

SUBMISSION OF WRITTEN WORK

Class code: GRPRO

Name of course: GRUNDLÆGGENDE PROGRAMMERING

Course manager: DAN HANSEN

Course e-portfolio: GRPRO-AUTUMN 2014

Thesis or project title: KAFFEKLUBBEN

Supervisor: JESPER MADSEN

Full Name: Birthdate (dd/mm/yyyy): E-mail:

1. MARK ROSTGAARD MORTENSEN 09/04-1993 MROM@itu.dk

2. AMANDA ENGEL THILO 21/03-1993 AENT@itu.dk

3. FREDERIK HOVMAND JØRGENSEN 20/01-1993 FHOV@itu.dk

4. _____ @itu.dk

5. _____ @itu.dk

6. _____ @itu.dk

7. _____ @itu.dk



EXAM PROJECT
Biobooking

KAFFEKLUBBEN

Amanda Engel Thilo

Frederik Hovmand Jørgensen

Mark Rostgaard Mortensen

DECEMBER 17, 2014

Abstract

In this report we address our workgroups attempt at designing a digital solution for broadening the audience for both the digital and physical offerings, that Statens Museum for Kunst(SMK) provides. We use a multitude of idea generation techniques, to arrive at a product, and then interviews with SMK. We propose an enhanced web-presence, for image viewing and discussion, based on feedback from SMK and visits to the museum. The report concludes with the proposal of a system in which users can comment and interact with the different artworks on the website.

Indhold

1	Forord	2
2	Indledning	3
2.1	<i>Baggrund</i>	3
2.2	<i>Problemstilling</i>	3
2.2.1	Krav til programmet	3
2.2.2	Brugerscenario	4
2.2.3	Systemdesign	4
3	Problemanalyse	6
3.1	<i>Vores løsning</i>	6
3.1.1	Alternative løsninger	6
3.2	<i>Databasedesign</i>	6
4	Brugervejledning	8
4.1	<i>Programmet</i>	8
4.1.1	Begrænsninger	10
4.1.2	Fejlmeddelelser	10
4.2	<i>Eksempel</i>	10
5	Teknisk Analyse	11
5.1	<i>Model View Controller</i>	11
5.1.1	Moduler	11
5.1.2	Datastrukturer	11
5.1.3	Algoritmer	11
5.2	<i>Brugergrænsedesign</i>	11
5.2.1	Begrænsninger	11
6	Afprøvning	12
6.1	<i>Brugerafprøvning</i>	12
6.2	<i>Unitet</i>	12
6.3	<i>Resultat</i>	12
7	Konklusion	13
8	Litteratur	14
9	Bilag	15

Kapitel 1

Forord

Kapitel 2

Indledning

2.1 Baggrund

Det problemområde vi har arbejdet med i forbindelse med vores projekt, er udviklingen af et softwaresystem til brug for en ekspedient der betjener en billetluge i en biograf. Det område der skulle dækkes var reservation både når kunden stod foran billetlugen, samt når kunden ringede ind på telefon. Vores løsning skulle håndtere ekspedientens arbejdsopgaver i sådanne situationer. Løsningen skal derfor fokusere udelukkende på reservationer, og ikke salg.

2.2 Problemstilling

Den overordnede problemstilling som vores program besvarer, er hvordan man udviklinger et simpelt, brugervenligt og databasebaseret softwaresystem til brug for en ekspeditent i en billetluge.

2.2.1 Krav til programmet

Af krav til programmet kan nævnes at der i forbindelse med reservationerne skal oplyses navn eller anden identitet på kunden. Vi har i vores program valgt at den centrale information omkring brugeren er dennes telefonnummer. Det har vi gjort af den grund af flere kunder godt kan have samme navn, men at ens telefonnummer er unikt.

Derudover skal biografen indeholde flere sale, og sende flere forskellige forestillinger i løbet af dagen. Salenes størrelse, antal pladser mm. skal fremgå af databasen frem for at være indkodet i selve programteksten. Derfor har vi i stedet for at skrive data ind i programteksten, valgt at lave en speciel klasse til at hente data fra databasen.

Det samme er gældende for de enkelte forestillinger, der derfor også bliver gemt i databasen.

2.2.2 Brugerscenario

- skal laves om Når ekspedienten åbner vores biobooking-system vil der på venstre side af vinduet være opelistet de film der spilles i biografen i øjeblikket. Klikker ekspedienten på en af disse film vil der på højre side af vinduet vise sig de tilhørende spilletider til den valgte film. Klikker ekspedienten derimod ikke på noget, vil der stå følgene tekst *Klik på en film til venstre for at vise tidspunkter.* Øverst oppe i vinduet vil man se tre faner: *Forestillinger, Reservation og Ret reservation.* Man kan godt navigere mellem fanerne, men har man ikke først valgt film og tidspunkt vil *Reservation* være ubrugelige.

Efter at ekspedienten har valgt en film og en spilletid vil systemet automatisk skifte videre til den næste fane: *Reservation.* Her kan ekspedienten vælge frit mellem de ledige sæder. Dette gøres ved at ekspedienten enten klikker enkeltvis på de grønne sæder, eller ved at trække musen over de ønskede sæder. De valgte sæder vil nu blive blå, så ekspedienten kan se hvilke der er valgt. Er sæderne tilfredsstillende skal ekspedienten skrive navn samt telefonnummer på kunden ind i et informationsfelt nedernst til højre på siden. Nedderst på siden ses i venstre hjørne antallet af sæder i alt, samt det ledige antal sæder. Dette kan ekspedienten også se på det grafiske billede over biografsalen, da de allerede reserverede sæder er røde, samt umulige at vælge for reservation. Efter kundens information er indtastet og ekspedienten har klikket på *Fuldfør reservation* kommer et pop-up vindue op, hvor der står skrevet: *Bestillingen er gennemført.*

Ønsker den pågældende kunde derimod at ændre eller slette en reservation, skal ekspedienten klikke sig ind på den sidste fane: *Ret reservation.* På denne side finder ekspedienten øverst oppe et input felt hvor kundens telefonnummer skal indtastes. Når ekspedienten har klikket enter vil kundens reservation(er) dukke op nedenfor i en pæn liste. Ekspedienten kan så klikke på den reservation som kunden ønsker ændret. Nu føres ekspedienten til en popup *Reservation* side, hvor de sæder tilknyttet den pågældende reservation nu ses med blåt. Ekspedienten kan således tilføje eller fjerne sæder ved at klikke på sæder med musen og derefter klikke på *Ret reservation.* Ønsker kunden helt at slette reservationen kan dette lade sig gøre ved at klikke på *Slet reservation.*

2.2.3 Systemdesign

Vores system er overordnet baseret på data fra databasen. Systemet genererer layout ud fra forskellige data i databasen. Dette indebærer bl.a. hvilke film der er i film-tabellen, hvilke forestillinger der er tilknyttet til en given film, hvilke reservationer der er tilknyttet en given forestilling, hvilke reservationer der er tilknyttet et givent telefonnummer mv. Programmet er delt op, så mest mulig databehandling foretages hos klienten/brugeren. Dette indebærer objekter med informationer om de forskellige forestillinger, arrays med valgte sæder mv. Databaseforespørgsler er

lavet så omfattende som muligt, så der udføres så få forespørgsler som muligt og data gemmes i objekter og arrays eller bliver benyttet med det samme i den grafiske fremstilling af programmet som eksempelvis ved generering af film-knapper.

Kapitel 3

Problemanalyse

3.1 Vores løsning

Den overordnede problemstilling som vi har valgt at fokusere på at løse er hvordan man udvikler et simpelt og implecit program som er brugervenligt og med så lidt data som muligt i selve programkoden.

Vores program løser problemstilligen ved at have alt information omkring forestillinger, sale, film og reservationer gemt i en database. Dette gør selve programteksten simpel og nem at ændre i, og vi har på den måde undgået for meget kodeduplikering. For at opfylde ønsket om brugervenlighed har vi valgt at vise størstedelen af programmet i ét vindue, hvor brugeren så nemt kan navigere mellem de forskellige muligheder programmet har.

3.1.1 Alternative løsninger

An alternativ måde at løse problemstillingen på ville være at fokusere på andre måder at booke billetter på. Vi har valgt at man først skal vælge film, derefter tid og sæder. Man kunne have valgt at gøre det muligt for brugeren at vælge data eller tid før valg af film. Vi har blot valgt den løsning vi syntes var mest brugervenlig. En anden mulighed er at man kunne have lavet en liste der viste *alle* forestillinger sorteret efter spilletid - uden at tage hensyn til sortering i forhold til film. Denne mulighed syntes vi designmæssigt ville fungere meget rodet. Derudover kunne vi have valgt at gemme vores reservationer i programkoden frem for i databasen. Vi diskuterede dette i begyndelsen af projektet, men fandt det for besværligt at arbejde med, da det ville kræve store mængder kode, fyldte mere plads samt resultere i at vores program formentlig ville brugere længere tid på at compile og kører.

3.2 Databasedesign

En af de to overordnede tabeller i vores database er *cinemas* som indeholder information omkring vores biografsale. Dette omfatter salens navn, id, antal rækker og

antal sæder i hver række. Vores sæder er ikke repræsenteret som en tabel i vores database, men derimod gemt som et 2 dimentionelt array.

Den anden vigtige tabel er *movies* som indeholder information omkring de film der går i biografen. Det eneste information denne tabel holder er navn og id.

I begge tabel er det, det pågældende id som fungerer som nøgle. Altså sal_id og movie_id.

De to tabeller *movie* og *cinema* bliver koblet sammen i tabellen *show*. Her bruges deres nøgler til at sammensætte de forskellige forestillinger som vores biograf kan vise. I shows tilføjes så et nyt ide til de enkelte forestillinger samt et timestamp - som er det tidspunkt forestillingen skal begynde.

Kapitel 4

Brugervejledning

4.1 Programmet

Programmet er designet med 3 tabs i toppen. *Forestillinger*, *Reservation* og *Ret reservation*. Når programmet åbnes starter brugeren i *Forestillinger*-vinduet og der vises en liste i venstre side med de film som kører i biografen. Brugeren klikker på den film, som kunden ønsker at reservere billetter til. Når en film vælges bliver tiderne for filmforestillinger vist for den pågældende film.

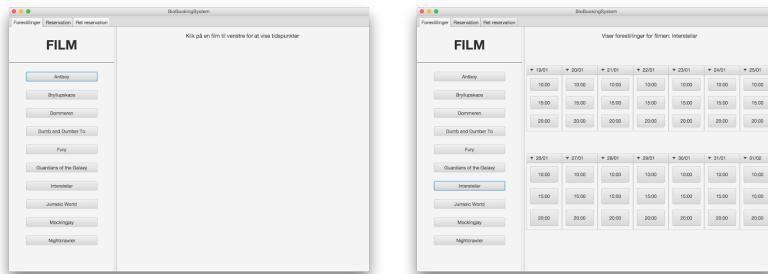


Figure 1. Forestillinger-vinduet før og efter en film er valgt

Klikker ekspedienten derimod ikke på noget, vil der stå følgene tekst *Klik på en film til venstre for at vise tidspunkter*, og har man ikke først valgt film og tidspunkt vil *Reservation* være ubrugelige.

Efter at have valgt film, vælger brugeren tidspunktet, som kunden ønsker at reservere til. Når en forestilling vælges bliver brugeren videreført til næste tab - altså *reservation*. Her vises et vindue med firkanter som illustrerer sæderne i biografen. Ledige sæder er illustreret som en grøn firkant og optagede sæder er illustreret som røde sæder. Brugeren klikker på de sæder som kunden ønsker at reservere - brugeren kan desuden holde musen nede og hove musen over de sæder som skal vælges. De valgte sæder skifter farve til blå og brugeren kan se hvor mange sæder vedkommende har markeret nederst i venstre hjørne.

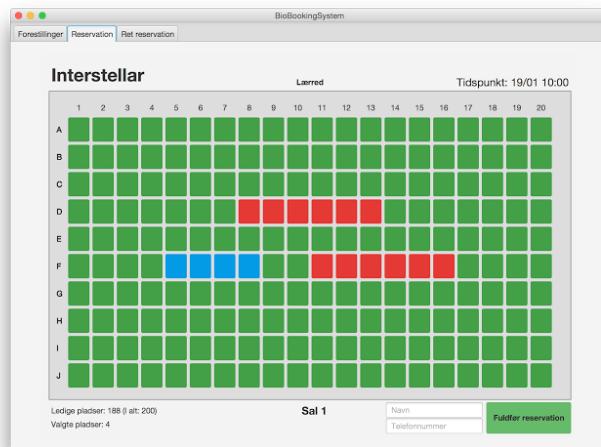


Figure 2. Biograf sal med reserverede sæder i rødt, samt sæder brugeren er ved at reservere i blå

Når sæderne er valgt skrives kundens navn og telefonnummer ind i tekstfeltene nederst i højre hjørne. Afslutningsvist trykkes der på knappen *Fuldfør reservation*.

Hvis kunden fortryder sin bestilling eller har ændringer til en eksisterende bestilling, så klikker brugeren på tabben *Ret reservation* i toppen af programmet. Brugeren bliver taget til et vindue hvor kundens telefonnummer skal indtastes. Når telefonnummeret er indtastet og der er trykket på Enter-tasten eller på *Tjek reservationer*-knappen, så vises en liste nedenunder med de reservationer som kunden har tilknyttet til sit telefonnummer.

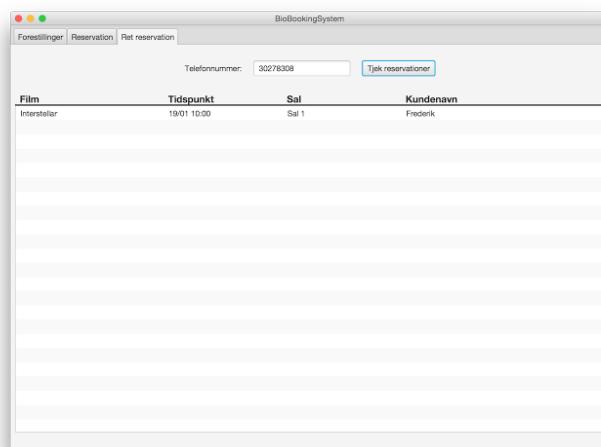


Figure 3. Forestillinger-vinduet før og efter en film er valgt

Der vises kun reservationer for forestillinger som ikke er blevet vist - gamle reservationer vises altså ikke. Brugeren dobbeltklikker på den reservation der skal rettes og et nyt vindue åbner som ligner reservationsvinduet.

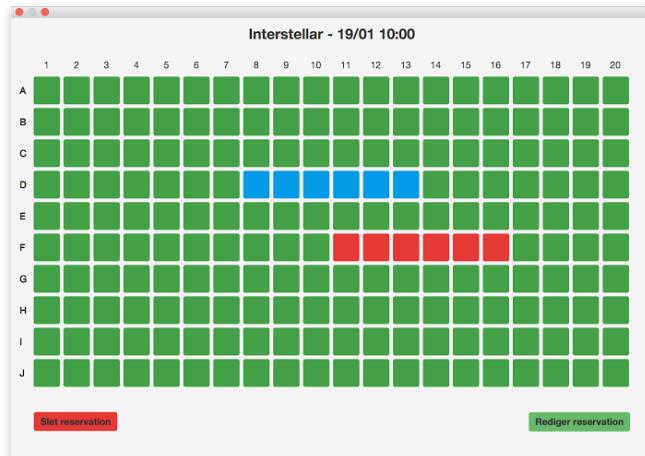


Figure 4. Vinduet hvor brugeren retter sin reservation

De optagede sæder er røde, de ledige er grønne og de sæder som er tilknyttet til den pågældende reservation vises som blå. Brugeren kan tilføje eller slette sæder. Et valgt er blåt og markeringen fjernes ved at klikke på sædet igen. Når ændringen til reservationen er valgt, så klikker brugeren på *Rediger reservation*. Skal reservationen derimod slettes helt, så klikker brugeren på *Slet reservation*.

4.1.1 Begrænsninger

4.1.2 Fejlmeddelelser

4.2 Eksempel

Kapitel 5

Teknisk Analyse

5.1 Model View Controller

5.1.1 Moduler

Programmet har 2 klasser: DBConnect og buildHolder. DBConnect foretager alt der har med databasen at gøre. Dette er eksempelvis databaseforespørgsler hvor alle film returneres i et LinkedHashMap, hvor data såsom reservationer, salstørrelse mv. for en given forestilling hentes, hvor data såsom reservationer indsættes eller opdateres i databasen. Den væsentligste del af DBConnect-klassen må være funktionen getCon, der sørger for at der er en gyldig forbindelse til databasen. buildHolder er et objekt hvor data for en given forestilling lagres. Der bruges getters og setters, så de data der lagres også kan hentes efterfølgende. buildHolder objektet bruges efterfølgende til at bygge reservationssalen med data som filmnavn, tidspunkt for forestillingen, størrelsen på salen (rows/columns), reserverede sæder mv.

Det er controlleren der gør brug af disse objekter, når data skal visualiseres og altså fremvises for brugeren.

5.1.2 Datastrukturer

5.1.3 Algoritmer

5.2 Brugergrænsedesign

5.2.1 Begrænsninger

Kapitel 6

Afprøvning

6.1 Brugerafprøvning

6.2 Unitet

Vores klasse *DBConnectTest* er en testklasse indeholdende unitets. Den primære funktion for for disse test er at teste om vores programtekst snakker rigtigt sammen med databasen.

Testen *testGetMovies()* tester om film og ide passer sammen. Dette gøres ved at vi giver testen et forventet uddata, som er bestemte id's tilknyttet bestemte film. Når testen kører, testes derfor om det forventede uddata passer overens med det faktiske uddata.

testTimeStamp() tjekker hvorvidt en given films forestillinger bliver sorteret korrekt efter tid. Det gøres ved at lave et while-loop som kører igennem alle tids punkterne og tjekker om det nuværende timestamp er lavere end det foregående.

For at tjekke om et reserveret sæde vise med rød farve, og dermed ikke har nogle funktioner. Sædernes funktioner er nemlig tilknyttet deres farve, således at de røde sæder ikke er mulige er klikke på. *testReservedSetColor()* henter et reserveret sæde og derefter sætter dens farve ligmed den forventede røde farve.

testInsetReservation() tester om metoden *InsetReservation()*

6.3 Resultat

Kapitel 7

Konklusion

Kapitel 8

Litteratur

Kapitel 9

Bilag