TextColumn<Reservation> semesterIdColumn = new TextColumn<
    Reservation>() {
        @Override
        public String getValue(Reservation object) {
                return object.getSemesterId();
        }
};
results.addColumn(semesterIdColumn, "Semester Id");

// Add Text column Semester type
```

```
                TextColumn<Reservation> semesterTypeColumn = new TextColumn<
                    Reservation>() {
                        @Override
                        public String getValue(Reservation object) {
                                return object.getSemesterType();
                        }
                };
                results.addColumn(semesterTypeColumn, "Semester Type");

                // Add Int column Teacher
                NumberFormat frmtTeacher = NumberFormat.getFormat("0000");
                NumberCell teacherCell = new NumberCell(frmtTeacher);
                Column<Reservation, Number> teacherColumn = new Column<
                    Reservation, Number>(
                                teacherCell) {
                        @Override
                        public Number getValue(Reservation object) {
                                return object.getCourseTeacher();
                        }
                };
                results.addColumn(teacherColumn, "Teacher");

                results.setWidth("100%", true);
                results.setColumnWidth(checkColumn, 7.0, Unit.PCT);
                results.setColumnWidth(roomColumn, 10.0, Unit.PCT);
                results.setColumnWidth(courseColumn, 15.0, Unit.PCT);
                results.setColumnWidth(courseIdColumn, 7.0, Unit.PCT);
                results.setColumnWidth(nrofHrsColumn, 7.0, Unit.PCT);
                results.setColumnWidth(dayColumn, 10.0, Unit.PCT);
                results.setColumnWidth(timeStartColumn, 5.0, Unit.PCT);
                results.setColumnWidth(scheduleGroupColumn, 10.0, Unit.PCT);
                results.setColumnWidth(semesterIdColumn, 7.0, Unit.PCT);
                results.setColumnWidth(semesterTypeColumn, 7.0, Unit.PCT);
                results.setColumnWidth(teacherColumn, 7.0, Unit.PCT);
        }

        public void clearList() {
                list.clear();
                selectionModel.getSelectedSet().clear();

        }

        public static final ProvidesKey<Reservation> KEY_PROVIDER = new
            ProvidesKey<Reservation>() {
                @Override
                public Object getKey(Reservation item) {
                        return item == null ? null : item;
                }
        };
}

package bachelor.client.gui;

import java.util.ArrayList;

import org.apache.poi.ss.usermodel.HorizontalAlignment;

import bachelor.client.model.Reservation;

import com.google.gwt.event.dom.client.ClickEvent;
```

```
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.Grid;
import com.google.gwt.user.client.ui.HTML;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.TabPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

public class DataView extends Composite {

        private TabPanel tabPanel = new TabPanel();
        private MainView main;
        private Grid dataGrid;
        private VerticalPanel vPanel = new VerticalPanel();
        private FlexTable datesTable = new FlexTable();

        public DataView(MainView main) {
                initWidget(tabPanel);
                this.main = main;
                tabPanel.setSize("80%", "70%");
                this.vPanel.setSize("100%", "100%");
                this.vPanel.setHorizontalAlignment(HasHorizontalAlignment.
                    ALIGN_CENTER);

                Button clearButton = new Button("Clear Week Schedule");
                clearButton.addClickHandler(new clearClickHandler());

                dataGrid = new Grid(10, 6);
                dataGrid.setSize("100%", "100%");
                dataGrid.setBorderWidth(1);

                dataGrid.setText(0, 1, "Monday");
                dataGrid.setText(0, 2, "Tuesday");
                dataGrid.setText(0, 3, "Wednesday");
                dataGrid.setText(0, 4, "Thursday");
                dataGrid.setText(0, 5, "Friday");

                dataGrid.setText(1, 0, "08:00");
                dataGrid.setText(2, 0, "09:00");
                dataGrid.setText(3, 0, "10:00");
                dataGrid.setText(4, 0, "11:00");
                dataGrid.setText(5, 0, "12:00");
                dataGrid.setText(6, 0, "13:00");
                dataGrid.setText(7, 0, "14:00");
                dataGrid.setText(8, 0, "15:00");
                dataGrid.setText(9, 0, "16:00");

                this.vPanel.add(dataGrid);
                this.vPanel.add(clearButton);

                // Week tab
                tabPanel.add(vPanel, "Week Schedule");

                datesTable.setBorderWidth(1);

                // Date tab
                tabPanel.add(datesTable, "Dates");
```

```
        // Tab start
        tabPanel.selectTab(0);
    }

    public void updateSchedule(ArrayList<Reservation> selections) {
        for (Reservation res : selections) {
            String courseIdBuilding = Integer.toString(res.
                getCourseid()) + "/"
                        + res.getRoom();

            if (res.getTimeStart() == 800) {
                addNrofHrs(courseIdBuilding, res.getNrofHrs(),
                    1,
                            dayNumber(res.getDay()));
            }
            if (res.getTimeStart() == 900) {
                addNrofHrs(courseIdBuilding, res.getNrofHrs(),
                    2,
                            dayNumber(res.getDay()));
            }
            if (res.getTimeStart() == 1000) {
                addNrofHrs(courseIdBuilding, res.getNrofHrs(),
                    3,
                            dayNumber(res.getDay()));
            }
            if (res.getTimeStart() == 1100) {
                addNrofHrs(courseIdBuilding, res.getNrofHrs(),
                    4,
                            dayNumber(res.getDay()));
            }
            if (res.getTimeStart() == 1200) {
                addNrofHrs(courseIdBuilding, res.getNrofHrs(),
                    5,
                            dayNumber(res.getDay()));
            }
            if (res.getTimeStart() == 1300) {
                addNrofHrs(courseIdBuilding, res.getNrofHrs(),
                    6,
                            dayNumber(res.getDay()));
            }
            if (res.getTimeStart() == 1400) {
                addNrofHrs(courseIdBuilding, res.getNrofHrs(),
                    7,
                            dayNumber(res.getDay()));
            }
            if (res.getTimeStart() == 1500) {
                addNrofHrs(courseIdBuilding, res.getNrofHrs(),
                    8,
                            dayNumber(res.getDay()));
            }
            if (res.getTimeStart() == 1600) {
                addNrofHrs(courseIdBuilding, res.getNrofHrs(),
                    9,
                            dayNumber(res.getDay()));
            }
        }
    }

    public void addNrofHrs(String text, int nrofHrs, int row, int column)
        {
```

```
            if (nrofHrs == 2) {
                    addText(text, row, column);
            }
            if (nrofHrs == 4) {
                    addText(text, row, column);
                    addText(text, row + 1, column);
            }
            if (nrofHrs == 6) {
                    addText(text, row, column);
                    addText(text, row + 1, column);
                    addText(text, row + 2, column);
            }
            if (nrofHrs == 8) {
                    addText(text, row, column);
                    addText(text, row + 1, column);
                    addText(text, row + 2, column);
                    addText(text, row + 3, column);
            }
    }

    public void addText(String text, int row, int column) {

            if (dataGrid.getText(row, column).length() != 1) {
                    String completeText = dataGrid.getText(row, column) +
                        "␣" + text;

                    dataGrid.setText(row, column, strDuplicate(
                        completeText));

            }
            if (dataGrid.getText(row, column).length() == 1) {
                    dataGrid.setText(row, column, text);
            }
    }

    public int dayNumber(int day) {
            int dayNumber = 10;
            if (day == 1000000) {
                    dayNumber = 1;
            }
            if (day == 100000) {
                    dayNumber = 2;
            }
            if (day == 10000) {
                    dayNumber = 3;
            }
            if (day == 1000) {
                    dayNumber = 4;
            }
            if (day == 100) {
                    dayNumber = 5;
            }
            return dayNumber;
    }

    public void clearSchedule() {
            for (int j = 1; j <= dataGrid.getRowCount(); j++) {
                    for (int i = 1; i <= dataGrid.getColumnCount() + 3; i
                        ++) {
```

```
                                dataGrid.clearCell(i, j);
                        }
                }
                main.clearScheduleList();
        }

        public String strDuplicate(String str) {
                String retstr[];
                String correctString = "";
                ArrayList<String> tempList = new ArrayList<String>();
                retstr = str.split(" ");

                for (String strTemp : retstr) {
                        tempList.add(strTemp);
                }

                int size = tempList.size();
                for (int i = 0; i < size; i++) {
                        for (int j = i + 1; j < size; j++) {
                                if (tempList.get(j).equals(tempList.get(i))) {
                                        tempList.remove(j);
                                        j--;
                                        size--;
                                }
                        }
                }

                for (String strPart : tempList) {
                        correctString = correctString + " " + strPart;
                }

                return correctString;
        }

        public String binaryToDates(int resWeeks) {
                String binaryNumber = Integer.toBinaryString(resWeeks);

                return null;
        }

        private class clearClickHandler implements ClickHandler {
                @Override
                public void onClick(ClickEvent event) {
                        clearSchedule();
                }
        }
}


package bachelor.client.service;

import java.util.ArrayList;

import bachelor.client.model.Reservation;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("RRService")
public interface RRService extends RemoteService {
```

```
        ArrayList<Reservation> searchReservation(String search);

        boolean importcheck(boolean check);



}


package bachelor.client.service;


public interface RRServiceClientInt {

        void searchReservation(String search);
        void importcheck(boolean check);

}

package bachelor.client.service;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface RRServiceAsync {

        void searchReservation(String search, AsyncCallback callback);
        void importcheck(boolean check , AsyncCallback callback);

}
```

Serversiden

```
package bachelor.server;

public class RRServiceImpl extends RemoteServiceServlet implements RRService {
        private static Connection conn = null;
        private static String url = "jdbc:mysql://localhost:3306/roomrequests"
            ;
        private static String user = "root";
        private static String pass = "";

        public static String[] databarBuildings = new String[] { "B303-IT043",
                        "B303-IT046", "B303-IT047", "B303-IT048", "B303B-IT034
                            ",
                        "B305-IT005", "B305-IT006", "B306-IT030", "B306-IT039"
                            ,
                        "B308-IT001", "B308-IT009", "B308-IT017", "B308-IT101"
                            ,
                        "B308-IT109", "B308-IT117", "B308-IT127", "B325-IT017"
                            ,
                        "B325-IT025", "B329A-IT108", "B341-IT003", "B341-IT015
                            ",
                        "B341-IT019" };

        @Override
        public ArrayList<Reservation> searchReservation(String search) {
                try {
                        getConnection();
```

```
} catch (Exception e) {
        e.printStackTrace();
}

ArrayList<Reservation> searchResult = new ArrayList<
    Reservation>();

try {
        PreparedStatement ps;

        if (freetimeslot(search)) {

                String[] str = search.split("␣", 3);
                ps = (PreparedStatement) conn
                                .prepareStatement("SELECT␣Room
                                    ,Course,CourseId,NrofHrs,
                                    Day,TimeStart"
                                                + ",SemesterId
                                                    ,
                                                    SemesterType
                                                    ,
                                                    ReservationWeeks
                                                    ,
                                                    CourseTeachers
                                                    ␣FROM␣
                                                    requests␣
                                                    WHERE␣Room␣
                                                    LIKE␣?␣AND␣
                                                    TimeStart␣
                                                    LIKE␣'%"
                                                + Integer.
                                                    parseInt(
                                                    str[1])
                                                + "%'␣AND␣Day␣
                                                    LIKE␣'%"
                                                + dayNumber(
                                                    str[2])
                                                + "%'");
                ps.setString(1, "%" + str[0] + "%");

        } else if (timeday(search)) {

                String[] str = search.split("␣", 2);
                ps = (PreparedStatement) conn
                                .prepareStatement("SELECT␣Room
                                    ,Course,CourseId,NrofHrs,
                                    Day,TimeStart"
                                                + ",SemesterId
                                                    ,
                                                    SemesterType
                                                    ,
                                                    ReservationWeeks
                                                    ,
                                                    CourseTeachers
                                                    ␣FROM␣
                                                    requests␣
                                                    WHERE␣
                                                    TimeStart␣
                                                    LIKE␣'%"
```

```
                                                + Integer.
                                                    parseInt(
                                                    str[0])
                                                + "%' AND Day
                                                    LIKE '%"
                                                + dayNumber(
                                                    str[1])
                                                + "%'");

                        } else if (buildingtime(search)) {

                                String[] str = search.split(" ", 2);
                                ps = (PreparedStatement) conn
                                                .prepareStatement("SELECT Room
                                                    , Course, CourseId, NrofHrs,
                                                    Day, TimeStart"
                                                + ",SemesterId
                                                    ,
                                                    SemesterType
                                                    ,
                                                    ReservationWeeks
                                                    ,
                                                    CourseTeachers
                                                    FROM
                                                    requests
                                                    WHERE Room
                                                    LIKE ? AND
                                                    TimeStart
                                                    LIKE '%"
                                                + Integer.
                                                    parseInt(
                                                    str[1]) + "
                                                    %'");
                                ps.setString(1, "%" + str[0] + "%");

                        } else if (buildingday(search)) {

                                String[] str = search.split(" ", 2);
                                ps = (PreparedStatement) conn
                                                .prepareStatement("SELECT Room
                                                    , Course, CourseId, NrofHrs,
                                                    Day, TimeStart"
                                                + ",SemesterId
                                                    ,
                                                    SemesterType
                                                    ,
                                                    ReservationWeeks
                                                    ,
                                                    CourseTeachers
                                                    FROM
                                                    requests
                                                    WHERE Room
                                                    LIKE ? AND
                                                    Day LIKE '%
                                                    "
                                                + dayNumber(
                                                    str[1]) + "
                                                    %'");
                                ps.setString(1, "%" + str[0] + "%");
```

```
                } else {
                        ps = (PreparedStatement) conn
                                        .prepareStatement("SELECT Room
                                            , Course , CourseId , NrofHrs ,
                                            Day , TimeStart"
                                                        + " , SemesterId
                                                            ,
                                                            SemesterType
                                                            ,
                                                            ReservationWeeks
                                                            ,
                                                            CourseTeachers
                                                            FROM
                                                            requests
                                                            WHERE Room
                                                            LIKE ? OR
                                                            Course LIKE
                                                            ? OR
                                                            CourseId
                                                            LIKE ?");
                        ps.setString(1, "%" + search + "%");
                        ps.setString(2, "%" + search + "%");
                        ps.setString(3, "%" + search + "%");
                }

                ArrayList<Reservation> resultTemp = new ArrayList<
                    Reservation>();

                ResultSet rs = ps.executeQuery();
                while (rs.next()) {
                        Reservation res = new Reservation();
                        res.setRoom(rs.getString("Room"));
                        res.setCourse(rs.getString("Course"));
                        res.setCourseid(rs.getInt("CourseId"));
                        res.setNrofHrs(rs.getInt("NrofHrs"));
                        res.setDay(rs.getInt("Day"));
                        res.setTimeStart(rs.getInt("TimeStart"));
                        res.setSemesterId(rs.getString("SemesterId"));
                        res.setSemesterType(rs.getString("SemesterType
                            "));
                        res.setReservationWeeks(rs.getInt("
                            ReservationWeeks"));
                        res.setCourseTeacher(rs.getInt("CourseTeachers
                            "));
                        res.setDayName(rs.getInt("Day"));
                        res.setScheduleGroup(rs.getInt("Day"), rs.
                            getInt("TimeStart"));

                        resultTemp.add(res);

                }
                searchResult = removeDuplicates(resultTemp);

                ps.close();
                conn.close();
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return searchResult;
}
```

```
@Override
public boolean importcheck(boolean check) {

        if (check == true) {
                deleteTable();
                File importfile1 = new File(
                                "C:\\Users\\Jakob\\Dropbox\\Bachelor -
                                    Room Reservation System\\Kode\\
                                    RoomReservation\\war"
                                            + "\\imports\\
                                                ExportUndervisningsActivities1
                                                .xls");
                File importfile2 = new File(
                                "C:\\Users\\Jakob\\Dropbox\\Bachelor -
                                    Room Reservation System\\Kode"
                                            + "\\RoomReservation\\
                                                war\\imports\\
                                                ExportUndervisningsActivities1
                                                .xls");
                importfile1.setReadable(true);
                importfile2.setReadable(true);
                try {
                        Import(importfile1);
                        Import(importfile2);
                } catch (IOException e) {
                        e.printStackTrace();
                }
                check = false;
        }
        return check;
}

private void Import(File importfile) throws IOException {

        FileInputStream file = new FileInputStream(importfile);

        HSSFWorkbook wb = new HSSFWorkbook(file);
        HSSFSheet sheet = wb.getSheetAt(0);

        for (int i = 0; i < sheet.getLastRowNum(); i++) {
                HSSFRow row = sheet.getRow(i);
                HSSFCell cell = row.getCell(1);
                if (Arrays.asList(databarBuildings).contains(
                                cell.getStringCellValue())) {

                        if (row.getCell(14).getCellType() == Cell.
                            CELL_TYPE_STRING) {
                                insertMySQL(
                                                row.getCell(1).
                                                    getStringCellValue
                                                    (),
                                                strSplitsCourse(row.
                                                    getCell(2).
                                                    getStringCellValue
                                                    ()),
                                                (int) row.getCell(3).
                                                    getNumericCellValue
                                                    (),
```

```
                (int) row.getCell(4).
                    getNumericCellValue
                    (),
                (int) row.getCell(5).
                    getNumericCellValue
                    (),
                datetoInt(row.getCell
                    (6).
                    getDateCellValue())
                    , row
                                    .
                                        getCell
                                        (7)
                                        .
                                        getStringCellV
                                        (),

                                        row
                                    .
                                        getCell
                                        (8)
                                        .
                                        getStringCellV
                                        (),
                                         (
                                        int
                                        )
                                        row
                                    .
                                        getCell
                                        (9)
                                        .
                                        getNumericCell
                                        (),

                                        Integer

                                    .
                                        valueOf
                                        (
                                        strSplits
                                        (
                                        row
                                        .
                                        getCell
                                        (14)

                                                    .

            } else if (row.getCell(14).getCellType() ==
                Cell.CELL_TYPE_NUMERIC) {
                    insertMySQL(
```

```
                                                        row.getCell(1).
                                                            getStringCellValue
                                                            (),
                                                        strSplitsCourse(row.
                                                            getCell(2).
                                                            getStringCellValue
                                                            ()),
                                                        (int) row.getCell(3).
                                                            getNumericCellValue
                                                            (),
                                                        (int) row.getCell(4).
                                                            getNumericCellValue
                                                            (),
                                                        (int) row.getCell(5).
                                                            getNumericCellValue
                                                            (),
                                                        datetoInt(row.getCell
                                                            (6).
                                                            getDateCellValue())
                                                            , row
                                                                    .
                                                                        getCell
                                                                        (7)
                                                                        .
                                                                        getStringCellV
                                                                        (),

                                                                        row
                                                                    .
                                                                        getCell
                                                                        (8)
                                                                        .
                                                                        getStringCellV
                                                                        (),
                                                                         (
                                                                        int
                                                                        )
                                                                        row
                                                                    .
                                                                        getCell
                                                                        (9)
                                                                        .
                                                                        getNumericCell
                                                                        (),
                                                        (int) row.getCell(14).
                                                            getNumericCellValue
                                                            ());
                                }
                        }
                        wb.close();
                }
        }

        public static ArrayList<Reservation> removeDuplicates(
                        ArrayList<Reservation> res) {

                int size = res.size();

                for (int i = 0; i < size - 1; i++) {
                        for (int j = i + 1; j < size; j++) {
```

```
                        if (res.get(i).getCourse().equals(res.get(j).
                            getCourse())
                                && res.get(i).getRoom().equals
                                    (res.get(j).getRoom())
                                && res.get(i).getDayName()
                                        .equals(res.
                                            get(j).
                                            getDayName
                                            ())
                                && Integer.toString(res.get(i)
                                    .getTimeStart()).equals(
                                        Integer.
                                            toString(
                                            res.get(j).
                                            getTimeStart
                                            ()))
                                && res.get(i).getSemesterId()
                                        .equals(res.
                                            get(j).
                                            getSemesterId
                                            ())
                                && res.get(i).getSemesterType
                                    ()
                                        .equals(res.
                                            get(j).
                                            getSemesterType
                                            ())) {
                            res.remove(j);
                            j--;
                            size--;
                        }
                    }
                }
                return res;
        }

        private static boolean freetimeslot(String search) {
                String str[];

                str = search.split("␣", 3);
                if (str.length >= 3 && isBuilding(str[0]) && isTimeStart(str
                    [1])
                                && daycheck(str[2])) {
                        return true;
                }
                return false;
        }

        private static boolean timeday(String search) {
                String str[];

                str = search.split("␣", 2);
                if (str.length >= 2 && isTimeStart(str[0]) && daycheck(str[1])
                    ) {
                        return true;
                }
                return false;
        }

        private static boolean buildingtime(String search) {
```

```
            String str[];

            str = search.split("␣", 2);
            if (str.length >= 2 && isBuilding(str[0]) && isTimeStart(str
                [1])) {
                    return true;
            }
            return false;
    }

    private static boolean buildingday(String search) {
            String str[];

            str = search.split("␣", 2);
            if (str.length >= 2 && isBuilding(str[0]) && daycheck(str[1]))
                 {
                    return true;
            }
            return false;
    }

    private static boolean isTimeStart(String string) {
            if (string.equals("800") || string.equals("900")
                            || string.equals("1000") || string.equals("
                               1100")
                            || string.equals("1200") || string.equals("
                               1300")
                            || string.equals("1400") || string.equals("
                               1500")
                            || string.equals("1600") || string.equals("
                               1700")) {
                    return true;
            }
            return false;
    }

    private static boolean isBuilding(String string) {
            String subString;
            for (String str : databarBuildings) {
                    for (int i = 0; i <= str.length(); i++) {
                            for (int j = i + 1; j <= str.length(); j++) {
                                    subString = str.substring(i, j);
                                    if (string.equals(subString)) {
                                            return true;
                                    }
                            }
                    }
            }
            return false;
    }

    private static boolean daycheck(String string) {
            if (string.equals("Monday") || string.equals("Tuesday")
                            || string.equals("Wednesday") || string.equals
                               ("Thursday")
                            || string.equals("Friday")) {
                    return true;
            }
            return false;
    }
```

```
public int dayNumber(String day) {
        int dayNumber = 0;
        if (day.equals("Monday")) {
                dayNumber = 1000000;
        }
        if (day.equals("Tuesday")) {
                dayNumber = 100000;
        }
        if (day.equals("Wednesday")) {
                dayNumber = 10000;
        }
        if (day.equals("Thursday")) {
                dayNumber = 1000;
        }
        if (day.equals("Friday")) {
                dayNumber = 100;
        }
        return dayNumber;
}

public String strSplits(String str) {
        String retstr[];
        retstr = str.split(",");
        return retstr[0];
}

public static String strSplitsCourse(String str) {
        String retstr[];
        String correctStr;

        retstr = str.split(" ", 2);
        String controlStr = retstr[1].substring(retstr[1].length() -
           2,
                        retstr[1].length() - 1);

        if (isInteger(controlStr) == true) {
                correctStr = retstr[1].substring(0, retstr[1].length()
                    - 3);
        } else {
                return retstr[1];
        }
        return correctStr;
}

public static int datetoInt(Date time) {
        int correctTime;
        Calendar cal = Calendar.getInstance();
        cal.setTime(time);
        correctTime = cal.get(Calendar.HOUR_OF_DAY) * 100
                        + cal.get(Calendar.MINUTE);
        return correctTime;
}

public static boolean isInteger(String str) {
        try {
                Integer.parseInt(str);
                return true;
        } catch (NumberFormatException nfe) {
                return false;
```

```
        }
    }

    public static Connection getConnection() throws Exception {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(url, user, pass);
            return conn;
        } catch (Exception e) {
        }
        return null;
    }

    public void deleteTable() {
        try {
            getConnection();
        } catch (Exception e) {
            e.printStackTrace();
        }

        try {
            PreparedStatement ps = (PreparedStatement) conn
                    .prepareStatement("TRUNCATE TABLE
                        requests");
            ps.executeUpdate();
            ps.close();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void insertMySQL(String room, String course, Integer courseid,
            Integer nrofHrs, Integer day, Integer timeStart,
                String semesterId,
            String semesterType, Integer reservationWeeks,
            Integer courseTeachers) {
        try {
            getConnection();
        } catch (Exception e1) {
            e1.printStackTrace();
        }

        try {
            PreparedStatement ps = (PreparedStatement) conn
                    .prepareStatement("INSERT INTO
                        requests(Room, Course, CourseId,
                        NrofHrs, Day, TimeStart"
                                + ", SemesterId,
                                    SemesterType,
                                    ReservationWeeks,
                                    CourseTeachers)
                                    VALUES
                                    (?,?,?,?,?,?,?,?,?,?)
                                    ");
            ps.setString(1, room);
            ps.setString(2, course);
            ps.setInt(3, courseid);
            ps.setInt(4, nrofHrs);
            ps.setInt(5, day);
```

```
                        ps.setInt(6, timeStart);
                        ps.setString(7, semesterId);
                        ps.setString(8, semesterType);
                        ps.setInt(9, reservationWeeks);
                        ps.setInt(10, courseTeachers);
                        ps.executeUpdate();
                        ps.close();
                        conn.close();

                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
}
```

Model

```
package bachelor.client.model;

import java.io.Serializable;

public class Reservation implements Serializable {
        private String room;
        private String course;
        private int courseid;
        private int nrofHrs;
        private int day;
        private int timeStart;
        private String semesterId;
        private String semesterType;
        private int reservationWeeks;
        private int courseTeacher;
        private String scheduleGroup;
        private String dayName;

        public Reservation(String room, String course, int courseid, int
            nrofHrs,
                        int day, int timeStart, String semesterId, String
                            semesterType,
                        int reservationWeeks, int courseTeacher) {
                this.room = room;
                this.course = course;
                this.courseid = courseid;
                this.nrofHrs = nrofHrs;
                this.day = day;
                this.timeStart = timeStart;
                this.semesterId = semesterId;
                this.semesterType = semesterType;
                this.reservationWeeks = reservationWeeks;
        }

        public Reservation() {
        }

        public String getScheduleGroup() {
                return scheduleGroup;
        }

        public void setScheduleGroup(int day, int timeStart) {
                String scheduleGroup = null;
```

```
            if (day == 1000000 && timeStart <= 1200 && timeStart >= 800) {
                    scheduleGroup = "E1A/F1A";
            }
            if (day == 1000000 && timeStart <= 1700 && timeStart >= 1200)
               {
                    scheduleGroup = "E2A/F2A";
            }
            if (day == 100000 && timeStart <= 1200 && timeStart >= 800) {
                    scheduleGroup = "E3A/F3A";
            }
            if (day == 100000 && timeStart <= 1700 && timeStart >= 1200) {
                    scheduleGroup = "E4A/F4A";
            }
            if (day == 10000 && timeStart <= 1200 && timeStart >= 800) {
                    scheduleGroup = "E5A/F5A";
            }
            if (day == 10000 && timeStart <= 1700 && timeStart >= 1200) {
                    scheduleGroup = "E5B/F5B";
            }
            if (day == 1000 && timeStart <= 1200 && timeStart >= 800) {
                    scheduleGroup = "E2B/F2B";
            }
            if (day == 1000 && timeStart <= 1700 && timeStart >= 1200) {
                    scheduleGroup = "E1B/F1B";
            }
            if (day == 100 && timeStart <= 1200 && timeStart >= 800) {
                    scheduleGroup = "E4B/F4B";
            }
            if (day == 100 && timeStart <= 1700 && timeStart >= 1200) {
                    scheduleGroup = "E3B/F3B";
            }

            this.scheduleGroup = scheduleGroup;
    }

    public String getDayName() {
            return dayName;
    }

    public void setDayName(int day) {
            String dayName = null;
            if (day == 1000000) {
                    dayName = "Monday";
            }
            if (day == 100000) {
                    dayName = "Tuesday";
            }
            if (day == 10000) {
                    dayName = "Wednesday";
            }
            if (day == 1000) {
                    dayName = "Thursday";
            }
            if (day == 100) {
                    dayName = "Friday";
            }

            this.dayName = dayName;
    }
```

```
public String getRoom() {
        return room;
}

public void setRoom(String room) {
        this.room = room;
}

public String getCourse() {
        return course;
}

public void setCourse(String course) {
        this.course = course;
}

public int getCourseid() {
        return courseid;
}

public void setCourseid(int courseid) {
        this.courseid = courseid;
}

public int getNrofHrs() {
        return nrofHrs;
}

public void setNrofHrs(int nrofHrs) {
        this.nrofHrs = nrofHrs/2;
}

public int getDay() {
        return day;
}

public void setDay(int day) {
        this.day = day;
}

public int getTimeStart() {
        return timeStart;
}

public void setTimeStart(int timeStart) {
        this.timeStart = timeStart;
}

public String getSemesterId() {
        return semesterId;
}

public void setSemesterId(String semesterId) {
        this.semesterId = semesterId;
}

public String getSemesterType() {
        return semesterType;
}
```

```
        public void setSemesterType(String semesterType) {
                this.semesterType = semesterType;
        }

        public int getReservationWeeks() {
                return reservationWeeks;
        }

        public void setReservationWeeks(int reservationWeeks) {
                this.reservationWeeks = reservationWeeks;
        }

        public int getCourseTeacher() {
                return courseTeacher;
        }

        public void setCourseTeacher(int courseTeachers) {
                this.courseTeacher = courseTeachers;
        }

}
```