Hjemmeside med mobilapplikation og vurderingssystem

Diplom IT Eksamensprojekt 2011

Studerende: **Christian Esbensen og Simon Gomez**

Studienummer: **s070050 og s072656**

DTU Vejleder: **Finn Gustafsson**

Eksamensprojekt: **Hjemmeside med mobilapplikation og vurderingssystem**

Afsluttende eksamensnummer: **IMM – B.Eng – 2010 -70**

Diplomingeniør retning: **IT**

Afleveringsdato: **30. juli 2011**

Danmarks Tekniske Universitet

Institut for Informatik og Matematisk Modellering

Bygning 305, DK – 2800 Kongens Lyngby, Danmark

Telefon +45 45 25 33 51, Fax +45 45 88 26 73

reception@imm.dtu.dk

www.imm.dtu.dk

IMM – B.Eng – 2010 – 70

Abstrakt

Prototype hjemmeside der giver overblik over steder, man kan hygge og feste I København.

Vurderingssystem der gør det muligt at give en karakter for stedet, som derefter vil blive vist for andre brugere

Prototype mobil applikation der kan benytte hjemmesiden.

Solidt design til af begge dele for fremtidig videreudvikling.

Forord

Eksamensprojektet er et afsluttende projekt på linjen; diplomingeniør IT på DTU (Danmarks tekniske universitet). Sammenlagt har projektet taget 18 uger og svarer til 20 ECTS point. Vejleder på projektet har været Finn Gustafsson fra Institut for Informatik og Matematisk Modellering DTU.

Christian Esbensen _____

Simon Gomez _____

Lyngby, juni 2011

Indholdsfortegnelse

Indholdsfortegnelse 5
1 Indledning 9
2 Problem 10
2.1 Problemstilling 10
2.2 Problemformulering 10
3 Planlægning 11
3.1 Metode 11
3.1.1 Unified Proces (UP) 11
3.1.2 SCRUM-metoden 12
3.2 Tidsplan 13
3.3 Afvigelse fra tidsplanen 14
3.4 Værktøjer 14
3.5 Delkonklusion 14
4 Analyse 15
4.1 Kravspecifikation 15
4.1.1 Produktkrav 15
4.1.2 Udvidede krav 16
4.1.3 Ikke funktionelle krav 17
4.1.4 Funktionelle krav 18
4.1.5 Use case diagram 18
4.1.6 Use case beskrivelser 19
4.2 Validering 24
4.2.1 Opret bruger 24
4.2.2 Log ind 25
4.2.3 Anmeldelser 25
4.3 Delkonklusion 25
5 Design 26
5.1 Sekvensdiagrammer 26
5.1.1 Besøgende – Søg på venue 26
5.1.2 Besøgende – Log på 27
5.1.3 Besøgende - Opret bruger 27
5.1.4 Medlem – Se kalender 28

5.1.5 Medlem – Redigér oplysninger ... 28
5.1.6 Medlem – Vurdér venue ... 29
5.1.7 Medlem – Slet vurdering ... 29
5.1.8 Medlem – Skriv anmeldelse ... 30
5.1.9 Medlem – Log ud ... 30
5.1.10 Medlem – Opret venue ... 31
5.1.11 Venue – Opret event ... 31
5.1.12 Venue – Slet venue ... 32
5.1.13 Mobil app bruger – Log på ... 32
5.2 Navigationsdiagram ... 33
5.3 Systemoversigt ... 34
5.3.1 UML Klassediagram ... 34
5.3.2 Database entiteter ... 34
5.3.3 Deltagere ... 36
5.4 Layout ... 36
5.4.1 Navngivning og domænenavn ... 36
5.4.2 Navigation af hjemmeside ... 36
5.4.3 CSS ... 36
5.4.4 Skærm Opløsning ... 37
5.4.5 Mobil applikation GUI ... 37
5.5 Delkonklusion ... 37
6 Implementering ... 38
6.1 Valg af værktøjer ... 38
6.1.1 Java Server Pages (jsp) ... 38
6.1.2 Dojo ... 39
6.1.3 Server ... 39
6.1.4 Android ... 39
6.1.5 Eclipse ... 39
6.2 Afgrænsning af implementeringen ... 40
6.3 DataProject ... 40
6.4 DBCommModule ... 41
6.5 ControllerProject ... 42
6.6 Servers ... 43

6.7 iPartyAt Webprojekt 44
6.7.1 Implementering af use case – Søgning på alle venues og events 44
6.7.2 Implementering af use case – Log ind 45
6.7.3 Implementering af use case – Log ud 46
6.7.4 Implementering af use case – Opret bruger 46
6.7.5 Implementering af use case – Se kalender 47
6.7.6 Implementering af use case – Opret Venue 48
6.7.7 Implementering af use case – Opret event 49
6.7.8 Implementering af use case – Tilmeld sig event 49
6.7.9 Implementering af use case – Rate Venue 49
6.7.10 Validering af bruger input 50
6.8 Implementering af mobil applikation 51
6.8.1 Implementering af use case – Log in 51
6.8.2 Browsing – Generelt 53
6.8.3 Implementering af use case – Rate venue 55
7 Test 57
7.1 Browser test 57
7.1.1 Resultat 57
7.2 Blackbox test 58
7.2.1 Test case – Besøgende 59
7.2.2 Test case – Medlem 59
7.2.3 Test case – Bruger Venue 60
7.2.4 Test case – Validering 60
7.2.5 Test case – Mobil app brugere 62
7.3 Konklusion af test 63
8 Forbedringer 64
8.1 Kravændring 64
8.2 Validering 64
8.3 Funktioner og rettelser 64
9 Offentliggørelse 66
9.1 Layout 66
9.2 Webhotel og domæne 66
9.3 Reklame 66

9.4 Release plan 67
10 Udvidelser 68
10.1 Søgefunktion 68
10.2 Kalender 68
10.3 Venue information 68
11 Konklusion 69
12 Litteraturliste 70
13 Bilag 71
13.1 Vedlagt DVD 71
13.2 Bilag til test 71
13.2.1 Blackbox test 71
13.2.2 Browser Test 87
13.3 Bilag til udskrift af XML 88
13.3.1 All Events XML udskrift 88
13.3.2 All Venues XML udskrift 88
13.3.3 User XML udskrift 89

1 Indledning

Dette projekt er baseret på en ide, der er blevet udviklet, formet og sidenhen begrænset.

Vi har i længere tid funderet over, hvorfor der aldrig er blevet udviklet noget der kunne give brugere et bedre overblik over det danske aften-natteliv. Både mht. praktiske informationer, men også for at se hvad andre mennesker mener. Opgaven om at lave lige netop dette, er enorm, men samtidig en udfordring vi mente ville passe perfekt til et eksamensprojekt.

Rapporten er delt op i hovedafsnittene:

- Problem
- Planlægning
- Analyse
- Design
- Implementering
- Test
- Forbedringer
- Udvidelser
- Konklusion

Projektet er delt op i to dele, da vi både laver en hjemmeside, samt en mobil applikation der kan benytte hjemmesiden. Vi har først og fremmest prioriteret funktionaliteten af hjemmesiden, da det er mest essentielt at lige netop denne virker, før mobil applikationen har nogen betydning. Derudover betyder prioritering af funktionalitet også, at der vil blive lagt mindre vægt på layout og den grafiske brugerflade.

Målet for projektet er at få lavet en protype, der kan videreudvikles inden en offentliggørelse(release) af hjemmesiden. Da det er venuet selv der står for oprettelse på siden, er det essentielt at det hele virker som det skal og at førstehåndsindtrykket overbeviser brugerne om at benytte siden.

Hjemmesiden omhandler barer, diskoteker og spillesteder. Der refereres til disse når der står ”venues” i opgaven.

Brugerne af sitet er delt op i to forskellige typer. Dem der har oprettet sig på sitet kaldes brugere og dem der ikke har oprettet sig, men befinder sig eller benytter sitet, kaldes besøgende.

2 Problem

2.1 Problemstilling

I dagens Danmark kan det være en udfordring, at finde ud af hvor man kan tage i byen og opfylde bestemte behov. Hvis man i forvejen ikke går meget ud, eller har kendskab til bylivet kan det være endnu sværere. Forskellige behov kan fx være priser, musik, alder, stemning og størrelse. Hvis man er flere der skal ud og der er forskellige ønsker, kan udvalget ofte ende med at blive begrænset, til de steder folk har kendskab til. Frem for et andet sted der eventuelt ville være mere optimalt, men som der ikke var kendskab til.

Der findes mange mennesker der gerne vil lære nye steder at kende, men som måske ikke har tænkt sig at satse sin lørdag aften, på noget der muligvis kan vise sigt at være spild af tid. Der findes andre muligheder for at læse om steder at tage ud, men det kræver som regel at mand ved hvad man søger efter.

2.2 Problemformulering

Vores ide har omhandlet en innovativ løsning til dette problem. En måde hvorpå man kan læse informationer om stedet og se hvad andre syntes om det.

Inden for andre kategorier såsom spisesteder eller cafeer, har en god løsning været et vurderingssystem, samt en måde hvorpå folk kan fortælle om deres oplevelse, eller skrive en anmeldelse. Dette koncept har eksisteret længe og har vist sig som en solid løsning, hvorpå folk kan danne sig et indtryk uden egentlig at have været der. Det innovative består i, at venuen selv opretter sig og har ansvaret for sin egen side.

Projektet består i at lave en prototype af hjemmesiden, samt en prototypeapplikation til en mobiltelefon med Android OS og hertil hørende vurderingssystem. Alt dette skal omhandle diskoteker og barer i København. Det er meningen at der skal være en side per bruger samt en anden side per diskotek/bar. En bruger kan herefter tilkendegive sin mening om dette sted ved at skrive en anmeldelse eller afgive en karakter.

Når man går på bar eller diskotek, kommer man der ikke for at sidde og bruge sin mobiltelefon på nettet. Det bliver derfor en udfordring, at lave det hele så det er så nemt for brugeren som muligt. Det skal derfor være enkelt for en bruger, at vurdere sin aften.

Derudover skal det også være meget let at finde informationer om steder, således man kan gøre det fra sin smartphone.

3 Planlægning

Dette afsnit forklarer om planlægningen af projektet.

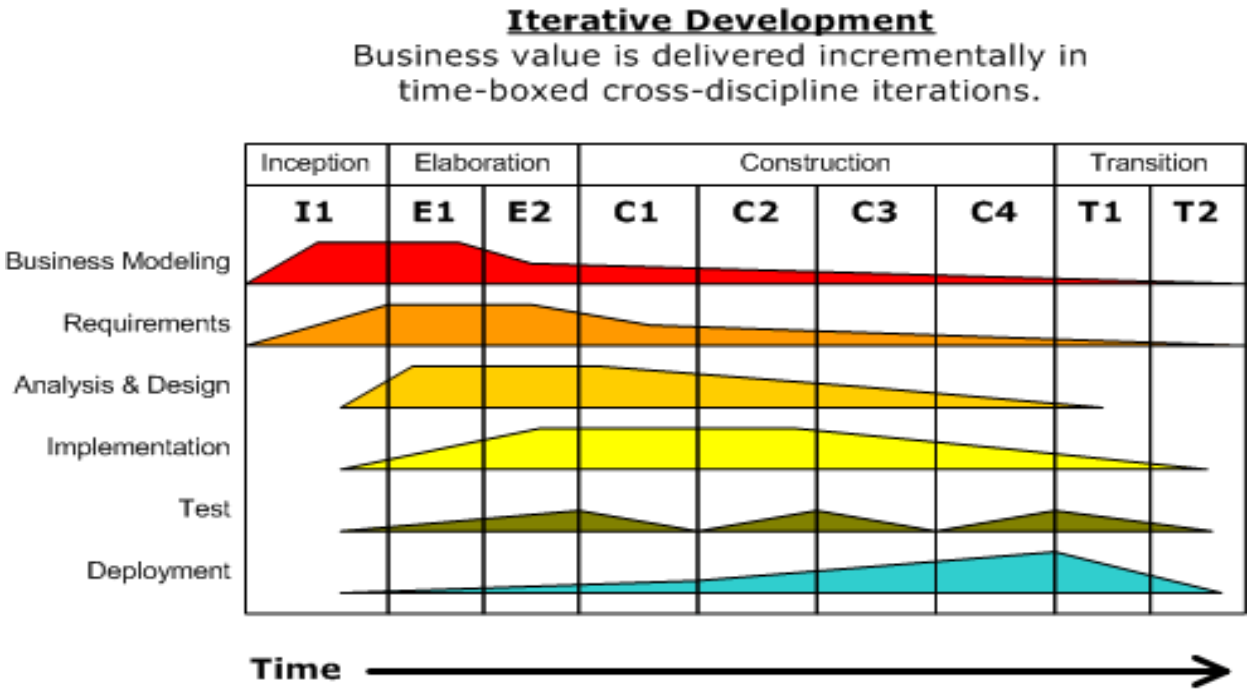
3.1 Metode

Til løsning af projektet har vi valgt en fremgangsmåde der inkludere to arbejdsmetoder, Unified Proces og SCRUM metoden. Metoderne vil hjælpe med at give det nødvendige overblik undervejs og holde hinanden opdateret.

3.1.1 Unified Proces (UP)

Overordnet er metoden delt op i fire faser: forberedelse, etablering, konstruktion og overdragelse. I vores tilfælde vil de fire faser inkludere krav, analyse, design, implementering, test og overdragelse.

En af de ting der gør UP til en foretrukken metode for vores vedkomne, er at der kan arbejdes parallelt mellem faserne. Dvs. at der fx under analyse/design også kan foretages implementering, dog på et ikke så omfattende niveau. Og under implementeringen kan der også arbejdes på analyse/design fasen. Nedenfor ses et typisk UP diagram, det er delt op i de fire faser (kolonnerne) og 6 afsnit, hvoraf vi ikke benytter os af Business Modelling.



Figur 1 - Unified Proces
http://en.wikipedia.org/wiki/File:Development-iterative.gif Unified Process.

3.1.1.1 Forberedelse

Den første fase omhandler forberedelse, formålet med denne er at analysere og danne et overblik over kravene. Dette vil hjælpe med at give en bedre forståelse for, hvad der egentlig skal laves. Dernæst skal hovedfunktionerne identificeres og beskrives. For at danne et bedre overblik over produktet, skal der laves et abstrakt design af arkitekturen.

I forberedelsesfasen skal vi også klargøre for hvilke udviklingsværktøjer vi vil benytte.

3.1.1.2 Etablering

Under etableringsfasen skal der udvikles nogle centrale dele af systemet, for at give et indblik i arkitekturen i praksis. Ved en tidlig prototype udvikling nedsættes risikoen for potentielle store fejl og gør det lettere at foretage større ændringer ved designet.

3.1.1.3 Konstruktion

Projektet implementeres og testes hovedsageligt i denne fase.

3.1.1.4 Overdragelse

Dette er den sidste fase inden aflevering. Der skal overvejes om projektet opfylder de stillede krav og eventuelle fejl rettes. Dokumentering skal helst ske undervejs, hvorefter det rundes af i denne fase.

3.1.2 SCRUM-metoden

Vi har også valgt at udnytte dele fra metoden, der er kendt som SCRUM. I denne metode findes nogle roller som skal overholdes, men da vi kun er 2 personer i dette projekt har vi valgt at disse roller frafalder. I SCRUM metoden findes 4 forskellige slags møder. Disse møder er: Daily sprint, Sprint planlægningsmøde, Sprint reviewmøde samt Sprint retrospektiv møde.

Vi har valgt at benytte os af 2 af disse, nemlig, Sprint planlægningsmøde og Sprint reveiwmøde. Dels fordi vi ikke ses hverdag, grundet arbejde og uddannelse, men også fordi vi ikke følte behov for et retrospektivt møde.

Grunden til vi også har valgt at bruge SCRUM til udviklingen af vores projekt, sammen med UP, er at vi får en detaljeret tidsplan og et rigtig godt overblik over hvad der skal laves. Dette overblik fås ved at dele projektet op i sprints (se afsnit nedenfor) og under disse sprints vil vi udnytte UP princippet.

3.1.2.1 Sprint planlægningsmøde

Først skal projektet deles op i sprints. Disse sprints indeholder en del af projektet som skal laves inden for den tid der er afsat til at få gennemført et sprint. Denne del kaldes for en backlog og indeholder endnu mindre dele af projektet. Disse mindre kan for eksempel være en knap på siden samt funktionaliteten bag denne. For at dette skal løkkes kræver det at projektet har en detaljeret kravspecifikation.

Til dette møde bliver en backlog defineret og estimeret. Dette gøres ved at udviklerne sammen diskuterer hvad denne backlog indeholder samt hvordan denne løses.

3.1.2.2 Reviewmøde

Dette møde afholdes efter hvert sprint. Ved dette møde gennemgås hvad der er nået i foregående sprint og hvad der ikke er nået. Det der er nået demonstreres for hvem end det må vedkomme.

3.2 Tidsplan

Til at starte med blev der formuleret en problemformulering der dannede rammen om opgaven. Derefter blev projektet delt op i fire faser, analyse, design, implementering og test. Der blev derefter lavet en projektplan der kunne styre projektets fremgang, som ses på tabel.1. De forskellige tegn har følgende betydning:

- S = Sprintplanlægnings møde
- R = Reviewmøde
- X = planlagte tidsforløb
- O = faktiske tidsforløb

	Uge 9	Uge 10	Uge 11	Uge 12	Uge 13	Uge 14	Uge 15	Uge 16	Uge 17	Uge 18	Uge 19	Uge 20	Uge 21	Uge 22	Uge 23	Uge 24	Uge 25	Uge 26
Planlægning	S		RS		RS		RS		RS		RS		RS		RS		RS	
Tidsplan	X O																	
Metode	X O																	
Problemformulering	X O																	
Analyse																		
Kravspecifikation		X O																
Use Case diagrammer		X O	X O					O		O								
Design																		
Sekvensdiagrammer				X	o	O		O		O								
UML				X		O												
Database design				X O	X O													
Implementering																		
Afgrænsning af design					X		O											
Databasen					X O													
Opsætning af hjemmeside						X O	X O	X O	X O	X O	X O	O	O	O				
						X	X	X	X	X	X							
Applikation												X	X	X	O	O		
Test																		

	Uge 9	Uge 10	Uge 11	Uge 12	Uge 13	Uge 14	Uge 15	Uge 16	Uge 17	Uge 18	Uge 19	Uge 20	Uge 21	Uge 22	Uge 23	Uge 24	Uge 25	Uge 26
Blackbox testing									X	X O	X	X O	X	X O		O	O	
Dokumentation																		
Rettelse af det der allerede er skrevet															X	O		
Færdiggøre rapport																X	X O	O
Konklusion																		X O
Print/overdragelse																		X O

Tabel 1 - Tidsplan

Da vi havde bestemt os for at benytte os af Unified Process metoden, ville det betyde at der ville blive arbejdet på tværs af faserne. Det betyder at faserne ikke er fastlåst, selvom vi bevæger os ind i en ny fase.

3.3 Afvigelse fra tidsplanen

Fra uge 9 til uge 11 blev tidsplanen overholdt, men grundet mere arbejde fra job og studie, skred den fra uge 12 af. Det betød at implementeringen af hjemmesiden først blev færdig i uge 22 og mobil applikationen i uge 24. Dette resulterede i at testen først kunne færdiggøres i uge 25. Der blev derfor også testet en del mere undervejs end planlagt. Det at tidsplanen ikke blev overholdt, medførte ingen større konsekvenser, da vi alligevel skulle arbejde på tværs af faserne. Dog opfylder nogle af funktionerne ikke kravene fuldt ud.

3.4 Værktøjer

Til udvikling af projekter har vi valgt at benytte følgende værktøjer

- Eclipse – Primære udviklingsplatform
- Mowes – Database server
- Tomcat – Webserver

Web delen og mobildelen vil blive implementeret i Java og database delen i MySQL. Der vil blive uddybet mere om dette i implementeringsafsnittet.

3.5 Delkonklusion

Det at vi har erfaring med UP metoden, gjorde at vi kunne føle os tilpas med måden at arbejde med den på. Vi har under hele projektet arbejdet på tværs af faserne bortset fra de sidste 2 uger, hvor der udelukkende blev arbejdet på rapporten.

Vi havde valgt at benytte SCRUM metoden for at kunne danne os et bedre overblik vha. sprintmøder og reviewmøder. Disse møder gik ikke helt som planlagt. Vores sprints blev ikke rigtig defineret så grundigt og gik mere ud på at prøve at overholde tidsplanen. Dette kan skyldes at ingen af os havde benyttet metoden før til noget uddannelsesrelateret arbejde.

4 Analyse

I dette afsnit defineres, beskrives og analyseres kravene til hjemmesiden og mobil applikationen.

4.1 Kravspecifikation

4.1.1 Produktkrav

Kravene er delt op 2 grupper: ”Need to have” er de funktioner der er nødvendige for at prototypen kan give et realistisk billede på hvad produktet går ud på. ”Nice to have” er de funktioner der kommer i anden række. Det er funktioner der ikke er essentielle for at kunne demonstrere prototypen, men som alligevel skal implementeres som noget af det første i det endelige produkt.

4.1.1.1 Hjemmeside

Vi ønsker at lave en hjemmeside der understøtte følgende funktioner.

4.1.1.1.1 Need to have

- Søg på venue – Alle kan søge efter en venue
- Brugeroprettelse – En besøgende kan oprette sig på siden vha. sin email
- Venueoprettelse – En brugerskal kunne oprette en venue
- Opret event – En bruger skal kunne oprette et venue event
- Tilmeld event – En bruger skal kunne tilmelde sig events
- Log ind – En bruger skal kunne logge sig på siden
- Log ud – En bruger skal kunne logge sig af igen
- Vurderingsfunktion – Brugere skal kunne tilkendegive en karakter om en venue
- Anmeldelser – En bruger skal kunne skrive en anmeldelse om en venue
- Validering af brugerinput – Det meste brugerinput skal valideres

4.1.1.1.2 Nice to have

- Redigering af brugeroplysninger
- Redigering af venueoplysninger
- Sletning/redigering af vurdering og anmeldelse

Need to have funktionerne er grundlæggende for projektet og prioriteres derfor højt

4.1.1.2 Mobil applikation

Vi ønsker at lave en mobil applikation der understøtter følgende funktioner

4.1.1.2.1 Need to have

- Log ind
- Log ud
- Søg på venue
- Vurder venue

4.1.1.2.2 Nice to have

- Sletning af vurdering
- Se kalender

Formålet med mobil applikation er, at gøre det lettere for brugere at vurdere en venue og benytte sig af siden.

4.1.2 Udvidede krav

Hvis alle kravene listet under produktkrav er implementeret tilfredsstillende og der er tid til det, ønskes der følgende

- Avanceret søgefunktion - Således man kan søge på venueinformationer og på den måde finde venues.
- CSS – Noget layout der kan give en ide om hvordan siden skal se ud/fungere

4.1.3 Ikke funktionelle krav

Hardware krav for PC:

1. 2 GHz processor
2. 2 GB RAM
3. 3 GB ledig harddisk plads
4. Netværkskort med min 10/100 Mbps
5. Skærmopløsning på min 1024x768

Softwave krav:

1. Webserver der kan køre JSP
2. Database server der kan køre MySQL
3. Java Runetime Enviroment (JRE) version 1.6 eller nyere skal være installeret.
4. Java Server Standard Tag Library(JSTL) version 1.2 eller nyere skal være installeret.
5. Jaxb 2.1.8 eller nyere installeret

Hardware krav for smartphone:

1. Adgang til internet via enten WiFi eller 3G
2. Minimum 1MB ledig plads

Softwave krav mobil:

1. Android 2.2 eller nyere

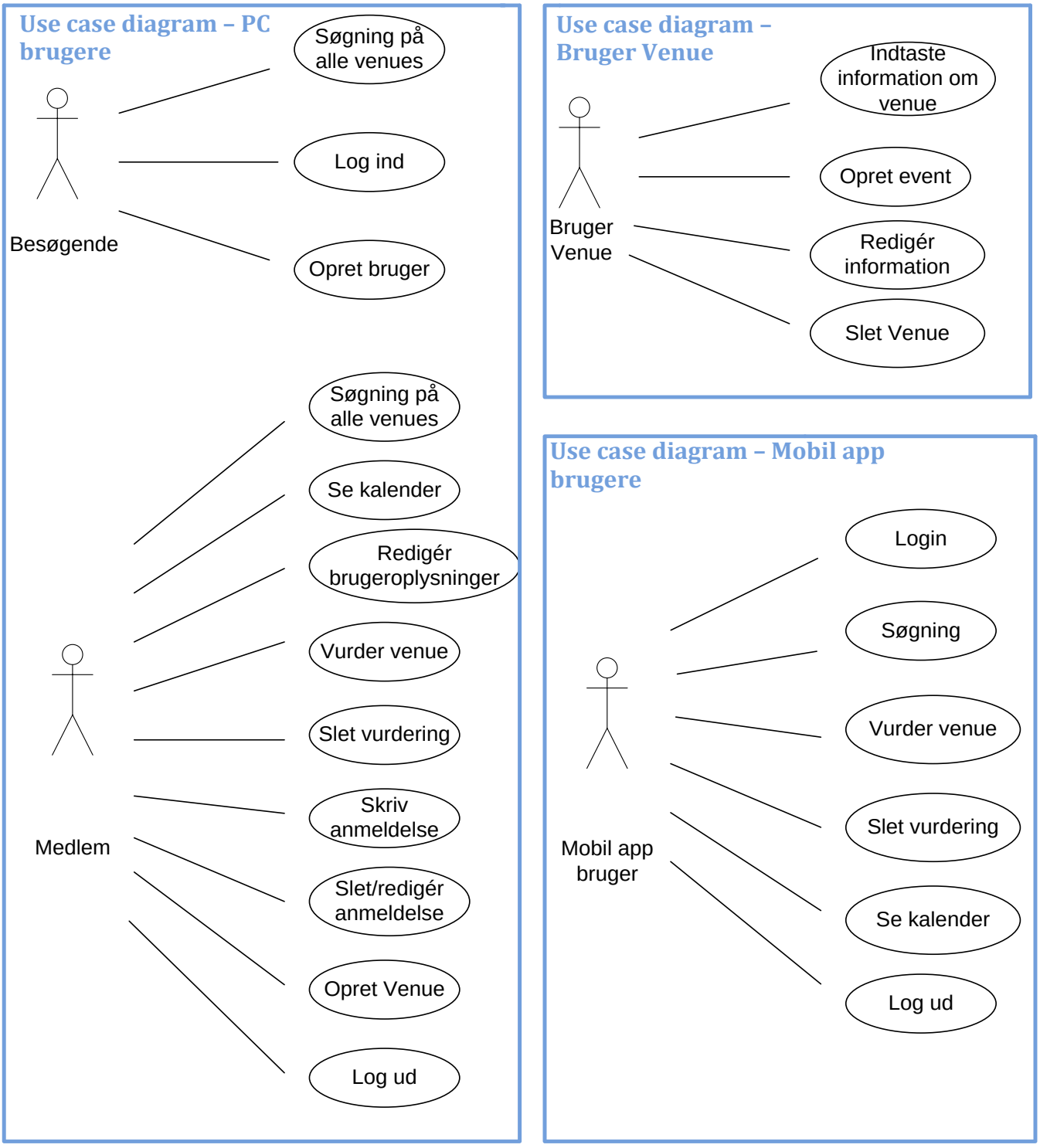
Hardware kravene for harddiskplads vil ændre sig i fremtiden, alt afhængigt af hvordan databasen vokser. For smartphonen, vil kravet om min 1MB ledig plads også kunne komme til at ændre sig, så snart det grafiske bruger interface udvides.

4.1.4 Funktionelle krav

De funktionelle krav er illustreret vha. use cases for bruger, venue og mobil app bruger, samt dertilhørende use case beskrivelser. "Nice to have" kravene er også inkluderet.

4.1.5 Use case diagram

Der er blevet identificeret følgende use cases:



Figur - 2 - Use case diagram

4.1.6 Use case beskrivelser

Alle use casene følger "happy path". Dvs. at brugeren udfylder informationerne korrekt og systemet fungerer uden fejl.

4.1.6.1 PC - Brugere

4.1.6.1.1 Besøgende – Søgning på alle venues

Primær aktør	**Besøgende**
Prækondition	Brugeren er inde på hjemmesiden
Aktør handling	**System respons**
Brugeren indtaster en tekst i søgefeltet	Gennemsøger databasen for venues og fremviser resultaterne
Brugeren vælger et af resultaterne ved at klikke	Videresender brugeren til siden for det pågældendes sted
Postkondition	Brugeren ender inde på sitet for det søgte sted

4.1.6.1.2 Besøgende – Log ind

Primær aktør	**Besøgende**
Prækondition	Brugeren er inde på hjemmesiden
Aktør handling	**System respons**
Brugeren indtastet en email og et kodeord i email - og kodeords felterne	Undersøger om brugeren eksisterer og om kodeordet er korrekt. Videresender brugeren til topsitet, som "logged on"
Postkondition	Brugeren logges på systemet

4.1.6.1.3 Besøgende – Opret bruger

Primær aktør	**Besøgende**
Prækondition	Brugeren er inde på hjemmesiden
Aktør handling	**System respons**
Brugeren vælger "opret bruger"	Systemet sender brugeren til brugeropretnings site, hvor der skal udfyldes felter
Brugeren udfylder felterne	Opretter brugeren og sender en aktiverings email
Postkondition	Brugeren oprettes i systemet og kan logge på med sin email og kodeord

4.1.6.2 Medlem

4.1.6.2.1 Medlem - Søgning på alle venues

Primær aktør	**Medlem**
Prækondition	Brugeren er logget på systemet
Aktør handling	**System respons**
Brugeren indtaster en tekst i søgefeltet	Gennemsøger databasen for resultater og fremviser dem
Brugeren vælger et af resultaterne ved at klikke	Videresender brugeren til sitet for det pågældendes sted
Postkondition	Brugeren ender inde på sitet for det søgte sted

4.1.6.2.2 Medlem – Se kalender

Primær aktør	**Medlem**
Prækondition	Brugeren er logget på systemet
Aktør handling	**System respons**
Brugeren trykker på sin kalender	Systemet videresender brugeren til brugerkalender siden
Postkondition	Brugeren ser sin brugerkalender

4.1.6.2.3 Medlem – Redigér brugeroplysninger

Primær aktør	**Medlem**
Prækondition	Brugeren er logget på systemet
Aktør handling	**System respons**
Brugeren trykker på rediger brugeroplysninger	Systemet videresender brugeren til brugeroplysningssiden
Brugeren udfylder derefter felterne efter ønske og trykker gem	De indtastede oplysninger overskriver de gamle, således det kun er de nye der gemmes
Postkondition	Brugerens oplysninger ændres og gemmes

4.1.6.2.4 Medlem – Vurder venue

Primær aktør	**Medlem**
Prækondition	Brugeren er logget på systemet og er inde på sitet for en venue. Brugeren har ikke vurderet stedet før
Aktør handling	**System respons**
Vælger en karakter og trykker på vurder knappen	Registrerer karakteren og gemmer den
Postkondition	Brugerens karakter bliver gemt og tilføjet til gennemsnitskarakter udregningen.

4.1.6.2.5 Medlem – Slet vurdering

Primær aktør	**Medlem**
Prækondition	Brugeren er logget på systemet og er inde på sitet for en venue. Brugeren har afgivet vurdering før
Aktør handling	**System respons**
Vælger slet vurdering	Den tidligere vurdering slettes
Postkondition	Vurderingen slettes og brugeren har muligheden for at indtaste en ny

4.1.6.2.6 Medlem – Skriv anmeldelse

Primær aktør	**Medlem**
Prækondition	Brugeren er logget på systemet og er inde på sitet for en venue.
Aktør handling	**System respons**
Brugeren skriver en anmeldelse af venuet og vælger tilføj	Anmeldelsen bliver tilføjet til venuets anmeldelser
Postkondition	Brugerens anmeldelse bliver læselig for andre brugere, på sitet for venuet

4.1.6.2.7 Medlem – Slet/redigér anmeldelse

Primær aktør	**Medlem**
Prækondition	Brugeren er logget på systemet og er inde på sitet for en venue. Brugeren har skrevet en anmeldelse af stedet
Aktør	**System respons**
Vælger redigér/slet anmeldelse	Anmeldelsen bliver åbnet og mulig for redigering
Brugeren vælger enten tilføj eller slet	Afhængig af brugerens valg slettes eller tilføjes anmeldelsen
Postkondition	Anmeldelsen bliver enten redigeret eller slettet

4.1.6.2.8 Medlem – Opret Venue

Primær aktør	**Medlem**
Prækondition	Brugeren er logget på systemet
Aktør	**System respons**
Vælger Opret venue	Videresender brugeren til Venueoprettelse siden
Brugeren indtaster informationerne og trykked godkend	Systemet oprettet venuen med de indtastede oplysninger
Postkondition	Brugerens venue oprettes og kan fremover redigere

4.1.6.2.9 Medlem – Log ud

Primær aktør	**Medlem**
Prækondition	Brugeren er logget på systemet
Aktør handling	**System respons**
Trykker på "Log af"	Systemet logger brugeren af
Postkondition	Brugeren logges af systemet og har mulighed for at logge på

4.1.6.3 PC Venue brugere

4.1.6.3.1 Venue– Opret event

Primær aktør	**Medlem - Ejer af venue**
Prækondition	Brugeren er logget på systemet og har oprettet en venue og befinder sig på venue siden
Aktør handling	**System respons**
Brugeren vælger "Opret event"	Systemet sender brugeren videre til opret event side
Brugeren udfylder formularen	Systemet opretter eventet og gemmer informationerne
Postkondition	Et event oprettes og bliver synligt i venue kalenderen

4.1.6.3.2 Venue– Indtast informationer om venue

Primær aktør	**Medlem - Ejer af venue**
Prækondition	Brugeren har oprettet venuen og er inde på venuesiden
Aktør handling	**System respons**
Brugeren udfylder de venue informationer der ikke blev udfyldt under venue oprettelsen	Gemmer de indtastede informationer
Postkondition	De ny indtastede oplysninger gemmes for venuen

4.1.6.3.3 Venue– Rediger oplysninger

Primær aktør	**Medlem - Ejer af venue**
Prækondition	Brugeren har oprettet venuen og er inde på venuesiden
Aktør handling	**System respons**
Brugeren vælger rediger oplysninger	Sender brugeren videre til redigerings side
Brugeren indtaster de oplysninger der ønskes ændret og trykker godkend	Systemet overskriver de gamle oplysninger med de nye
Postkondition	De ny indtastede oplysninger gemmes frem for de gamle

4.1.6.3.4 Venue– Slet venue

Primær aktør	**Medlem - Ejer af venue**
Prækondition	Brugeren er inde på venuesiden af den venue der skal slettes
Aktør handling	**System respons**
Brugeren vælger slet venue	Venuen slettes derefter
Postkondition	Venuen eksistere ikke efter sletning

4.1.6.4 Mobil app bruger

4.1.6.4.1 Mobil app bruger – Log ind

Primær aktør	**Mobil app bruger**
Prækondition	Brugeren har startet applikationen og har en konto i forvejen
Aktør handling	**System respons**
Brugeren indtastet en email og et kodeord i email - og kodeords felterne	Systemet undersøger om brugeren eksisterer og om kodeordet er korrekt. Videresender brugeren til topsitet, som ”logged on”
Postkondition	Brugeren logges på systemet

4.1.6.4.2 Mobil app bruger – søgning på venues

Primær aktør	**Mobil app bruger**
Prækondition	Brugeren har startet applikationen
Aktør handling	**System respons**
Brugeren indtaster en tekst i søgefeltet	Gennemsøger databasen for venues og fremviser resultaterne
Brugeren vælger et af resultaterne	Videresender brugeren til siden for det pågældendes sted
Postkondition	Brugeren ender inde på sitet for det søgte sted

4.1.6.4.3 Mobil app bruger – Vurder på venues

Primær aktør	**Mobil app bruger**
Prækondition	Brugeren er logget på systemet og er inde på sitet for en venue. Brugeren har ikke vurderet stedet før
Aktør handling	**System respons**
Vælger en karakter og trykker på vurder knappen	Registrerer karakteren og gemmer den
Postkondition	Brugerens karakter bliver gemt og tilføjet til gennemsnitskarakter udregningen.

4.1.6.4.4 Mobil app bruger – Slet vurdering

Primær aktør	**Mobil app bruger**
Prækondition	Brugeren er logget på systemet og er inde på sitet for en venue. Brugeren har afgivet vurdering før
Aktør handling	**System respons**
Vælger slet vurdering	Den tidligere vurdering slettes
Postkondition	Vurderingen slettes og brugeren har muligheden for at indtaste en ny

4.1.6.4.5 Mobil app bruger – Se kalender

Primær aktør	**Mobil app bruger**
Prækondition	Brugeren er logget på systemet
Aktør handling	**System respons**
Brugeren trykker på sin kalender	Systemet videresender brugeren til brugerkalender siden
Postkondition	Brugeren ser sin brugerkalender

4.1.6.4.6 Mobil app bruger – Se kalender

Primær aktør	**Mobil app bruger**
Prækondition	Brugeren er logget på systemet
Aktør handling	**System respons**
Vælger "Log af"	Systemet logger brugeren af
Postkondition	Brugeren logges af systemet og har mulighed for at logge på

4.2 Validering

Valideringen er en vigtig del af projektet, da det har ansvaret for at hjemmesiden kan fungere efter hensigten. Der skal valideres på det meste af det input der modtages fra brugerne. Følgende tabeller viser hvad der skal valideres.

4.2.1 Opret bruger

Input til validering	**Obligatorisk ja/nej**	**Tilladte tegn**	**Anden Validering**
Brugernavn	Ja	a-z, A-Z, 0-9, ".", "@"og "_"	Der skal checkes efter om brugernavnet allerede er brugt
Email	Ja	Styres af Dojo library (standard tegn for email oprettelse)	Der skal checkes efter om emailen allerede er brugt
First name	Ja	a-z, A-Z, 0-9, ".", "@"og "_"	At feltet udfyldes og det er med tilladte tegn
Last name	Ja	a-z, A-Z, 0-9, ".", "@"og "_"	At feltet udfyldes og det er med tilladte tegn

Input til validering	Obligatorisk ja/nej	Tilladte tegn	Anden Validering
Password	Ja	a-z, A-Z, 0-9, ”.”, ”@”og ”_”	At feltet udfyldes og det er med tilladte tegn
Re-enter password	Ja	a-z, A-Z, 0-9, ”.”, ”@”og ”_”	Skal stemme overens med ”password”
Gender	Nej	Vælges vha. dropdown menu	Ingen
Birthday	Nej	Vælges vha. dropdown menu	Ingen

Tabel 2 - Validering af opret bruger

4.2.2 Log ind

Input til validering	Obligatorisk ja/nej	Tilladte tegn	Anden Validering
Brugernavn	Ja	a-z, A-Z, 0-9, ”.”, ”@”og ”_”	Brugernavnet eksisterer
Password	Ja	a-z, A-Z, 0-9, ”.”, ”@”og ”_”	Koden passer til brugernavnet

Tabel 3 - Validering af log ind

4.2.3 Anmeldelser

Da alle kan oprette sig på siden og har mulighed for at skrive anmeldelser. Kan siden nemt blive udsat for spam, eller personer der fx reklamere for sig selv på venuesiderne. Det kan være svært at validere på en frit skreven tekst, der skal derfor være minimum en side ansvarlig, som kan holde styr på dette. Derudover skal dem som opretter sig, acceptere nogle betingelser. Overtrædelse af disse kan så medføre advarsler eller bortvisning (blokering af konto). Dette er til fremtidig udvikling og ikke medtaget i kravene.

4.3 Delkonklusion

Kravspecifikationen er blevet udarbejdet og illustreret vha. Use cases og dertilhørende beskrivelser. Til validering er der lavet et skema over hvad der skal inkluderes. Dette kan nu lægge grundlag for et design.

5 Design

I dette afsnit, beskrives hvordan hjemmesiden og mobil applikationen skal designes ud fra use casene, således kravene i kravspecifikationen bliver opfyldt. Vha. sekvensdiagrammer beskrives flowet i systemet for hver use case. UML diagrammerne er designet til at give et overblik over systemets struktur og hvad databasen skal indeholde.

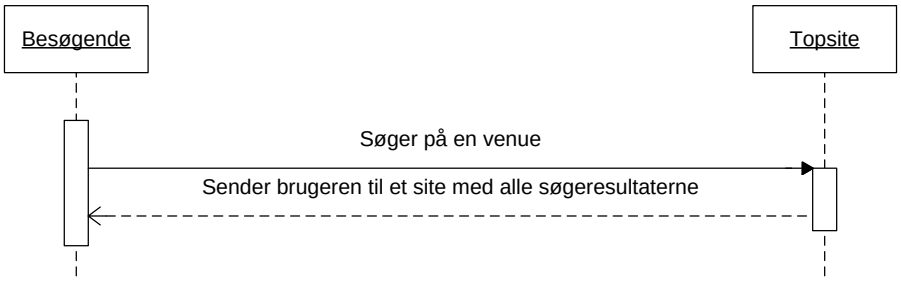
5.1 Sekvensdiagrammer

Følgende viser sekvensdiagrammer for use casene, nogle af use casene er næsten ens og der er derfor kun medtaget den ene i tilfælde der er to der minder om hinanden. De use cases der ikke er lavet sekvens diagrammer for er:

- Medlem – Søg på venues, har samme flow som besøgende søg på venue.
- Medlem – Slet anmeldelse har samme flow som medlem – slet vurdering.
- Venue – indtaste informationer, har samme flow som medlem - indtaste informationer.
- Venue – rediger oplysninger, har samme flow som medlem – rediger oplysninger.

Flowet for mobil app brugere, har samme flow som pc brugere.

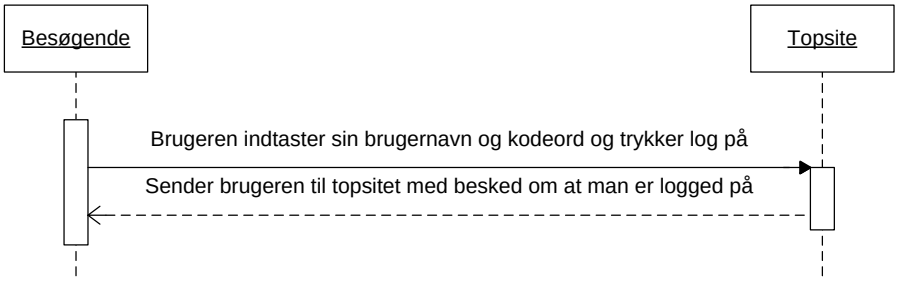
5.1.1 Besøgende – Søg på venue



Figur 3 Sekvensdiagram - Besøgende søg på venue

Dette sekvensdiagram viser flowet for, når en besøgende søger på en venue. På topsitet taster brugeren en søgning ind og trykker søg. Derefter videredirigerer topsitet brugeren til en side med søgeresultaterne.

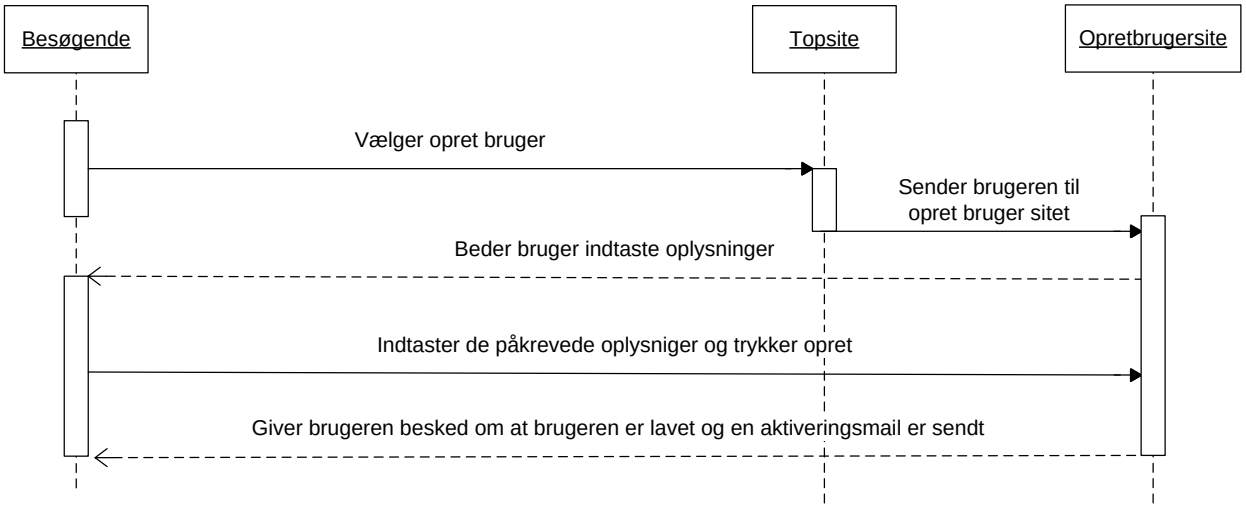
5.1.2 Besøgende – Log på



Figur 4 - Besøgende log på

En besøgende som er medlem kan vælge at logge på ved at indtaste et brugernavn og kodeord. Når dette er gjort valideres det af topsitet, hvorefter brugeren får besked om at han/hun nu er logget på.

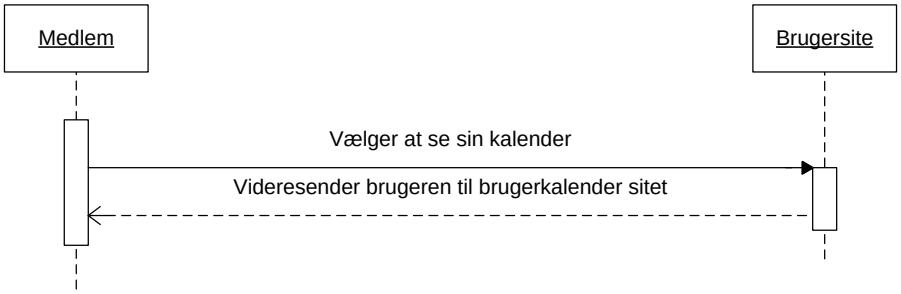
5.1.3 Besøgende - Opret bruger



Figur 5 - Besøgende opret bruger

En besøgende kan oprette en bruger ved hjælp af en opretbrugersite. Når der vælges opret bruger, videresender topsitet den besøgende til opret brugersitet. Dette site beder brugeren indtaster de påkrævede informationer. Når brugeren har gjort dette får han/hun besked om at han/hun er blevet oprettet og at der er sendt en aktiveringsmail.

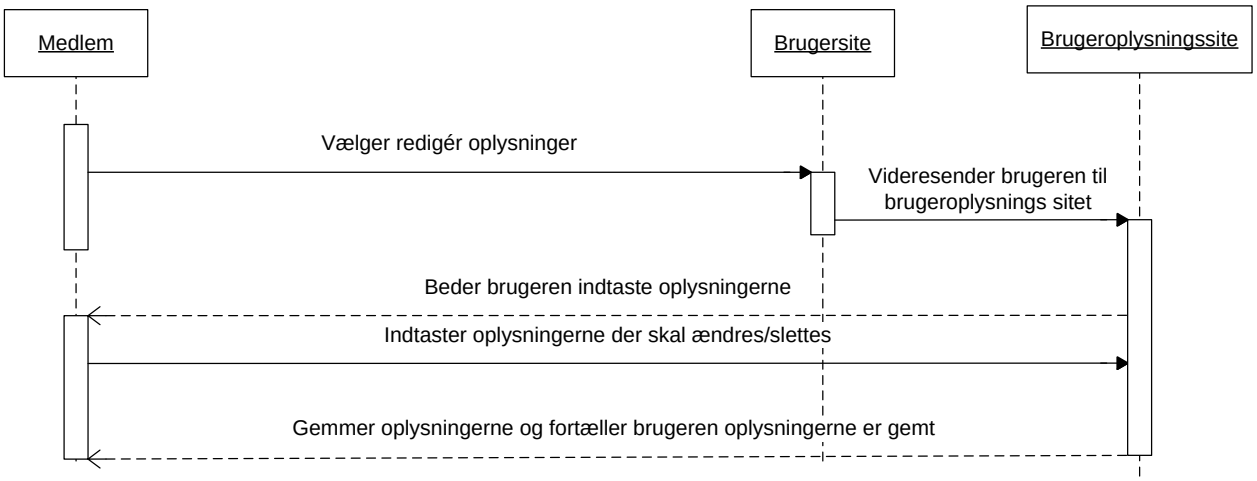
5.1.4 Medlem – Se kalender



Figur 6 - Medlem se kalender

Dette sekvensdiagram viser flowet for når en bruger er logget på systemet og vil se sin egen kalender. Brugeren trykker på sin kalender hvorefter brugersitet videresender brugeren til brugerkalender sitet.

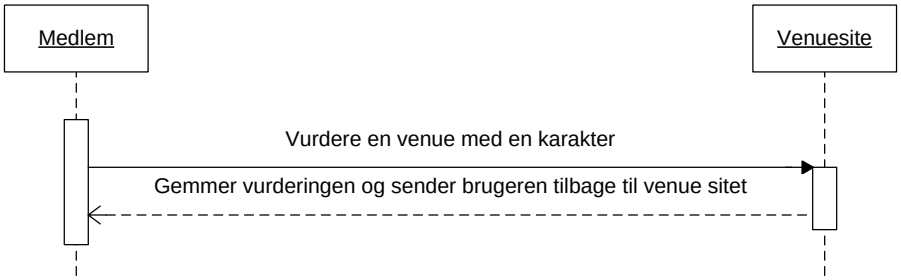
5.1.5 Medlem – Redigér oplysninger



Figur 7 - Medlem redigér oplysninger

Når et medlem vil redigere sine oplysninger, vælger han/hun redigér oplysninger på brugersitet. Brugersitet videresender dernæst brugeren til brugeroplysningssitet. Dette site beder brugeren indtaste de oplysninger der skal ændres, hvorefter den gemmer oplysningerne og giver brugeren besked om at de er gemt.

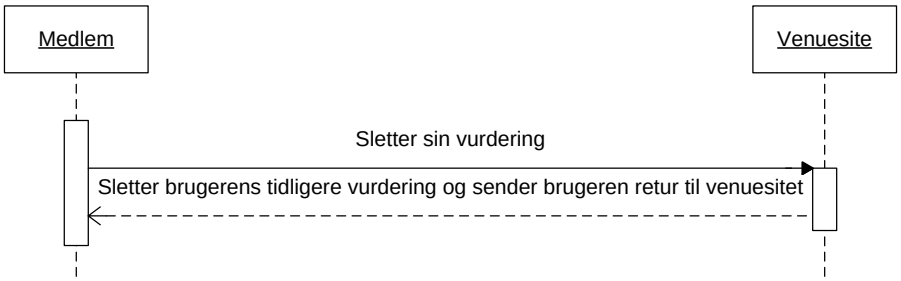
5.1.6 Medlem – Vurdér venue



Figur 8 - Medlem vurdér venue

Når en venue skal vurderes, kræver det at brugeren logget på. Når han er inde på et venuesite og trykker en karakter ind og derefter vurder. Gemmer sitet vurderingen og fjerner muligheden for at vurdere igen, medmindre brugeren sletter sin gamle vurdering.

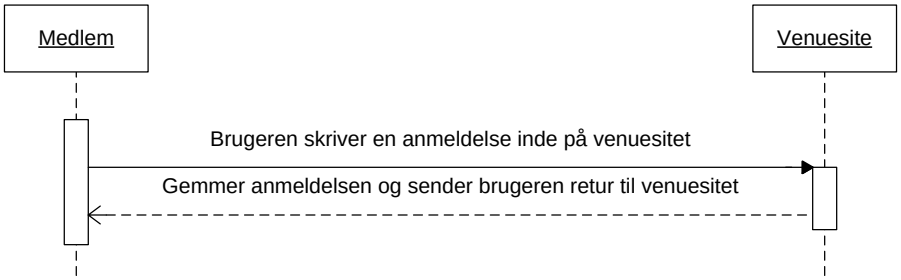
5.1.7 Medlem – Slet vurdering



Figur 9 - Medlem slet vurdering

Når en bruger har afgivet en vurdering, har han/hun muligheden for at slette den igen. Når brugeren gør dette, sletter venuesitet vurderingen og opdatere derefter siden, således brugeren kan afgive en ny stemme.

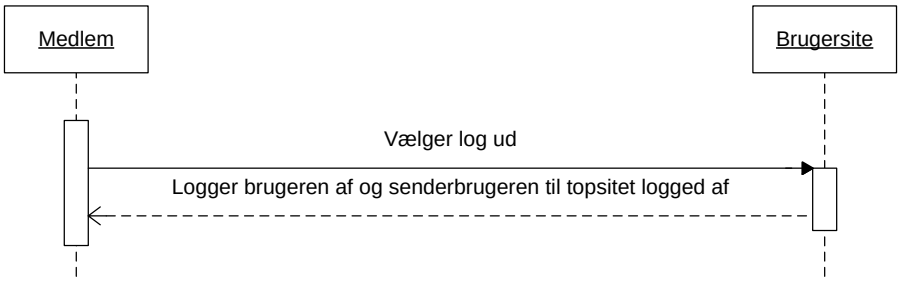
5.1.8 Medlem – Skriv anmeldelse



Figur 10 - Medlem skriv anmeldelse

En bruger kan skrive en anmeldelse af en venue. Brugeren skal blot være logget på og være på venuesitet. Når anmeldelsen er skrevet og der trykkes send, gemmer venuesitet anmeldelsen og opdatere siden, således brugeren ryger tilbage til venuesitet igen.

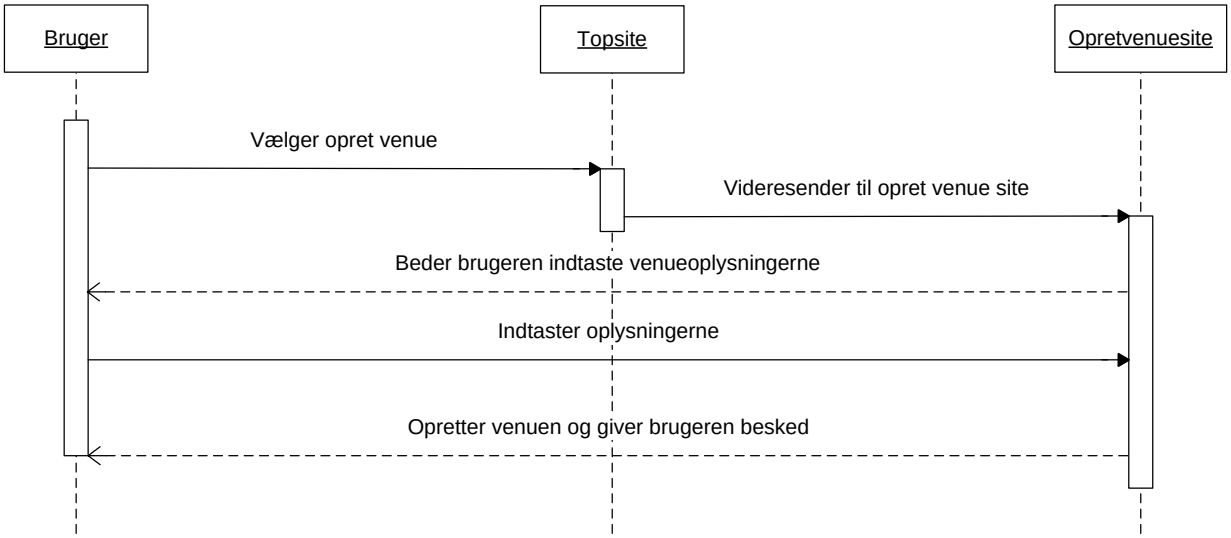
5.1.9 Medlem – Log ud



Figur 11 - Medlem log ud

Når en bruger er logget på, er der en log af mulighed i stedet for en log på knap. Når brugeren trykker på denne, vil brugersitet logge brugeren af og sende ham/hun til topsitet.

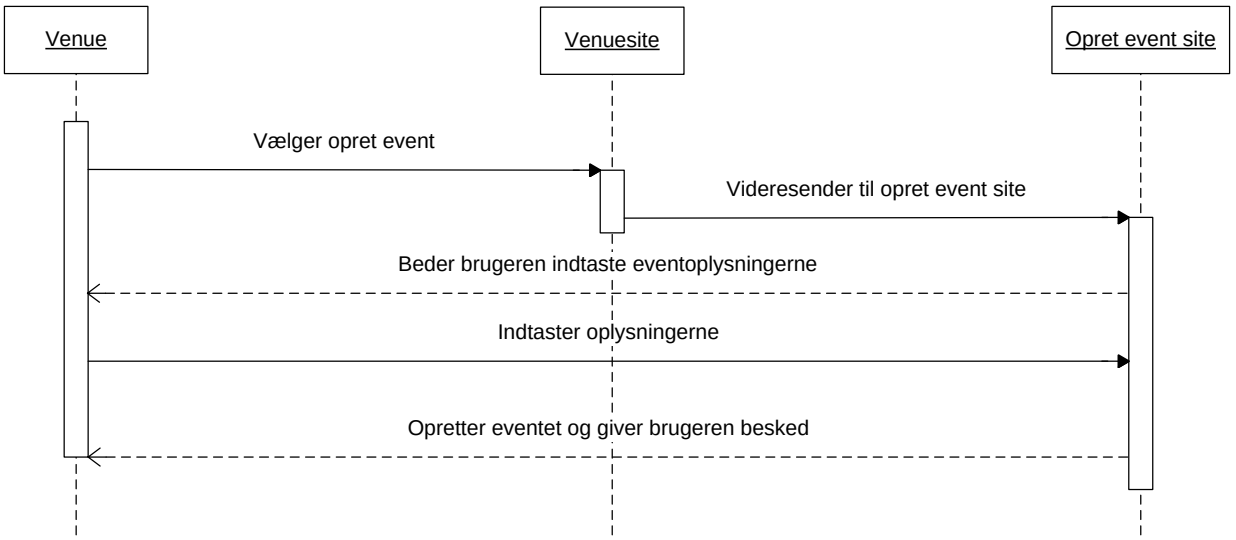
5.1.10 Medlem – Opret venue



Figur 12 - Medlem opret venue

Et medlem der er logget på systemet kan oprette en venue. Når opret venue er blevet valgt sender topsitet brugeren til en opret venuesite, denne site beder brugeren indtaste de nødvendige venueoplysninger. Når dette er gjort gemmes de og brugeren får besked om at venuen er blevet oprettet.

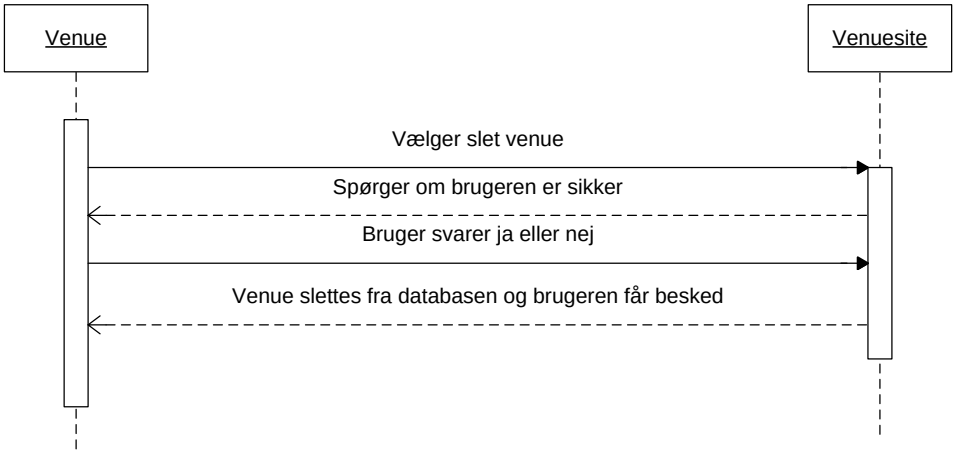
5.1.11 Venue – Opret event



Figur 13 - Venue opret event

En bruger der har oprettet en venue, kan via venuesitet oprette et event. Når brugeren har valgt "opret event", skal han efterfølgende udfylde en formular. Når dette er gjort og brugeren gemmer de indtastede oplysninger, sørger venuesitet for, at oprette eventet og give brugeren besked.

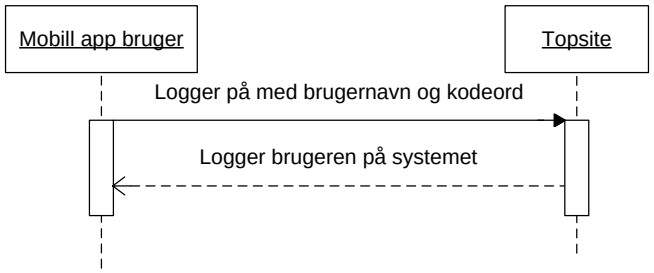
5.1.12 Venue – Slet venue



Figur 14 - Venue slet venue

En bruger der vil slette sit venue, bliver spurgt om han/hun er helt sikker. Når brugeren har svaret ja slettes venuen fra databasen og brugeren får besked herom. Årsagen til at brugeren skal svare ja, er for at man ikke sletter ved en fejl. En anden løsning er at man kan lukke sin venue og derefter har muligheden for at genåbne den. Det vil sige den stadig fjernes fra siden, men brugeren har altså mulighed for at kunne fortryde eller genoprette den igen.

5.1.13 Mobil app bruger – Log på

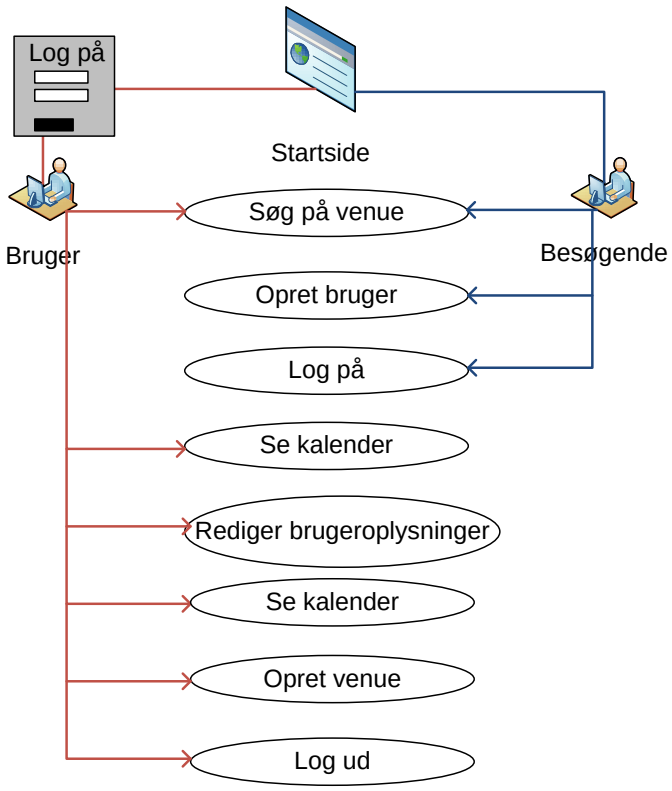


Figur 15 - Mobil app bruger log på

En mobil app bruger logger på med sit brugernavn og kodeord. Informationerne skal godkendes før brugeren logges på systemet.

5.2 Navigationsdiagram

Figur 16 viser et navigationsdiagram over mainsitet.



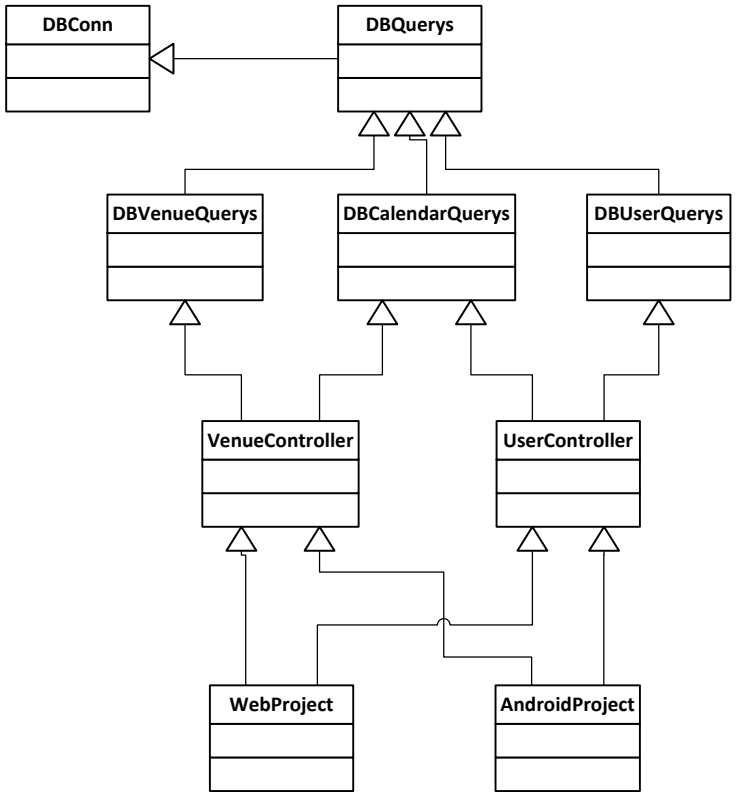
Figur 16 - Navigationsdiagram

Det ses på navigationsdiagrammet hvordan en besøgende eller en brugeren navigere rundt på mainsitet. En besøgende der logger på er en "bruger", det betyder han/hun har et andet view end den besøgende. Det ses bl.a. hvordan en bruger ikke kan oprette en ny bruger hvis personen er logget på i forvejen, derudover har brugeren også muligheden for at logge af i stedet for log på.

5.3 Systemoversigt

5.3.1 UML Klassediagram

For at bedre kunne illustrere systemet, blev der oprettet et UML diagram (Unified Modelling Language). Diagrammet viser en grov model af systemets klasser og relationer.



Figur 17 - UML diagram over systemet

For at kunne implementere use casene, var disse klasser nødvendige. Diagrammet er en grov model og viser ikke alle klasserne i systemet. DBConn er database-connecter klassen, denne kaldes af DBQuerys. DBQuerys bliver benyttet af de tre klasser DBvenueQuerys, DBCAlenderQuerys og DBUserQUerys. Disse 5 nævnte klasser, står for alt kommunikationen mellem databasen og resten af systemet. Der vil i implementeringen være en detaljeret forklaring af klasserne og en figur der indeholder deres attributter og metoder.

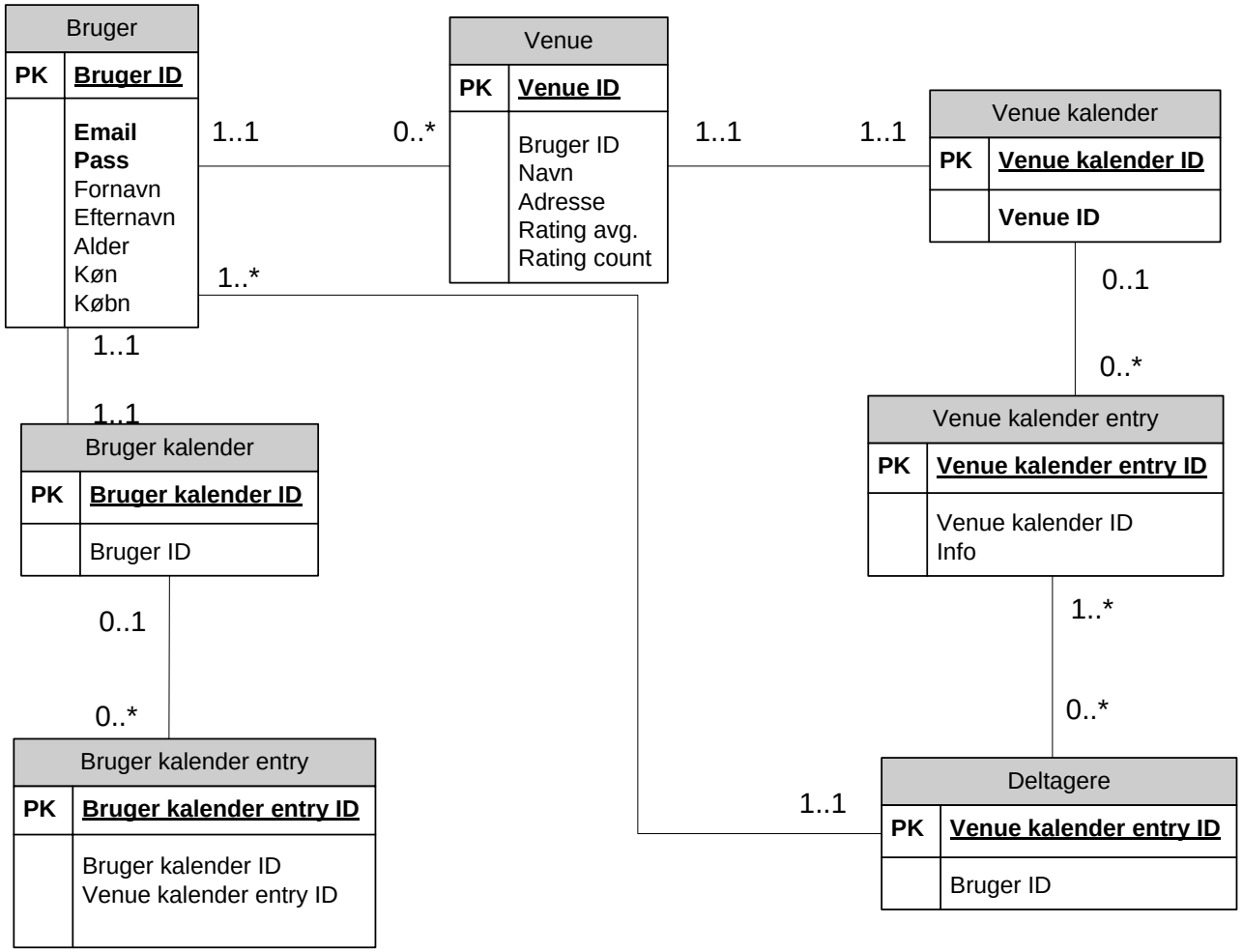
VenueController og UserController klasserne skal styrer kommunikationen, imellem web projektet og androidprojektet.

WebProjekt klassen repræsenterer alt hvad der har med hjemmesiden at gøre. Ligeledes repræsenterer AndroidProject klassen alt hvad der har med mobilapplikationen at gøre.

5.3.2 Database entiteter

For at kunne implementere use casene, er det nødvendigt med en database. De fleste af informationerne i databasen, vil stamme fra brugerne. Det betyder at der ofte vil blive lagret ændret og slettet i tabellerne.

Der skal eksistere tabeller indeholdende brugere, venues, kalendere og anmeldelser. Nedenfor ses de tabeller der skal implementeres, samt en beskrivelse.



Figur 18 - Database oversigt

5.3.2.1 Bruger tabel

Denne tabel indeholder alle informationer der er om brugeren. For at holde styr på disse, tildeles hver bruger et unikt ID nummer som primary key. De attributter der er markeret med fed skrift, er informationer der skal udfyldes. Dvs. for at oprette en bruger skal der indtastes Email og Pass.

5.3.2.2 Venue

Denne tabel indeholder alle informationer der er om en venue. Ligesom brugere tildeles venues en Venue ID som primary key. Derudover har det en bruger ID som ejer.

5.3.2.3 Venuekalender

Hver venue har en kalender, denne tabel indeholder denne kalender. Hver kalender har et Kalender ID som primary key og Venue ID for at holde styr på hvilke venue kalenderen tilhører.

5.3.2.4 Bruger kalender

Ligesom venue kalenderen, har en bruger sin egen kalender. Bruger kalender har et bruger kalender ID som primary key og indeholder derudover et bruger ID, som refererer til den bruger, der ejer denne kalender.

5.3.2.5 Venue kalender entry

Denne tabel indeholder venue events. Primary key er Venue kalender entry ID, venue kalender ID holder styr på hvilken kalender eventet tilhørere og info indeholder event information.

5.3.3 Deltagere

Denne tabel indeholder de brugere der skal deltage til eventet. Primary key er Venue kalender entry ID. User ID indeholder bruger ID, på dem som deltager til eventet.

5.3.3.1 Bruger kalender entry

Denne tabel indeholder de events den specifikke bruger er tilmeldt. Bruger kalender entry ID er primary key. Bruger kalender ID bruges til at se hvilken bruger, kalenderen tilhører og Venue kalender entry ID benyttes til at se hvilket event der er tale om.

5.4 Layout

Hjemmesiden skal have et layout der gør den behagelig at bruge og se på. Det er meget vigtigt at det endelig produkt har et gennemført design og layout der passer til. I dette projekt er det ikke blevet prioriteret da vi anser funktionaliteten for den ”tunge del” og layout som ”lettere”. Dette er ikke ens betydende med at det er let at lave et flot layout.

For at kunne give en ide om hvad siden går ud på, laves der en forside, som kan bruges, eller benyttes som ide, til en fremtidig udvikling.

5.4.1 Navngivning og domænenavn

Lige siden projektets begyndelse havde vi overvejet et navn man evt. kunne give projektet. Vi tænkte på en URL der hed ”www.iPartyAt.dk” og har brugt det navn undervejs i implementeringen. Vi fandt dog senere ud af at iPartyAt blev brugt til noget andet online. Vi kom derefter på navnet ”www.NightRaters.com” og syntes det havde en god dobbeltbetydning. Projektet vil derfor i fremtiden navngives NightRaters.

5.4.2 Navigation af hjemmeside

Det er vigtigt at siden er organiseret og struktureret. Det betyder at ”tingene” skal være hvor de forventes at være. Der er mange forskellige måde at løse dette, fx ved at have en menu langs en af siderne, eller i toppen og bunden, eller en ramme som forbliver hvor den er på alle siderne.

Vi forestiller os ”ramme-løsningen” som den mest optimale, da det også er den vi selv oplever mest behagelig. Derudover er det let at gøre hele siden tilgængelig hvor end man befinder sig. Der skal være en menu bar i toppen, hvorpå brugerne kan se hvad siden omhandler. Kanterne skal være tomme og så skal sidekonteksten vises i midten.

5.4.3 CSS

Til hjemmesiden skal der laves et Cascading Style Sheet (CSS) som gælder for alle undersiderne. Da dette er en ”Nice-to-have” krav, skal det kun gøres til forsiden. Men derefter er det relativt let at kopiere til de resterende sider.

Temaet skal være festligt, men skal på en måde også virke lidt afslappet. Det kan være svært at finde et tema der opfylder dette, men vi har bestemt os for at bruge en lilla farve, med noget sort gradient i kanterne.

5.4.4 Skærm Opløsning

Layoutet skal primært designes til skærm opløsning 1024x768 og højere. Fra w3schools[1] statistik ses det at kun 1,1 % benytter anden opløsning end 1024x768 og højere.

5.4.5 Mobil applikation GUI

Til mobilapplikationen er kravene at der vises nogle knapper, hvilke der kan trykkes på. Der skal heller ikke her sættes særlig meget tid af til layout, men funktionalitet.

5.5 Delkonklusion

Designet af løsningen er udført vha. Sekvensdiagrammer og UML diagrammer. På baggrund af analysen og designet kan implementeringen udføres.

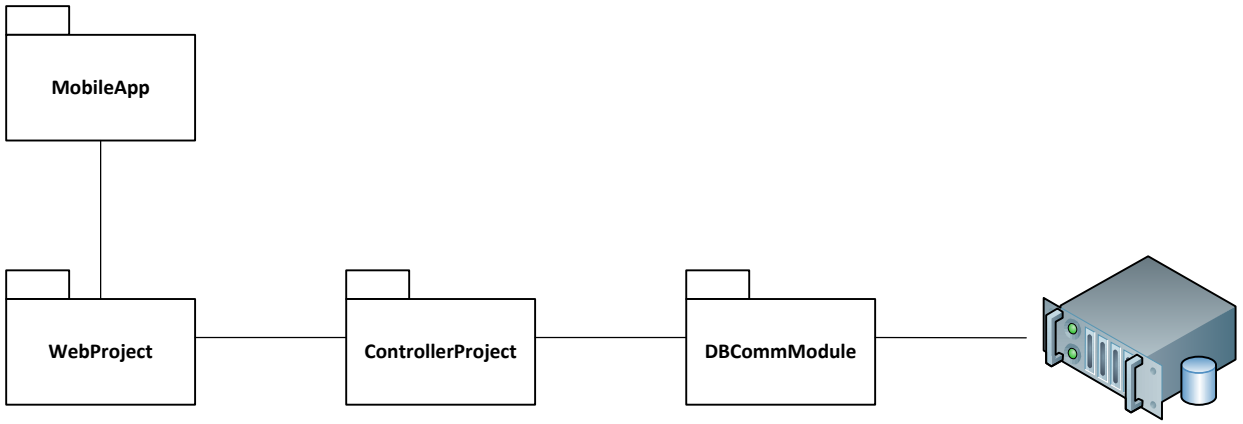
Undervejs i implementeringen er der blevet ændret i designet. Dette er som regel blevet gjort når der blev fundet en mere passende løsning. Database modellen og Klassediagram er resultater heraf og illustrere systemet som det endeligt ser ud.

[1] http://www.w3schools.com/browsers/browsers_display.asp

6 Implementering

Dette afsnit vil omhandle selve implementeringsfasen af projektet. Først vil værktøjerne som er brugt, blive gennemgået. Herefter vil de 6 projekter, som projektet er delt op i, blive gennemgået. Disse vil blive gennemgået hver for sig og under webprojektet (*iPartyAt*) og mobilprojektet (*MobiPartyAt*) vil indelingen være de forskellige use cases som er blevet defineret tidligere.

En fuld oversigt over kommunikationen mellem alle de forskellige projekter kan ses på figuren.



Figur 19 - Diagram over projekterne i systemet

Her ses kommunikationen mellem projekterne. Mobilprojektet snakker samme med Webprojektet, som bruger ControllerProjektet til at sende og modtage informationer til og fra databasen via DBCommModule.

6.1 Valg af værktøjer

Der er, til udvikling af projektet, brugt nogle forskellige værktøjer. Disse værktøjer vil blive gennemgået og der vil blive redegjort for, hvorfor valget er landet på det pågældende værktøj.

6.1.1 Java Server Pages (jsp)

Til at opbygge hjemmesiden har vi valgt at bruge Java Server Pages (Jsp). Dette har vi valgt fordi vi så en fordel i, at jsp gjorde det muligt for os at skrive vores server scripts i Java, hvilket er et sprog vi kender godt. At udvikle i jsp gør det også muligt for applikationen at køre på forskellige platforme, hvor f.eks. Active Server Pages (Asp) er begrænset til Microsoft platforme. Effektiviteten på Jsp spiller også en vigtig rolle for dette valg, og dens error-handling gør systemet mere stabilt. Muligheden for at bruge tags er endnu et plus der bør nævnes. Her valgte vi at bruge JavaServer Pages Standard Tag Library (JSTL). Ved at bruge dette bliver opbygningen af jsp filerne meget mere overskuelige, da meget af server script funktionaliteten bliver skrevet ved hjælp af JSTL og det på denne måde ligner helt almindeligt HTML. Det er også muligt at skrive sine egne tag libraries og derved opnå en ønsket funktionalitet, der let kan bruges andre steder i applikationen eller overføres til fremtidige applikationer. Ved at bruge JSTL, kan vi også få mest muligt ud af at bruge Expression Language (EL). EL er en let måde at tilgå data i applikationen på og en hurtig måde at lave branches og sammenligninger osv. på. En lille ulempe ved jsp, er at mange webhoteller ikke understøtter dette. Der findes dog så mange webhoteller i dag at dette ikke er noget egentligt problem.

6.1.2 Dojo

Til vores applikation valgte vi at bruge et Javascript bibliotek. Det blev hurtigt klart at valget stod mellem jQuery og Dojo. Disse tilbyder stort set det samme hvor jQuery kan en smule mere end Dojo. Dog opnås dette primært ved hjælp af forskellige plug-ins hvorimod Dojo har det meste i sig fra start. Dojo er blevet valgt af IBM som standard i deres Websphere Application Server, og vi tror derfor at Dojo kan komme til at blive førende på dette punkt og der vil derfor være fordel i at kende noget til brugen af dette. Det primære vi vil bruge Dojo til er Ajax (Asynchronous JavaScript and XML) kald og event-handling samt input validering.

6.1.3 Server

Til at køre applikationen, er der brug for to servere. En til selve hjemmesiden samt en til databasen. Til hjemmesiden valgte vi en Apache Tomcat server. Dette var af den simple grund at den er lavet til at køre jsp og så er den let og simpel at sætte den op, let at konfigurere og hurtig til at teste vores system. Det var vigtigt for os at det skulle være v.7.0+ da det først er fra denne version at EL er understøttet.

Til database serveren valgte vi at bruge noget software, der hedder Mowes. Dette software gør det let og hurtigt at sætte et system op med en Apache server, MySQL database samt PHP til at administrere databasen.

6.1.4 Android

Valget til platformen hvorpå mobil applikation skulle køre, faldt også ret hurtigt på Android. Det stod hurtigt klart at valget ville ligge mellem Android fra Google eller iOS fra Apple. Disse to er de største for tiden på smartphone markedet og senest har Android overgået iPhone i salg på det amerikanske marked.[2] Disse tal passer ikke helt til Europa, hvor vi godt kan gå ud fra at Symbian har den største andel, med iOS i hælene. Android er dog ved at komme godt med og vi tror på, at i den nærmeste fremtid vil Android komme til at ligge i toppen.

Udvikling til Android er hurtigt og let at komme i gang med. Hvis vi skulle udvikle til iPhone skal man først have en Mac samt registreres som Apple udvikler. Det er selvfølgelig også en fordel at eje en iPhone.

Da vi begge ejer en Android telefon og ingen af os ejer en Mac var valget hurtigt truffet.

Da Android udvikles i Java ville det falde naturligt ind i vores projekt og vi ville let kunne bruge datastrukturen og andre værktøjer fra web projektet uden at skulle ændre noget i disse.

6.1.5 Eclipse

At bruge Eclipse til at udvikle hele projektet i, var også naturligt. Dette værktøj gør det let at konfigurere vores Tomcat. Eclipse er lavet i Java og lavet til udvikling i Java samt andre udviklingssprog via plug-ins.

Der findes et plug-in til Eclipse, til udvikling af Android, hvilket er meget let at installere og som indeholder nogle rigtig gode værktøjer til udvikling samt debuging og test, både på emulator men også på en fysisk telefon.

2 http://buzzintechnology.com/2011/04/android-tops-ios-to-become-the-most-desired-smartphone-os/

6.2 Afgrænsning af implementeringen

Som udgangspunkt skal alle use casene implementeres. Der vil undervejs i implementeringen, blive gjort op, om der skal skæres nogle krav fra, således tidsplanen overholdes. Vi har dog valgt at ændre "Søg på venues" til en browse funktion, der lister venues'ne op. Da en søge funktion ville kræve for meget tid at lave. "Need to have" kravene prioriteres i første omgang og hvis der er tid, implementeres "Nice-to have" kravene.

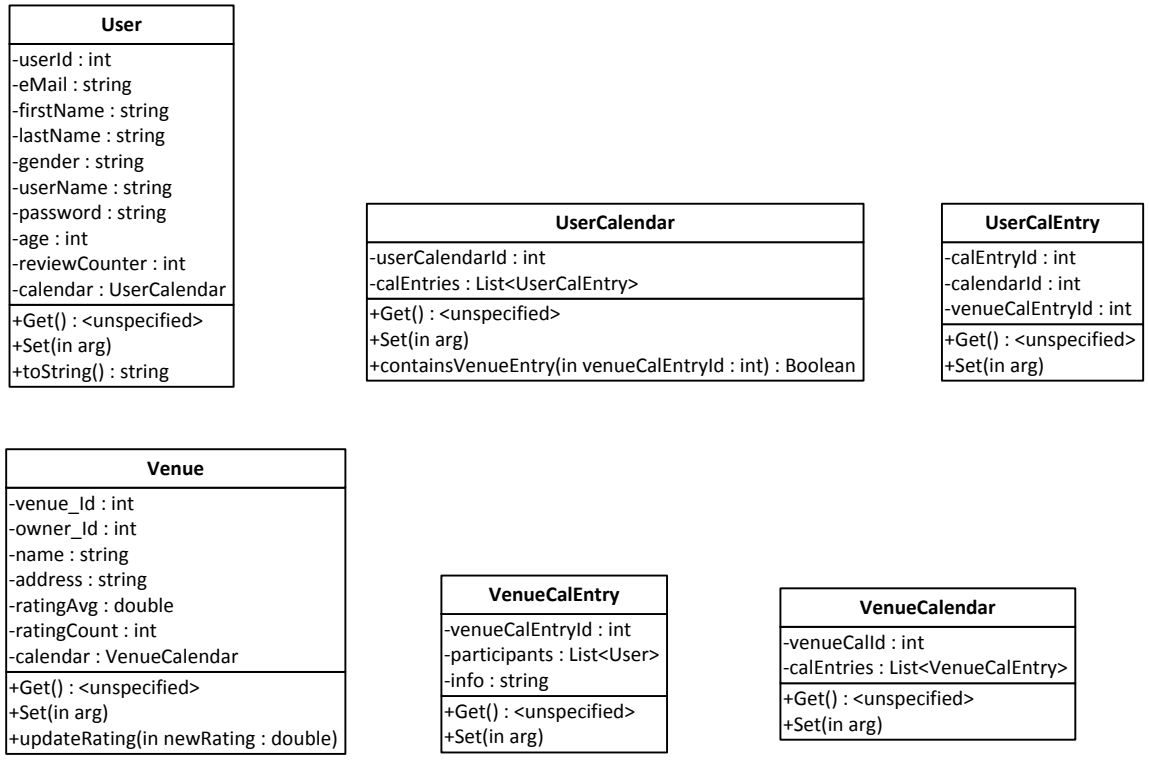
6.3 DataProject

Til udvikling af projektet ville vi få brug for nogle veldefinerede datastrukturer. Hver af disse datastrukturer repræsenterer en række i en databasetabel, dog med nogle enkelte afvigelser. En af disse afvigelser ses i VenueCalEntry (se *fig. 20*), hvor participants findes som en liste af brugere. Denne liste bliver genereret ud fra databasetabellen *participants*.

Den anden afvigelse er i UserCalendar. Her findes en metode der hedder *containsVenueEntry(in venueCalEntryId: int)* som returnerer en boolean. Denne metode gennemgår *calEntries*-listen og returnerer "true" hvis der findes en UserCalEntry med VenueCalEntryId, som er lig med venueCalEntryId som er sendt med som parameter til metoden.

Den tredje, og sidste, af disse afvigelser ses i Venue klassen. Her findes en updateRating metode som tager en ny rating værdi (1-5) og lægger den til det nuværende gennemsnit, hvorefter tælleren *ratingCount*, som holder styr på hvor mange gange en venue er blevet rated, inkrementeres med 1.

Datastrukturerne findes i et selvstændigt Java projekt der hedder DataProject. De er placeret her således at de simpelt kan refereres til fra de andre projekter som alle bruger disse datastrukturer.



Figur 20 - Datastruktur

Fig. 20 Viser de forskellige datastrukturer der bliver brugt i projektet. Get() og Set(in arg) metoder refererer til at alle attributter i klassen har en getter og en setter metode.

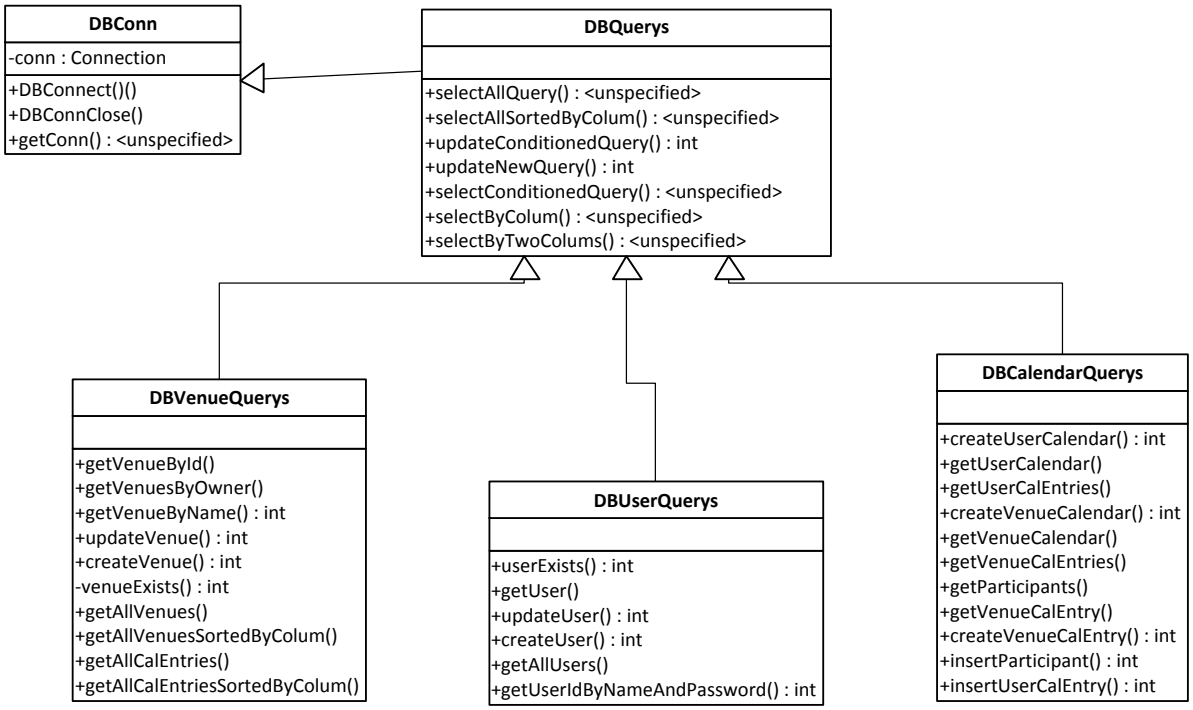
6.4 DBCommModule

Dette Java projekt indeholder kommunikationen mellem webserver og databaseserver. Dette sker via klassen *DBConn.java*. Forbindelsen bliver oprettet når en klasse kalder *getConn()* metoden. Denne metode finder først ud af om der er en forbindelse åben og returnere denne, hvis den findes. Ellers kaldes *DBConnect* metoden og en forbindelse åbnes, hvorefter denne returneres. Klassen indeholder derudover en metode til at lukke en eventuel åben forbindelse. Selve forbindelsen er en *Connection*-klasse som findes i den eksterne jar "*my-sql-connector-java-5.1.15-bin.jar*". Denne jar indeholder alt der skal bruges til at oprette og vedligeholde en forbindelse samt sende og modtage informationer fra en database, der kører MySQL.

Klassen *DBQuerys.java* står for at konstruere og eksekvere SQL sætninger og returnere data, der bliver modtaget fra databasen. I denne findes forskellige metoder til at oprette en SQL sætning, som kan kaldes fra andre klasser. Når en SQL sætning er genereret hentes databaseforbindelsen fra *DBConn.java*, hvorefter sætningen eksekveres. Er det en sætning der indsætter noget i databasen bliver en *int* returneret indeholdende 0 eller 1. 1 betyder at 1 række er blevet sat ind i databasen. 0 betyder at ingen rækker er blevet indsat. Er det derimod en sætning der henter data fra databasen bliver et *ResultSet* returneret indeholdende de data der er blevet hentet.

Yderligere findes der 3 klasser i dette projekt: *DBCalendarQuerys.java, DBUserQuerys.java* og

DBVenueQuerys.java. Disse klasser indeholder metoder der samler de korrekte informationer, som skal bruges af *DBQuerys.java* til at opbygge en SQL-sætning. Som deres navne antyder findes der en klasse, der omhandler kalendere, en der omhandler brugere samt en der omhandler venues. Det er også disse klasser der læser informationen hentet fra databasen og opretter den korrekte datastruktur og sender denne tilbage i systemet. Da informationer, som sendes tilbage fra databasen, ligger i en datastruktur der hedder *ResultSet* er det simpelt at finde de korrekte informationer. Denne datastruktur findes fra den samme MySQL-connector som er nævnt ovenfor. Når der findes informationer i denne datastruktur ligger de i en tabel som de ville gøre hvis man prøver at køre SQL-sætningen direkte i en kommandolinje tilhørende databasen. Ved at bruge metoden *next()* flytter man markøren til næste række i denne tabel. Er det første gang metoden kaldes flyttes markøren til den første række i tabellen. Når markøren er på en række kan informationen fra et bestemt felt, fra denne række, hentes ved at kalde getter metoder med kolonnenavnet som argument. Når typen af informationen, der findes i denne kolonne kendes, vides også hvilken getter der skal bruges (f.eks. *getInt()* eller *getString()* osv.).



Figur 21 - Klasser i DBCommModule

Fig. 21 Viser et diagram over hvordan klasserne i dette projekt snakker sammen. De tre klasser DBVenueQuerys, DBUserQuerys og DBCalendarQuerys sender information til DBQuerys som bruger fobindelsen conn fra DBConn klassen til at forbinde til databasen.

6.5 ControllerProject

Dette projekt styrer kommunikationen mellem frontend og backend. Hvis frontenden skal hente informationer fra databasen eller indsætte information i databasen, sker det gennem dette projekt.

Der findes 2 klasser i projektet. En til at styre informationer der sendes til og modtages fra databasen om brugere: *UserController.java*, og en til at styre informationer der sendes og modtages fra databasen om venues: *VenueController.java*.

6.6 Servers

Dette projekt indeholder alle konfigurationsfiler for Tomcat serveren. Dette projekt indeholder 6 filer hvoraf et par af dem er blevet konfigureret for at leve op til kravene til hele opgaven.

Når en bruger opretter sig, logger ind eller kigger rundt på siden mens man er logget ind, skal klienten og serveren kommunikere via en Secure Sockets Layer (SSL) forbindelse. For at opnå dette skal serveren sættes op til det.

For det første skal serveren bruge et certificate. Dette kan laves lokalt eller man kan få et fra firmaer. Et firma der uddeler certifikater er VeriSign.com og her koster den billigste $399 for et år[3]. Derfor har vi valgt at gøre det lokalt. Dette kan gøre ved at bruge et værktøj der kommer med Java: *keytool* som ligger i bin mappen i jde'en. For at oprette en keystore indeholdende et nyt certifikat bruges kommandoen på følgende måde:

%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA

Et password vælges samt navn på ejer og organisation. Dette opretter en keystore i brugerens egen mappe. I XP vil denne ligge under Documents and Settings og i Vista og Win7 vil denne ligge i mappen Users. Herefter skal der ind kommenteres en entry i server.xml filen. Denne entry er en SSL connector til port 8443. Denne connector peger på den nyligt lavede keystore med dertilhørende password.

Til sidst skal serveren sættes op til hvornår der skal bruges SSL. Dette gøres i filen web.xml. Følgende sættes ind sidst i denne fil:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Protected Context</web-resource-name>
    <url-pattern>/UserSection/*</url-pattern>
  </web-resource-collection>
  <!-- auth-constraint goes here if you requre authentication -->
  <user-data-constraint>
     <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

Dette definerer, at når en jsp fra mappen UserSection bliver hentet, går forbindelsen over på en Https forbindelse.

3 https://www.verisign.com/ssl/buy-ssl-certificates/secure-site-ssl-certificates/index.html

6.7 iPartyAt Webprojekt

Dette projekt er selve hjemmesideprojektet. Dette afnsit er delt op således at implementeringen af hver use case bliver forklaret.

Fælles for alle jsp'er er at de inkluderer filen der hedder EnvironmentSetup.jspf. En .jspf fil er et fragment af en jsp fil som bliver statisk inkluderet i jsp-filen. Dette betyder at vi kan gemme variabler og andre ting heri og som kan læses af alle jsp'er, der inkluderer denne fil. I EnvironmentSetup gemmes om en besøgende på siden er af typen 'R' eller af typen 'G'. R (registered) betyder at den besøgende er en registreret bruger som er logget ind og G (guest) betyder at den besøgende ikke er logget ind og derfor betragtes som en gæst.

6.7.1 Implementering af use case – Søgning på alle venues og events

Selve søgningen blev taget ud som krav og erstattet med en browse-funktion. Der er blevet oprettet en jsp, som tager sig af disse to funktioner. Denne hedder *BrowsePage.jsp* og findes i mappen *SearchSection*.

For at denne side skal vise noget som helst, skal den kaldes med en parameter der hedder *browseType*. Denne parameter kan have to værdier: ”venues” eller ”events”, og definerer hvad der browses på. Derudover tager klassen en parameter der hedder *sortBy*, som ikke er krævet at være udfyldt. Hvis denne ikke er udfyldt vil listerne blive hentet som de er indsat i databasen. Disse parametre kan læses ud fra den ”request” der er blevet modtaget. Denne ”request” er af typen *HttpServletRequest*, og indeholder en metode: *getParameter(String parameterName)*, til at hente en parameter med givne navn.

Hvad der sker gennem systemet er nogenlunde det samme når man kigger på venues som når man kigger på events, og derfor vil kun den ene blive gennemgået her. Venues er en smule mere omfattende og derfor vil det være den der gennemgås.

Det første der sker, er at en VenueController bliver oprettet og henter alle venues/events. Hvis *sortBy*-parameteren er tom, hentes alle venues/events ved at bruge metoden *getAllVenues()* fra VenueController-klassen. Er *sortBy* derimod ikke tom bliver alle venues/events hentet ved at bruge metoden *getAllVenuesSortedByColum(String columName)*. For at finde ud af hvilken kolonne i databasen, der skal sorteres efter, har hver kolonne på siden (Name, Address, Rating og Number of ratings) et id der svarer til navnet på kolonnen i databasen; f.eks.: Name kolonnen på siden har id=”name” og Rating kolonnen på siden har id=”ratingAvg”. På denne måde sendes id'et på kolonnen, der er trykket på, videre som kolonnenavn, og SQL-sætningen i *DBCommModule*-projektet står for at hente et sorteret set af informationer ud fra databasen og sende det tilbage.

Når informationen er kommet tilbage, skal de skrives ud på siden. Da de ligger i et *ResultSet* bruges denne *next()*-metode til at iterere hen over informationerne og skrive dem ud på siden.

Hvis der skal sorteres efter ”rating” eller efter ”number of ratings” er sættet der kommer fra databasen, sorteret efter mindste tal først. Da det er ønskeligt at den rating med højest rating ligger først, køres sættet igennem fra sidst til først. Ved at bruge metoden *last()* fra *ResultSet*, bliver markøren sat frem til den sidste række i sættet. Denne skrives ud og resten af sættet kan itereres i modsat retning ved at bruge metoden *previous()*. Dette gøres kun i forbindelse med venues da events ikke har nogen kolonne der indeholder tal.

Alt den html der bliver genereret i disse løkker, bliver gemt i en String. Når alt til sidst er genereret bliver denne String skrevet til klienten via *out*, som er en jspWriter.

6.7.2 Implementering af use case – Log ind

Den grafiske brugerflade til at logge ind findes i *UserSection/LoginPage.jsp*. Denne klasse indeholder en form hvori der findes tre inputs: en text, et password og en submit. Via JavaScript er der knyttet et event til denne submit. JavaScriptet til denne side ligger i filen *js/Login.js*. Dojo har en metode, der hedder *addOnLoad()*. Denne metode kører når siden er loaded og det er her eventhandleren bliver kædet sammen med, tryk på submit knappen. Dette sker ved at køre Dojo's *query()*-metode. Som argument tager denne metode en streng indeholdende en CSS selector. Dvs. at man i denne metode finder frem til elementet i html'en på sammen måde som man ville gøre det i en CSS fil. Metoden returnerer en liste af DOM Nodes. I dette tilfælde har submitknappen en attribut der hedder ”class='submit'” og elementet kan derfor findes og en eventlistener kan tilknyttes via Dojo, på følgende måde:

```
dojo.query(".submit").connect("onclick", function(){ .. });
```

Connect sætter en event på hver enkelt DOM node i denne liste. I dette tilfælde ved vi at der kun er en node og det er derfor ikke nødvendigt at finde frem til den korrekte node. Denne metode tager to argumenter. Hvilket event der skal lyttes på, samt en funktion der definerer, hvad der skal ske når denne event sker.

Når der klikkes på knappen sker der først noget validering. Denne validering vil blive forklaret senere og er derfor udeladt her. Efter valideringen er gennemført uden fejl bliver et Ajax kald sat op. Dette gøres ved at bruge en af Dojo's xhr metoder. Der findes to af disse metoder: *xhrGet* og *xhrPost*. De bruger hver deres submission metode, Get og Post hhv. I dette tilfælde bruges *xhrPost(..)*. Dette er for at undgå at få alle informationerne der skal bruges til at ligge i URL'en. Koden ser ud som følgende:

```
dojo.xhrPost({
 url: "../tools/Login.jsp",
 handleAs: "text",
 content: {
  userName: dojo.byId("userName").value,
  password: dojo.byId("password").value
 },
 handle: function(data){
  data = parseInt(data.replace(/^\s+|\s+$/g, '')) ;
  if(data >= 1){
   window.location = "UserPage.jsp";
  }
  else if(data == -2){
   setErrorText("Something went wrong while processing your request. Please try
                 again later.");
  }
  else if(data == -6){
   setErrorText("Username and/or password incorrect.");
  }
 }
});
```

Denne metode tager et map som argument. Dette map indeholder en url til det hvorfra Ajax kaldet skal få den information der skal bruges. I dette tilfælde er det *Login.jsp*, som vil blive gennemgået nedenfor. HandleAs definerer hvordan dataen der kommer tilbage fra serveren skal fortolkes. Herudover sendes et map med, der hedder content. Dette map indeholder username og password, som hentes fra de to sidste input elementer. Til sidst findes handle som er en funktion der styrer hvad der skal ske når dataen kommer

tilbage fra serveren. Det allerførste der sker her er at dataen strippes for whitespaces. Herefter tjekkes der på hvilket tal der er sendt tilbage fra serveren.

Tabellen viser hvad disse talkoder kan være og hvad de betyder. Hvis loginkaldet fejler af en eller anden grund skrives en fejltekst ud på siden. Lykkedes det at logge ind bliver brugeren sendt videre til *UserPage.jsp*.

Talkode	Betydning
1	Login er successful
-2	Noget gik galt under login på serversiden
-6	En bruger med dette brugernavn/password kombination findes ikke i systemet.

Tabel 4 - Talkoder for login

Tabel 2 viser hvilke svar et loginkald kan få tilbage fra serveren samt deres betydning.

6.7.2.1 Login.jsp

Denne fil findes i tools-mappen og bliver kaldt når en bruger skal logges ind.

Det første der sker, er at parametrene, som er sendt med, hentes ud via *getParameter()*-metoden. Det er samme metode at hente parametre på, uanset om Post eller Get er brugt. Herefter bliver en *UserController* oprettet som så spørger databasen om en bruger med givne username og password findes. Her modtages et tal som er enten -6, -2 (se tabel 2) eller brugerens userID. Er svaret ikke lig med eller over 1, bliver svaret sendt tilbage. Hvis det er et tal lig med eller større end 1 betyder det at brugeren findes i databasen og denne brugers userID er sendt tilbage. Dette ID bruges til endnu et kald til databasen. Denne gang skal brugeren hentes frem og gemmes i den nuværende session. Variablen *userType* bliver også opdateret så denne nu læser at det er en registreret bruger der er logget ind. Til sidst bliver der sendt et 1-tal tilbage så handle-funktionen i Ajax-kaldet ved at brugeren er logget ind. Dette gøres ved at skrive til objektet *response*'s PrintWriter. *Response* er et objekt af typen HttpServletResponse.

6.7.3 Implementering af use case – Log ud

Til dette formål findes en lille klasse i tools mappen der hedder *Logout.jsp*. Denne klasse indeholder en enkelt linje kode som invaliderer den kørende session. Dette nulstiller den og brugeren er dermed logget ud.

6.7.4 Implementering af use case – Opret bruger

GUI'en til oprettelse af ny bruger findes i *UserSection/SignUpPage.jsp*. Lige som login, findes der her en form indeholdende flere inputelementer. Her findes en til nyt brugernavn, et til brugerens email, et til fornavn samt et til efternavn. Derudover findes en til password samt en til gentagelse af password. Til sidst findes 4 drop downs. En til valg af køn samt 3 til valg af fødselsdato, fødselsmåned og fødselsår hhv. Det er muligt at vælge fødselsdag fra 1 til 31 og måned 1 til 12 og år 1930 til nu. Der er dog ikke implementeret validering eller lagring af dataen. Så det kan ikke bruges til noget endnu.

Javascripten til denne funktionalitet minder rigtig meget om den der findes til at logge ind. En forskel er dog at her sendes hele formen videre vha. *xhrPost,* til forskel fra login hvor der var et content-map. Så er der selvfølgelig også forskel på URL parameteren. Resten af Javascripten er stort set den samme og vil ikke blive gennemgået yderligere.

Her findes også nogle returneringskoder som er listet i tabellen Herunder.

Talkode	Betydning
1	Brugeren er oprettet uden fejl
-1	Der findes allerede en bruger med denne email i databasen
-2	Der er sket en fejl under indsætning af ny bruger.

Tabel 5 - Talkoder over opretbruger

Tabel 3 viser hvilke svar opret bruger kan få tilbage fra serveren samt deres betydning.

6.7.4.1 CreateUser.jsp

Denne klasse befinder sig i tools og er en ret simpel klasse. Det der sker i denne, er at først blive alle de relevante parametre gemt i variable. Herefter oprettes en *User*-datatype, som er en datatype fra *DataProject*, som er forklaret ovenfor, med de nyoprettede variable. En *UserController* oprettes og metoden herfra kaldet *createUser()* kaldes med det oprettede User objekt som argument. Denne metode bruger en metode med samme navn i *DBUserQuerys*, som validerer om der findes en bruger med samme email i databasen. Dette sker via en metode i samme klasse der hedder *userExists*. Denne metode returnerer 0 hvis brugeren ikke findes, -1 hvis brugeren findes og -2 hvis noget er gået galt under kaldet. Findes brugeren ikke i databasen, opretter *DBUserQuerys,* SQL-sætningen og sender den videre. Herefter oprettes en kalender til denne bruger. Tilbage i *CreateUser.jsp* bliver svaret skrevet i klassen *response*.

6.7.5 Implementering af use case – Se kalender

Denne use case er blevet ændret en smule så man på samme tid som man ser kalenderen, kan man også se alle sine personlige brugerinformationer samt brugerens kalender og alle de venues som denne bruger har oprettet.

GUI'en til denne use case findes i filen *UserSection/UserPage.jsp*. I denne klasse bruges der noget JSTL. Der bruges det bibliotek der hedder ”core” og hentes på følgende måde:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

Dette fortæller siden at vi bruger et taglib der findes på adressen defineret via parameteren *uri*. Parameteren *prefix* definerer hvad tagget skal starte med for at siden ved det er dette tag bibliotek der skal kigges i. I dette tilfælde er denne ”c” og et eksempel på et kald kunne være:

```
<c:when test="${ empty user.calendar.calEntries }">
```

Denne kalder tagget *when* fra core biblioteket. Herudover bruger dette kald også noget EL. Dette findes i testparameteren. Disse tags kan også bruges til at gemme værdier i JavaScript variable. På denne side gemmes userID i en JavaScript variabel ved brugen af *out* tagget fra core biblioteket.

På denne side findes alle informationer omkring bruger. Selve bruger informationen er gemt i sessionen som en *User*-datatype. Disse informationer skrives ud på siden ved hjælp af EL. Brugerens kalender findes også på denne måde men da der skal ske 2 forskellige ting, afhængigt af om kalenderen er tom eller ej, så bruges et tag fra core biblioteket, der hedder *choose*. Denne virker lidt som en switch som kendes fra Java eller andre programmeringssprog. I et *choose*-tag findes nogle *when*-tag (se eksemplet ovenfor), som hver virker som en case. Til sidst i *choose* tagget kan et *otherwise* tag bruges. Dette virker som et default, hvis ingen af *when*-tags'ne bliver evalueret til sandt. I vores tilfælde kan kalenderen kun være tom eller ej. Derfor findes først en *when*, som tjekker om kalenderen for brugeren er tom. Er det falskt må der være noget i den og vi kan gå direkte til *otherwise*-tagget. Ved at bruge *otherwise* i stedet for endnu et *when* tag gør compileringen af siden en smule hurtigere.

Hvis kalenderen ikke er tom køres denne liste igennem via et *forEach* tag, som også findes i core biblioteket. Dette er en løkke som kan køre enhver itererbar collection igennem og virker på samme måde som en for løkke til Java generics. Ud fra brugerens kalender hentes hver VenueCalEntry fra databasen og skriver informationerne om denne ud på siden. Et eksempel på en for løkke som den førnævnte, bliver en sådan brugt til at skrive venues, som brugeren ejer, ud sidst på siden.

I Javascripten til denne side findes kun to eventhandlers. Disse styrer hvad der sker når brugeren klikker på linksne ”Create Venue” samt et link tilhørende en venue brugeren ejer. Disse er lavet med Dojo's *connect* metode på samme måde som nævnt i afsnittet ovenfor.

6.7.6 Implementering af use case – Opret Venue

En bruger kan oprette en venue ved at klikke på linket ”Create Venue” fra *UserPage.jsp*. Dette fører brugeren til *VenueSection/CreateVenuePage.jsp* hvori GUI'en til dette formål findes. Også i denne klasse bliver brugerens userID gemt i Javascripten. Ellers findes 2 text input elementer på siden samt en button.

Javascripten til denne side indeholder en eventhandler til knappen på siden. Når denne knap bliver klikket på bliver der kørt noget validering, som vil blive forklaret i et senere afsnit. Hvis valideringen gennemføres uden fejl bliver informationerne samlet ind og sendt videre til *tools/CreateVenue.jsp* via en *dojo.xhrPost* metode. Herfra kan data der komme tilbage igen være et af 3 tal:

Talkode	Betydning
-2	Der er sket en fejl under oprettelse af ny venue
-4	En venue med samme navn findes allerede i databasen
-5	Brugeren der prøver at oprette denne venue findes ikke i databasen.

Tabel 6 - Talkoder for opret venue

Tabel 4 viser hvilke svar opret venue kan få tilbage fra serveren samt deres betydning.

6.7.6.1 *CreateVenue.jsp*

Denne klasse bruges Ajax kaldet fra *CreateVenuePage.jsp* bliver brugt. Informationerne der er sendt med hentes ud fra request'en og gemmes. En af disse informationer er userID tilhørende den bruger der prøver at oprette den nye venue. En *UserController* bliver oprettet og validerer at brugeren findes i databasen. Hvis dette ikke er tilfældet bliver "5" skrevet i response. Når brugeren er valideret bliver en *VenueController* oprettet sammen en *Venue* med de nye data der skal indsættes. Herefter bliver *createVenue* kaldt fra *VenueController*'en. Denne sender videre til en metode af samme navn i *DBVenueQuerys*. Her bliver det valideret at der ikke findes en venue med samme navn. Herefter bliver informationerne samlet i to Strings. En til kolonnenavne og en til værdierne. Disse sendes videre til *DBQuerys* som indsætter dem i databasen.

6.7.7 Implementering af use case – Opret event

GUI'en i *VenueSection/CreateVenueEvent.jsp* har kun fire elementer i sig. Det er en input tekstboks til en titel og et textarea til info. Så findes der også en knap. Denne knap er styret af JavaScript som forklaret i afsnit ovenfor. Når den bliver klikket på sker noget validering inden informationerne sendes videre til *tools/RegisterVenueEvent.jsp*. Her sendes informationerne videre via en *VenueController.* Denne bruger metoden *createVenueCalEntry* fra *DBCalendarQuerys som igen bruger updateNewQuery* fra *DBQuerys*. Denne returnerer -2 ved fejl og 1 for succes. Brugeren sendes tilbage ved succes og får vist en fejl meddelelse ved fejl.

6.7.8 Implementering af use case – Tilmeld sig event

Til dette formål blev en side til selve eventet oprettet. Her findes alle informationer om eventet. Deltagerne bliver også skrevet ud på siden. Når disse skrives ud kontrolleres samtidig at brugeren selv er sat som deltager. Er dette tilfældet bliver en variable kaldet *isSignedUp* sat. Denne bruges senere på siden og styrer om der skal være en knap på siden, som brugeren kan klikke på for at skrive sig op som deltager til dette event. For at brugeren kan få lov til at melde sig til som deltager skal vedkommende være logget ind. Dette kontrolleres ved at kigge på variablen *userType* som er 'R' hvis brugeren er logget ind. Javascripten til denne side findes i filen *js/EventPage.js*. Her laves et Ajax kald til *tools/SignUpEvent.jsp*. Denne sender userID og venueCalEntryId videre til indsættelse i databasen. Igen betyder talkoden 1 at det er gået godt og siden bliver genindlæst for at få den opdaterede liste skrevet ud på siden.

6.7.9 Implementering af use case – Rate Venue

Når en bruger klikker på en venue bliver vedkommende sendt videre til *VenueSection/VenuePage.jsp*. På denne side findes alle informationer vedrørende denne venue. Der bliver lavet et tjek for at se om brugeren er den samme som den bruger der ejer denne venue. Hvis dette er tilfældet bliver et link vist som brugeren kan klikke på for at ændre informationer omkring denne venue. Denne funktionalitet er dog ikke implementeret. Derudover bliver denne venues kalender også vist på siden. Igen er det muligt for ejeren at klikke på et link til at oprette nye events. Ud over det er det fra denne side muligt at vurdere denne venue. Dette kræver dog at brugeren er logget ind. Dette kontrolleres ved at læse variablen *userType* og se om den er 'R'. Er den vises tallene 1-5 som brugeren kan klikke på alt efter hvor mange point vedkommende vil give denne venue.

Igen er det Javascripten der står for at sende de informationer, der skal gemmes i databasen, videre til serveren. Dette sker i *js/VenuePage.js*. Først samles alle informationer der skal bruges og gemmes i et map,

der hedder *content*. Hvis det går som det skal bliver siden genindlæst for at de nye ændringer kan vises på siden. Den jsp der står for at opdatere vurderingen er *tools/RateVenue.jsp*. Denne bruger en metode fra *Venue*-klassen, som hedder *updateRating()*. Denne metode opdaterer antallet af vurderinger (*ratingCount*) samt den pågældende venues nye vurdering (*ratingAvg*). Herefter opdateres denne venue i databasen ved at bruge *insertIntoVenue()*-metoden fra *VenueController*-klassen.

6.7.10 Validering af bruger input

Når en bruger skal skrive noget ind, via input elementer, til systemet, sker der noget validering af dette. Dette foregår først client-side via JavaScript og derefter findes der noget server-side validering af inputtet.

6.7.10.1 Client side

Dette foregår på alle sider, hvor en bruger kan skrive noget input. I det følgende vil der tages udgangspunkt i den validering der sker når en bruger første gang registrerer sig som ny bruger i systemet. Det der sker i denne kode er det samme som der sker andre steder, hvor der er brugerinput, dog med den forskel at her sker der en smule mere validering. Denne kode findes i *js/RegisterUser.js*. For at et element skal blive valideret skal det have klassen 'required' *(class='required').* Det første der sker, er at funktionen *checkInput* køres. Denne funktion henter en liste af alle elementer med klassen 'required'. Først kontrolleres det at der står noget i dem ved at tjekke at strengen i elementerne ikke er lige med den tomme streng. Er der et element, som fejler denne validering får elementet en rød kant. Dette gøres ved at kalde funktion *setInvalidStyle()* med elementet som argument. Kanten bliver lavet ved at manipulere elementets style. Denne funktion tager også en valgfri parameter, *errorText*, som indeholder en fejl meddelelse som udskrives ved siden af elementet i en rød farve. Tilbage i *checkInput()* køres hver node også igennem et tjek for om der bruges tilladte tegn. Dette sker i funktionen *validateChars()*, som også tager det pågældende element som argument. Denne funktion laver følgende tjek:

```
node.value.match(/^[A-Za-z0-9_\-@\.]*$/)
```

Her checkes værdien af elementet og der kontrolleres at det består af de tilladte tegn: A-Z, a-z, 0-9, _ (underscore), - (bindestreg), @ (snabel-a) og . (punktum). Findes der andre tegn end disse returnerer metoden *null*. Hvis bare en enkelt ting har fejlet i disse tjek returnerer *checkInput()* værdien *false* og valideringen stopper. Er dette derimod gået godt, bliver funktionen *validateInput* kaldt. Denne funktion kontrollerer først at emailen har det rigtige format. Til dette formål bruges en Dojo funktion der hedder *isEmailAddress()*, som findes i Dojox.validate. Herefter kontrolleres det at de to passwords, som brugeren har indtastet, er ens. Igen returneres *false* hvis valideringen på noget tidspunkt har fejlet og *true* i modsatte tilfælde.

6.7.10.2 Server side

Serverside validering sker efter client side valideringen. Dette sker på forskellige tidspunkter.

- Når en bruger registrerer sig som ny bruger af systemet, sker den en validering server side som tjekker at der ikke findes en bruger med samme brugernavn i databasen i forvejen.

- Når en registreret bruger opretter en ny venue tjekkes der først at denne bruger rent faktisk findes i systemet. Herefter tjekkes der også at der ikke allerede findes en venue med dette navn i systemet.

6.8 Implementering af mobil applikation

Dette afsnit omhandler implementeringen af mobilapplikationen. Det er delt på således at det vil blive gennemgået hvordan hver use case er blevet implementeret. Kommunikationen mellem telefon og server foregår via jsp'er, der ligger på serveren. Når en jsp er relevant for et afsnit vil denne også blive forklaret her, men den ligger ikke i mobilapplikationsprojektet men derimod i webprojektet.

6.8.1 Implementering af use case – Log in

Først skulle forsiden laves. Dette gøres ved at lave en klasse der extender *Activity*-klassen. Selve GUI'en er opbygget i XML og findes i to filer: *res/layout/main.xml* samt *res/layout/mainloggedin.xml*. Disse filer kan ikke navngives med standard camel-case da det er et krav at de er stavet med lutter små bogstaver. Disse to filer indeholder stort set det samme. Et TableLayout med to TextViews. Et til at begynde, med og et til slut. Imellem disse TextViews findes to Buttons. Den første Button er den eneste forskel mellem disse to views. Forskellen på dem er teksten der vises på knappen, hvad enten der skal stå log in eller log out. Denne tekst, sammen med teksten i det første TextView, findes i filen *res/values/strings.xml*. Denne fil indeholder alle statiske strenge i applikationen og refereres til fra layoutfilerne ved at skrive:

```
@string/FPlogin
```

Dette henter strengen *FPlogin* fra strings-filen. Inden en aktivitet kan benyttes i applikationen skal den registreres i *AndroidManifest.xml*. Dette gøres på følgende måde:

```
<activity android:name="Login">

</activity>
```

Selve funktionaliteten til forsiden ligger i filen: *src/dk.sich.mobipartyat.MobiPartyAtActivity.java*. Denne fungerer som main klasse til at starte med og det er *onCreate()*-metoden der bliver kaldt når denne klasse oprettes. Det første der sker i denne metode er at selve GUI'en bliver loaded ind. Først laves et tjek på om brugeren er logget ind. Dette sker ved at kalde metoden *isLoggedIn()* fra klassen *src/dk.sich.mobipartyat.server.ServerCon.java*. Dette returnerer en variabel, *isLoggedIn*, der er af typen boolean og holder styr på om brugeren er logget ind eller ej. Efter dette kaldes metoden *setContentView(R.layout.main)* eller for *R.layout.mainloggedin* for at få vist GUI'en. Klassen *R* er en autogenereret klasse der indeholder referencer til alle oprettede elementer, der findes i *res* mappen. Herefter sættes en onClickListener på hver af de to knapper. Hvis brugeren er logget ind, vil et klik på log out knappen logge brugeren ud ved at kalde *logOut()*-metoden fra *ServerCon*-klassen. Herefter ændres teksten på knappen til *FPlogin*-strengen. Hvis brugeren ikke er logget ind bliver en anden Activity startet. Denne Activity ligger i *src/dk.sich.mobipartyat.Login.java*. GUI'en til denne Activity ligger i *res/layout/login.xml*. Denne GUI består af et RelativeLayout. I dette layout ligger elementerne relativt til hinanden. Dvs at man kan definere et element til at ligger til højre eller under eller noget helt tredje, i forhold til et andet element. Denne indeholder tre TextViews, to EditTexts og to Buttons. EditTexts er elementer hvor brugeren kan skrive i. Disse EditTexts skal brugeren skrive brugernavn og password ind i. Den ene knap er til hvis brugeren vil tilbage til forsiden og den anden er til at sende brugernavn og password til serveren. Dette gøres ved at benytte bruge metoden *login()* fra *ServerCon.java*. Denne bruger en HttpClient til at forbinde til serveren.

Herefter laves en Get til serveren via URL'en defineret i metoden. Denne URI peger på jsp'en *tools/AndroidSection/AndLogin.jsp*, som findes i webprojektet. Det første denne klasse gør er at finde brugernavn og password frem fra de medsendte parametre. Herefter ser den i databasen, på samme måde som forklaret i webprojektet, om brugeren findes. Hvis dette er tilfældet bliver der svaret tilbage til telefonen via XML.

Til at lave denne XML er der oprettet nogle skema til at arbejde ud fra. Til at logge ind bruges skemaet: *User.xsd* som ligger i mappen *WEB-INF/xsd* i webprojektet. Dette ser ud som vist i tabellen.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
targetNamespace=http://www.w3schools.com
xmlns=http://www.w3schools.com
elementFormDefault="qualified">

<xs:element name="usertype">
 <xs:complexType id="user">
  <xs:sequence>
   <xs:element name="userId" type="xs:int" />
   <xs:element name="username" type="xs:string" />
   <xs:element name="email" type="xs:string" />
   <xs:element name="firstname" type="xs:string" />
   <xs:element name="lastname" type="xs:string" />
   <xs:element ref="usercalendartype" />
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="usercalendartype">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="usercalendarId" type="xs:int" />
   <xs:element name="usercalentrytype" minOccurs="0"maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="usercalentryId" type="xs:int" />
      <xs:element name="venuecalentryId" type="xs:int"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Tabel 7 - XML skema over user og usertype

User.xsd indeholdende elementer for en bruger (usertype) og en brugerkalender (usercalendartype)

Dette skema indeholder 2 elementer. Et til en bruger (usertype) og et til en brugerkalender (usercalendartype). Disse to skemaer indeholder så de informationer der svarer til *User* og *UserCalendar* fra *DataProject*'et. Usertype bruger så usercalendartype som et element. Ud fra disse skemaer er det muligt at bruge en indbygget funktion i Eclipse til at generere nogle klasser til opbygning af XML. Dette gøres ved at

bruge JAXB biblioteket, hvilket generer en klasse per element i skemaet samt en klasse, der hedder *ObjectFactory.java* og en klasse der hedder *package-info.java*. Disse klasser findes i *src/iPartyAt.User* i webprojektet. I *AndLogin.jsp* oprettes et objekt af *ObjectFactory.java*. Denne klasse kan så oprette elementer af usertype og usercalendartype. Først oprettes en usertype hvori alle brugerinformationer, som er hentet fra databasen, indsættes. Herefter laves en usercalendartype som indsættes i usertype. Til sidst bliver usertype "marshal'et" tilbage til mobilapplikationen. Et eksempel på en udskrift af denne XML kan ses i "Bilag – Bilag til udskrift af XML". Findes brugeren ikke i databasen bliver statusheaderen i response sat til http koden 403 (Forbidden).

Tilbage i mobilapplikationens *ServerCon* klasse bliver XML-svaret læst via en *Document* type fra *org.w3c.dom* pakken. Alle informationerne herfra bruges derefter til at oprette et objekt af typen *User* fra *DataProject*'et. Herefter bliver denne gemt og variablen *isLoggedIn* bliver sat til *true*.

Den anden knap i *Login.java* er back-knappen. Denne sender brugeren tilbage til forsiden. Dette gøres ved først at starte *MobiPartyAtActivity.java*, for derefter at slutte sig selv ved at bruge *finish()*-metoden fra moderklassen *Activity*.

6.8.2 Browsing – Generelt

Der er tre ting en bruger kan browse i: venues, events og sin kalender. Disse tre deler klassen *src/dk.sich,mobipartyat.Browse.java*. GUI'en til denne klasse ligger så i *res/layout/browse.xml*. Denne GUI er bygget op af en TabHost. Denne skal indeholde en TabWidget samt en FrameLayout. En TabHost laver et antal tabs i toppen af skærmen. Disse tabs oprettes i *Browse.java*. Efter GUI'en (uden tabs) er kommet på, oprettes et *LocalActivityManager* objekt så det er muligt at holde styr på flere forskellige Activities. Dette er lavet fordi, at i de tabs, der skal til at blive tilføjet, hører en Activity til hver enkelt. Herefter findes den TabHost, som blev defineret i XML filen og indsat i klassen tidligere, frem og sættes op med den nyligt lavede *LocalActivityManager*. Herefter laves en tab til at browse venues, en til at browse events og en, hvis brugeren er logget ind, til at browse brugerens kalender.

6.8.2.1 Implementering af use case – Browse venues

GUI'en til denne aktivitet ligger i *res/layout/browsevenues.xml*. Denne GUI indeholder et ScrollView som indeholder et TableView. Dette TableView inder holder en TableRow til at begynde med. Et ScrollView bruges til at gøre siden scrollable. Der bliver oprettet en TableRow per venue er det meget muligt at der kommer flere TableRows end der kan vises på skærmen. Den TableRow der findes til at begynde med indeholder tre TextViews. Disse tre Views indeholder den forklarende tekst for resten af siden.

Src/dk.sich.mobipartyat.Venues.java er den klasse der styrer denne aktivitet. Efter GUI'en er indsat i aktiviteten findes alle venues via *getVenues()*-metoden fra *ServerCon*-klassen. Denne metode bruger en Http-client til at hente alle venues fra serveren. Dette gøres ved et kald til jsp'en *tools/AndroidSection/AndShowVenues.jsp* fra webprojektet. Denne klasse henter alle venues og marshaller dem tilbage i XML format ved hjælp af et XML skema. Skemaet til en venue ser ud som vist i tabel 6.

Ved igen at bruge Eclipse's indbyggede funktion til autogenerering af hjælpeklasse, opretter dette skema 7 klasser som findes i *src/iPartyAt.Venue* i webprojektet. Et eksempel på en udskrift af denne XML kan ses i " Bilag – Bilag til udskrift af XML". Igen bliver denne information læst i mobilapplikationen via et *Document* og herefter lagt over i en liste af datastrukturen *Venue* fra *DataProject*'et, som returneres. I *Venues.java*, i

mobilapplikationen, oprettes en TableRow for hver venue i denne liste. Til hver TableRow bliver oprettet en onTouchListener, som sender brugeren videre til *VenueActivity.java*, der indeholder alle informationer om denne venue. Der vides hvilken venue der er klikket på ved at hente id'et fra den TableRow der er blevet trykket på. Dette id blev sat til venueId'et, da disse TableRows blev genereret.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="venuetype">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="venueId" type="xs:int" />
   <xs:element name="ownerId" type="xs:int" />
   <xs:element name="venuename" type="xs:string" />
   <xs:element name="venueaddress" type="xs:string" />
   <xs:element name="venueavg" type="xs:double" />
   <xs:element name="venuecount" type="xs:int" />
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="allvenues">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="venuetype" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="venuecalendar">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="venuecalId" type="xs:int" />
   <xs:element name="venueId" type="xs:int" />
   <xs:element ref="venuecalentry" minOccurs="0" maxOccurs="unbounded"
/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="venuecalentry">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="venuecalentryId" type="xs:int" />
   <xs:element name="info" type="xs:string" />
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="allcalentries">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="venuecalentry" maxOccurs="unbounded" minOccurs="0"
/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Tabel 8 - Skema for venuetype, venuecalendar, venuecalentry, allvenues og allcalentries.

6.8.2.2 Implementering af use case – Browse events

Aktiviteten til denne use case ligger i filen *src/dk.sich.mobipartyat.Events.java*, og GUI'en findes i *res/layout/browseevents.xml*. GUI'en er opbygget på samme måde som browse venues med den forskel at der her kun er to TextViews i stedet for 3. *Events.java* henter alle events ved at bruge metoden *getAllEvents()* fra *ServerCon.java*. Denne kalder klassen *AndShowEvetns* fra webprojektet. Denne henter alle events ud fra databasen, opbygger XML ved at bruge tabel 6 og marshaller dem tilbage til mobilapplikationen. Tilbage i mobilapplikationen bliver disse events itereret hen over og en TableRow bliver oprettet per event. Hvis brugeren er logget ind bliver der også genereret en knap som brugeren kan bruge til at tilmelde sig dette event. Det tjekkes dog først om brugeren allerede er tilmeldt eventet, da vedkommende ikke skal kunne tilmelde sig til samme event flere gange.

6.8.2.3 Implementering af use case – Tilmelde sig events

Når knappen ud for en event, på browse events siden trykkes på bliver denne knaps onClickListener kaldt. Denne kalder metoden *joinEvent()* fra *ServerCon*. Denne metode bruger to argumenter. Brugerens kalender id (userCalendarId) og denne events id (venueCalEntryId). Herefter sendes disse informationer, via en HttpClient, til webserveren og dennes *tools/AndJoinEvent.jsp*. I denne klasse indsættes eventet i brugerens kalender og den nye og opdaterede bruger bliver marshallet tilbage til mobilapplikationen. Her bliver den nuværende bruger opdateret og http status koden 200 (OK) sendes tilbage. Hvis noget sker fejl sendes http status koden 500 (Internal Server Error) tilbage.

Tilbage på browse events siden bliver en meddelelse vist for brugeren der fortæller om det er gået godt eller om der er sket fejl. Dette gøres ved at bruge en *Toast* som er en lille timet besked der vises på skærmen når dennes *show()*-metode kaldes. Hvis det hele er gået godt bliver listen af events genindlæst, så den knap der var til det pågældende event ikke længere findes. Denne genindlæsning sker ved at et broadcast bliver sendt ud. Denne broadcast kan samles op af en *BroadcastReceiver*, og når dette sker, og det er den korrekte broadcast der er sendt ud, bliver siden nulstillet for derefter at blive opbygget på ny.

6.8.2.4 Implementering af use case – Se kalender

Vis brugerens kalender aktivitet findes i *src/dk.sich.mobipartyat.Calendar.java*. GUI'en findes i *res/layout/browsecalendar.xml*. Selve GUI'en er stort set ens med de andre browse GUI'er. Denne GUI har dog kun en enkelt TextView med forklarende tekst. Da denne tab kun kan klikkes på når en bruger er logget ind er alle de events som brugeren er tilmeldt allerede hentet. Dette blev gjort i det øjeblik brugeren loggede ind. Så for at få disse skrevet ud, hentes listen af events fra brugerens kalender fra den gemte bruger (*user*) i *ServerCon*. Herefter itereres der hen over denne liste og en TableRow oprettes for hvert event, hvorefter det indsættes i det TableView der er defineret i GUI'en.

6.8.3 Implementering af use case – Rate venue

Det er muligt for en bruger at vurdere en venue inde fra venuesiden. For at komme ind på en venueside klikkes på en venue fra browse venues siden, som forklaret ovenfor. GUI'en til venuesiden ligger i *res/layout/venue.xml* og aktiviteten ligger i *src/dk.sich.mobipartyat.VenueActivity.java*. GUI'en er et TableLayout indeholdende TableRows der indeholder information forklaring. Hvis brugeren er logget ind er det muligt for vedkommende at vurdere denne venue. Dette gøres ved at klikke på en af knapperne 1-5, som hver har deres eget TextView. Til hvert af tallene bliver en onTouchListener tilknyttet som styrer hvad der sker når brugeren rører ved et af tallene. Hvis brugeren ikke er logget ind lavet et TextView der forklarer

brugeren at vedkommende skal være logget ind for at vurdere en venue. Når brugeren rører ved et tal kaldes *onTouch()* fra tallets onTouchListener. Denne metode kalder *rateVenue()* fra *ServerCon*, som sender venue id og tallet der er blevet klikket på videre til *AndRateVenue.jsp* fra webprojektet via en HrrtClient. Her bliver denne venues rating opdateret på samme måde som forklaret i webprojektet, hvorefter en header indeholdende den nye og opdaterede rating findes. Tilbage i mobilapplikationen bliver denne venue fundet i listen af venues hvorefter den nye rating sættes ind. En broadcast sendes så, som giver besked på at en nyere rating findes, hvorefter en *BroadcastReceiver* fjerner den gamle og indsætter den nye.

7 Test

I test afsnittet, testes projektet i en Blackbox test og en browser test. I bilagene vil der være skærmdumps (screen shots) af de testede funktioner

7.1 Browser test

Browser testen har til formål at finde ud af, hvordan siden ser ud i forskellige browsere. Da der findes mange browsere på forskellige styresystemer, har vi valgt at undersøge, hvilke der benyttes mest. Ifølge w3schools.com er de mest populære browsere fra 2011 følgende:

2011	Internet Explorer	Firefox	Chrome	Safari	Opera
May	24.9 %	42.4 %	25.9 %	4.0 %	2.4 %

Tabel 9 - http://www.w3schools.com/browsers/browsers_stats.asp

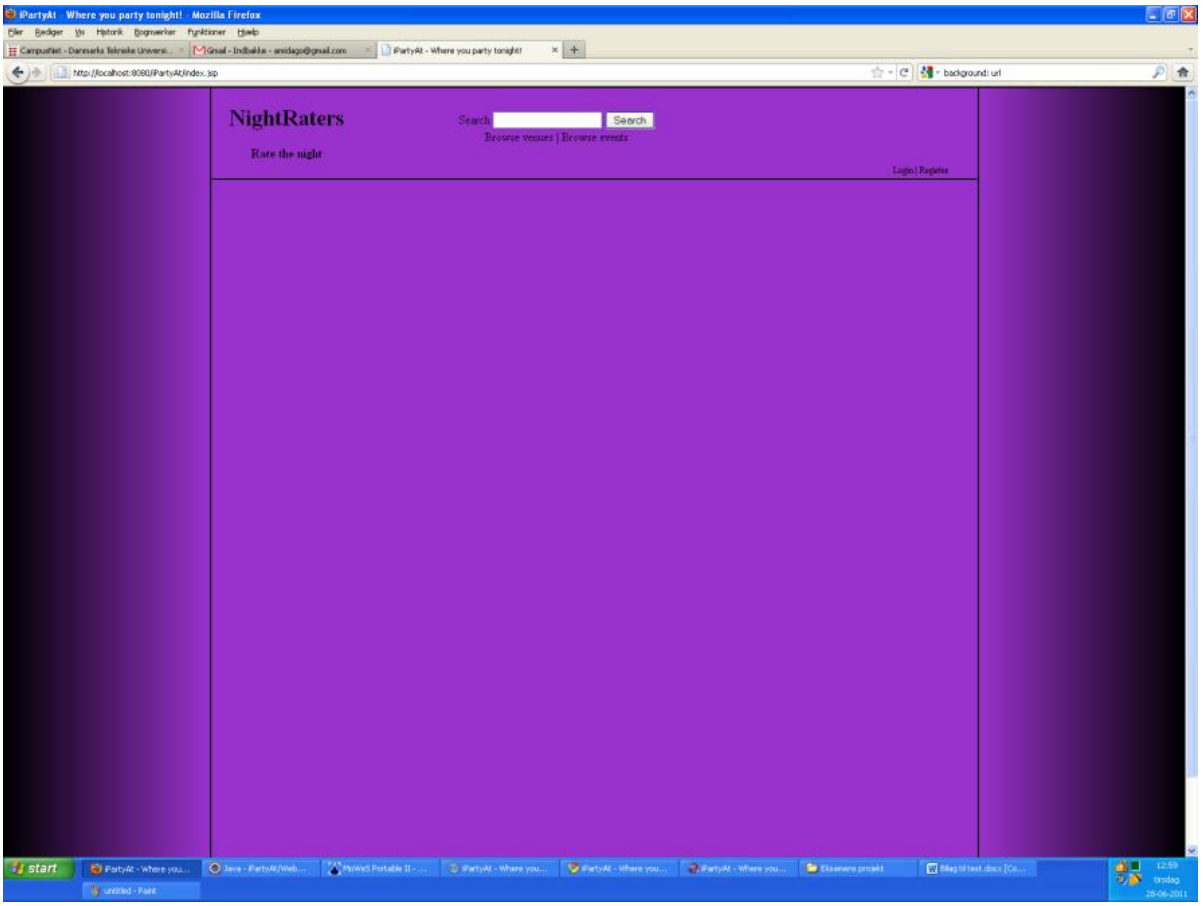
Safari benyttes hovedsageligt af Mac bruger med styresystemet Mac OS X. Resten er Windows baserede, men virker også på andre styresystemer, herunder fx på Mac OS X, Ubuntu, Unix eller Linux (Ubuntu er dog Linux baseret og Mac OS X er Unix baseret). Disse 5 browsere udgør tilsammen 99,6 % af alle browsere der benyttes, det betyder at hvis hjemmesiden fejler i andre browsere er der max tale om 0,4 %.

Som regel er w3schools en ret pålidelig kilde og vi har derfor valgt at stole på tallene. På baggrund af dette har vi valgt at teste siden med de 5 ovenævnte browsere.

Der testes kun på forsiden, da der ikke er udviklet noget layout til de andre sider endnu. Ved fremtidig implementering af layout, prioriteres det, at de ser ud som de skal i alle 5 browsere.

7.1.1 Resultat

Forsiden så ud som den skulle i alle 5 browsere. Der er taget screenshots af resultaterne vedlagt i ”Bilag til test – Browser test”. Til test under implementering benyttede vi Mozilla Firefox. Følgende screenshot viser forsiden, som den blev designet, og hvordan den forventes at se ud i de andre browsere.



Figur 22 - Layout i mozilla

7.2 Blackbox test

Blackbox testen er foretaget med Mozilla Firefox 4.0.1. Ifølge w3schools.com benyttede 42,4 % af alle internet brugere Firefox i maj 2011[4]. Heraf benytter 23,6 % version 4 og 15,7 % benytter version 3.

Blackbox princippet går ud på at der gives et input og baseret på det input, er der en forventning om outputtet. Hvis outputtet ikke er som forventet, kan det skyldes en fejl og en fejlsøgning kan påbegyndes. En af fordelene ved Blackbox test er at den er let at udføre. Den er bare ikke så informativ når man endelig finder fejl.

Testen er delt op i 8 testcases. Først testes use casene, om de kan føres igennem, derefter testes validering og funktionalitet.

[4] http://www.w3schools.com/browsers/browsers_firefox.asp

7.2.1 Test case – Besøgende

Hvad der testes (Bilag til test)	Prækondition	Forventet resultat	Faktiske resultat	Status
Oprettelse af bruger (Bilag til test A. 1)	Den besøgende vælger "register" og udfylder felterne korrekt	Brugeren bliver oprettet og får besked herom	Brugeren oprettes og sendes til forsiden igen. Men modtager ingen besked	OK
Log ind (Bilag til test A. 2)	Brugeren vælger "Log in" og indtaster sine log in informationer i felterne	Brugeren videresendes til brugersitet	Brugeren videresendes til brugersitet	OK

7.2.2 Test case – Medlem

En prækondition for disse test er at brugeren er logget på systemet.

Hvad der testes	Prækondition	Forventet resultat	Faktiske resultat	Status
Redigér bruger information	Bruger vælger "Change personal information" og udfylder felterne korrekt	Brugeren får muligheden for at ændre sine informationer og gemme	Intet sker når der trykkes på knappen "Change personal information"	FEJL
Find venue (Bilag til test B. 1)	Brugeren vælger "Browse venues"	En liste over eksisterende venues vises	Listen vises	OK
Vurder venue (Bilag til test B. 2)	Brugeren er inde på venuesitet	Karakteren gemmes efter brugeren har trykket på 1,2,3,4 eller 5	Karakteren gemmes og Venue Rating opdateres med det samme. Se rating for "DiskotekSjov" På Bilag til test B.1 og dernæst på B.2	OK
Tilmelding til event (Bilag til test B. 3)	Brugeren er inde på et event og trykker "Sign up for event"	Brugeren tilmeldes og kan ses under "Participants"	Brugeren tilmeldes og står som "Participant"	OK
Se tilmeldte events i kalender (Bilag til test B. 4)	Brugeren trykker på "my account"	Et afsnit med sin kalender vises for brugeren	Inde på brugersiden er "your calender" tomt selvom der er tilmeldt events	FEJL
Opret venue (Bilag til test B. 5)	At venuen ikke eksisterer i forvejen	At venuen oprettes og brugeren får besked derpå	Venuen oprettes, men der modtages ingen besked herom	OK
Log ud	Ingen	Brugeren logges af og videresendes til forsiden	Brugeren logges af og videresendes til forsiden	OK

FEJL "Redigér bruger information": Denne opstår fordi den del ikke var blevet implementeret, men vi har stadig medtaget knappen som en funktion.

FEJL "Se tilmeldte events i kalender: Den anden fejl i "Se tilmeldte events i kalender", opstår fordi cachen ikke opdatere før man logger på systemet igen. Dvs. man skal logge af og på for at se nye tilmeldte events.

7.2.3 Test case – Bruger Venue

En prækondition der gælder for disse er at brugeren har oprettet en venue.

Hvad der testes	Prækondition	Forventet resultat	Faktiske resultat	Status
Indtaste information om venue (Bilag til test C. 1)	Brugeren vælger "Create a venue"	Brugere videresendes til "Create venue" sitet	På "Create venue" sitet skal der indtastes Venue name og Adress	OK
Opret event (Bilag til test C. 2)	Brugeren befinder sig på venuesitet og trykker "Create new event"	Brugeren videresendes til CreateVenueEvent	Brugeren sendes til CreateVenueEvent og skal indtaste titel og info for eventet.	OK
Rediger information	Brugeren befinder sig på venuesitet og har indtastet informationer der kan redigeres	Brugeren videresendes til "Redigér venue informationer" sitet	Intet sker når der trykkes på linket	FEJL

FEJL "Rediger information": Denne fejl opstår, fordi funktionen ikke er implementeret, men knappen/linket er stadig synlig.

7.2.4 Test case – Validering

7.2.4.1 Log in validering

Hvad der testes	Prækondition	Forventet resultat	Faktiske resultat	Status
Forkert brugernavn (Bilag til test D.1)	Brugernavnet der forsøges testet med ikke eksisterer	Brugeren meddeles at der er en fejl	Fejlen " Username and/or password incorrect." vises	OK
Forkert password (Bilag til test D.2)	At brugeren eksisterer	Brugeren meddeles at der er en fejl	Fejlen "Username and/or password incorrect." vises	OK
Ingen indtastning (Bilag til test D.3)	Ingen	Brugeren meddeles at der ikke er indtastet noget	Fejlen "* Field(s) marked red has not been correctly filled out." vises	OK

7.2.4.2 Opret bruger validering

Hvad der testes	Prækondition	Forventet resultat	Faktiske resultat	Status
Manglende udfyldning af felterne, enkeltvis (Bilag til test E.1)	De andre felter udfyldes korrekt	Det felt der ikke er udfyldt, meddeler fejl	Hver enkelt felt bliver omkranset af en rød streg og en fejl meddelelse vises	OK
Forkert e-mail-adresse syntax (Bilag til test E.2)	De andre felter udfyldes korrekt	Brugeren bedes indtaste en gyldig email adresse	Fejlen ” Please type in a valid email.” vises	OK
Forkert gentagelse af password (Bilag til test E.3)	De andre felter udfyldes korrekt	Brugeren meddeles at de to passwords ikke er ens	Fejlen ” Passwords do not match. Please enter the same password in the box above.” vises	OK
Brug af specielle tegn udover ”_”, ”-”, ”@” og ”.” (Bilag til test E.4)	De andre felter udfyldes korrekt	Brugeren meddeles at der er ulovlige tegn in feltet	Fejlen vises ud fra feltet hvor et ulovligt tegn er brugt	OK
Fejl ved oprettelse af allerede eksisterende brugernavn (Bilag til test E.5)	En bruger med det testede brugernavn skal allerede eksistere	Brugeren får en fejlmeddelelse om at brugernavn allerede eksisterer	Brugeren kan godt oprette sig selvom brugernavnet allerede eksisterer	FEJL
Fejl ved oprettelse af bruger hvor email allerede er brugt (Bilag til test E.6)	En bruger med den testede email er allerede oprettet	Brugeren får en fejl om at e-mailen er brugt før	Fejlen ” A user with that email already exists.” vises ved email feltet	OK

FEJL “Specielle tegn udover ”_”, ”-”, ”@” og ”.””: Fejlen når at blive vist, men informationerne bliver stadig gemt i databasen. Denne fejl kan skyldes

7.2.4.3 Opret venue validering

Hvad der testes	Prækondition	Forventet resultat	Faktiske resultat	Status
Manglende udfyldning af felter (Bilag til test F. 1)	De andre felter udfyldes korrekt	En fejlmeddelelse beder brugeren udfylde felterne	Fejlen ” * Field(s) marked red has not been correctly filled out.” vises	OK
Brug af specielle tegn udover ”_”, ”-”, ”@” og ”.” (Bilag til test F. 2)	De andre felter udfyldes korrekt	En fejlmeddelelse beder brugeren om at udfylde feltet korrekt	Fejlen ” Please only use letter (A-Z & a-z), numbers (0-9) and signs (- .@).” vises ved de felter hvor tegnene er benyttet	OK

Fejl ved oprettelse af allerede eksisterende venue	Venue med det angivne navn skal allerede eksistere	En fejlmeddelelse fortæller brugeren at venue navnet allerede eksisterer	Fejlen ” A venue with that name already exists in the system.” vises	OK

7.2.4.4 Sortering af venue liste

Hvad der testes	Prækondition	Forventet resultat	Faktiske resultat	Status
Sortering efter navn (Bilag til test G. 1)	At der er oprettet venues i systemet	Venues’ne vil blive sorteret i alfabetisk rækkefølge fra a til z	Venues’ne sorteres fra a til z	OK
Sortering efter adresse (Bilag til test G. 2)	At der er oprettet venues i systemet	Venues’ne vil blive sorteret efter adresse i alfabetisk rækkefølge fra a til z	Venues’ne sorteres efter adresse fra a til z	OK
Sortering efter rating (Bilag til test G. 3)	At der er oprettet venues i systemet og de har en rating	Venues’ne vil blive sorteret efter højeste rating mod laveste	Venues’ne sorteres med højeste rating øverst mod laveste	OK

7.2.5 Test case – Mobil app brugere

Hvad der testes	Prækondition	Forventet resultat	Faktiske resultat	Status
Log in (Bilag til test H.1)	Brugeren har oprettet sig via. Hjemmesiden	Brugeren logges på systemet	Brugeren er logget på systemet	OK
Vurder venue (Bilag til test H.2)	Brugeren er logget på og inde på en venue	Venue får en ny rating som straks ses på venue siden.	Venue får ny rating og siden opdateres korrekt.	OK
Se kalender (Bilag til test H.3)	Brugeren er logget på	Brugeren får lov at se sin kalender	Brugeren kan se sin kalender.	OK
Log ud (Bilag til test H.4)	Brugeren er logget på	Brugeren logges af.	Brugeren logges af.	OK
Tilmeld Event (Bilag til test H.5)	Brugeren er logget på	Brugeren bliver tilmeldt event som kan ses i kalenderen.	Brugeren tilmeldes event som kan ses i kalenderen hvis der findes events i kalderen fra start. Kan ikke ses i kalenderen hvis kalenderen var tom fra start.	FEJL

7.3 Konklusion af test

Ud af de fejl der er fundet, skyldes mange af dem manglende implementering. Vi har testet på det funktionelle, og det der kræver bruger input, da det er mest relevant. Dette betyder vi ikke har taget højde for bugs/fejl, som kan opstå pga. database problemer, forbindelses problemer eller lignende.

8 Forbedringer

Dette afsnit omhandler de forbedringer der er til prototypen som den er nu. Forbedringerne er en del af de mål der skal opnås før prototypen kunne være klar til fremvisning. Større funktioner hører under udvidelsesafsnittet.

8.1 Kravændring

Undervejs i implementeringen blev følgende krav ikke implementeret fuldt ud

- En bruger skal kunne vurdere en venue en gang og derefter rette sin vurdering.
    - Dette krav er implementeret uden validering, således en bruger kan afgive et uendeligt antal stemmer.
- En brugers personlige oplysninger.
    - Informationer omkring brugerens fødselsdato samt køn gemmes ikke. Brugeren har heller ikke mulighed for at rette personlige oplysninger.
- Venue Events
    - En bruger kan ikke framelde sig et event, som brugeren har tilmeldt sig.
- Slet venue
    - Det er ikke muligt for en venueejer at slette en venue igen.
- En ejer skal ikke kunne vurdere sin egen venue
    - Der mangler validering på dette krav, en bruger kan godt vurdere sin egen venue.
- Anmeldelse af venue
    - Dette krav blev ikke implementeret
- Feedback til brugeren
    - Brugeren får ikke noget feedback i form af beskeder når fx en venue er oprettet uden fejl.

8.2 Validering

Følgende liste indeholder de forbedringer der kan tilføjes til valideringen:

- Brugerne kan lige nu stemme flere gange på samme venue, dette skal ændres så de kun kan stemme en gang, medmindre de har slettet deres tidligere vurdering
- Venue ejeren skal ikke kunne stemme på sin egen venue
- Hvis en bruger prøver at oprette en bruger med en email der er brugt, får de en fejlmeddelelse om at emailen er brugt. Det kan evt. ske for en bruger at de har glemt at de havde oprettet sig, eller glemt sin kode. Det ville derfor være passende hvis de modtog en mulighed for at få tilsendt koden til deres email.

8.3 Funktioner og rettelser

Følgende liste indeholder de funktioner og rettelser, der kan forbedre prototypen som den er nu, og som samtidig ikke kræver omfattende implementering:

- Dato og tidspunkt, på events og i kalenderen. Pt. står der ”Her skal findes en dato”
- Titel på events og i kalenderen. Pt. står der ”Her skal findes en titel”
- Et layout til kalenderen, således at man kan se alle dagene i en måned el lign.

- Venue rating repræsenteres lige nu som en double, det kan give mange decimaler efter kommaet. Dette skal rettes til maks. 2 decimaler efter kommaet.
- Implementering af ”Redigér informationer”, således fejlene fundet i testen undgås
- En funktion der opdaterer cachen når man tilmelder sig events. Og dermed ikke behøver at logge af og på for at se sine events i kalenderen.
- Når man trykker på ”Enter” tasten at man som standard, trykker på ”submit” – knappen. For eksempel under log in og register.
- Navigationen på siden skal optimeres, så man ikke er afhængig af ”tilbage” knappen.
- En funktion, der gør at brugeren kan få nulstillet sit password vha. email, hvis de nu glemte deres brugernavn eller kode.
- Man skal kunne logge på systemet med sin email eller brugernavn, i tilfælde af en bruger glemmer sit brugernavn
- Adressen på en venue skal kunne repræsenteres vha. Google maps.

9 Offentliggørelse

Det ville ikke kræve meget mere, før en reel "release" kunne lade sig gøre, hvis forbedringer blev implementeret. Følgende afsnit viser, hvad der mangler før en officiel release kunne godkendes og produktet kunne benyttes til at udfylde de nødvendige behov. Samt en releaseplan og hvilke PR muligheder der er overvejet

9.1 Layout

En vigtig udvidelse er layoutet. Finde en stil eller tema der passer til emnet og derefter style det. Et flot layout betyder enormt meget, det skal både give et godt førstehåndsindtryk og være behageligt at benytte. Dette inkluderer også at finde det rette navn og logo for sitet. Der skulle derfor investeres tid i udviklingen af et flot CSS til siden.

9.2 Webhotel og domæne

En anden nødvendighed er at finde det rette webhotel til at hoste sitet. Des mindre nede tid des bedre, det rette webhotel skal være så stabilt som muligt og samtidig kunne tilbyde de nødvendige services.

Services:

- **Web hosting** – til at starte med kan man starte med et par gigabyte og så udvide alt efter behov.
- **MySQL** – da databasen er lavet i MySQL, skal webhotellet understøtte dette.
- **PHP** – da PHPMyAdmin benyttes til at konfigurere databasen, er det en nødvendighed at det understøttes
- **JSP** – da hjemmesiden er skrevet i JSP er det nødvendigt at det understøttes

9.3 Reklame

Reklamering for siden er en vigtig del af lanceringen. For at få folk til at benytte siden, skal de vide den eksisterer. Der findes en masse måder at gøre dette på, både gratis og hvor det koster penge. Effektive måder at gøre det gratis på, kan være følgende

- **Facebook** – Oprette en side på facebook for siden. Skriv de vigtigste detaljer ned og indsæt link osv. Der findes næsten en facebook side for alle franchises, da det er en effektiv og gratis måde at reklamere. Derudover skal der sendes besked til så mange som muligt, dette kan gøres vha. venners venner
- **Twitter** – Ligesom facebook, er twitter også en kæmpe portal hvor der kan blogges så flere millioner kan læse det.
- **Andre sociale medier** – Der findes mange sociale medier der kan gøres reklame på, des flere des bedre.
- **Klistermærker "stickers"** – klistermærker er en stadig effektiv måde at reklamere for en hjemmeside. De kan hænge overalt og koster ikke meget at få produceret. Den brugergruppe hjemmesiden henvender sig til, er hovedsageligt unge. Det ville derfor være oplagt at sprede dem i "unge" miljøer.
- **Search Engine Optimization** – Ved at optimere SEO, vil siden rangere højere på de forskellige søgetjenester, dette er en reklame i sig selv og meget effektivt.

- **Online Ads** – Man kan lave en advertisement (ad), som kan vises på forskellige sider. Det fungere ligesom en normal reklame, man altså betaler for at få vist. Denne løsning er dog ikke en vi ville benytte os af til at starte med.
- **Kontakt til nyhedskilder –** Ved at fortælle nogle journalister om projektet, er der mulighed for at de vil/kan skrive en artikel om det.

Til at starte med, har projektet ikke til formål at sælge et produkt, eller tjene penge på reklame. Det er derfor begrænset hvad der kan bruges af penge og de gratis metoder for reklame er derfor mest oplagt. Under alle omstændigheder skal der foretages en eller anden form for reklame så snart projektet får release.

9.4 Release plan

Før en release, skulle der helst oprettes så mange venues som muligt. Dette kunne gøres ved at kontakte alle de venues man kan og fortælle dem hvordan og hvorledes. Hvorvidt de gider bruge tid på det er op til dem, men hvis det kun tager 10 minutter, kunne det forventes at de fleste godt gad.

Årsagen til at få oprettet nogle venues først, er det indtryk det giver de besøgende. Hvis en besøgende brugte hjemmesiden og ser meget få venues, ville chancen for at han/hun vender tilbage blive lille og "chancen" dermed forspildt. Da førstehånds indtrykket er så vigtigt, ville det derfor være godt at optimere chancen for et godt førstehåndsindtryk. Netop ved at lade så mange venues som muligt oprette sig først.

10 Udvidelser

Projektet indeholder en realistisk forestilling om hvad der kunne nås. Vi har defineret og implementeret en afgrænset version af hvad vores ide går ud på. Muligheden for udvidelser er stor og kun fantasien sætter grænser.

10.1 Søgefunktion

En af de større udvidelser, som har høj prioritering er søgning. Ikke blot en søgning efter venues, men en avanceret søgefunktion. Den avancerede søgning skal gøre det muligt at søge på venueinformationer. Så hvis man fx vil finde en venue der ligger på vesterbro og som spiller salsa musik, kan man finde den ved at søge "Musik: Salsa, Lokation: Vesterbro" og så få en liste af de venues der opfylder disse krav. Lokation kan evt. indtastes som postnummer i stedet. Selve implementeringen af den avancerede søgefunktion kunne umiddelbart implementeres ret let. Så snart der blev lavet en søgefunktion, kunne den sættes til at søge i venuedatabasen efter en match i begge tabeller, for musik og lokation.

10.2 Kalender

For at kunne give et endnu større overblik skal kalenderen udvides, således at man kan vælge at se en kalender for alle events, og ikke kun ens tilmeldte events.

10.3 Venue information

Venues'ne skal kunne indeholde informationer. Følgende er en liste over hvilke:

- Musik – hvilken slags musik der spilles
- Priser – indgang, barpriser, garderobe osv.
- Bartype – Om det er en cocktail bar, værtshus osv.
- Aldersgrænse – Minimumsalder for at komme ind
- Åbningstider – Hvornår holder venuen åben.
- Dresscode – Hvis der er regler for beklædning

Informationerne skal ikke være obligatoriske, da det blot skal være en ekstra service. Gjordes de obligatoriske, kunne det muligvis få nogle venueejere til ikke at oprette deres venues, pga. det ville være tidskrævende, eller de måske ikke lige havde alt informationen ved hånden og derfor måtte vente.

Disse informationer er også nødvendige for at kunne tilføje den avancerede søgning.

11 Konklusion

Da vi begyndte på projektet havde vi en meget omfattende ide, som vi rigtig gerne ville have lavet så meget som muligt af. Det var urealistisk at få implementeret hele løsningen og vi afgrænsede det derfor. Målet for projektet blev så at udvikle en prototype, af hvad vi godt kunne tænke os at lave senere hen. Vi fik defineret krav, analyseret og designet dem, for til sidst at implementere det.

Som tiden skred frem blev det klart for os at vi ikke fra kunne nå alle kravene, vi opdelte dem derfor i ”Nice to have” og ”Need to have” og sigtede efter at få implementeret ”Need to have” kravene. En af hovedårsagerne til at vi havde overestimeret vores evner i forhold til kravene, var primært vores kendskab til de forskellige værktøjer der blev brugt. Der blev derfor brugt en del tid på at sætte udviklingsmiljøet op og få de forskellige dele til at spille sammen. Vi havde ikke medtaget dette i vores planlægning og undervurderet vigtigheden af det. Det at vi ikke havde planlagt det gjorde at vi arbejdede efter tidsplanen under det meste af projektet.

På trods af dette fik vi lavet en prototype, der fint demonstrerer, hvad vores ideer går ud på og på en sådan måde at vi nu let vil kunne videreudvikle det.

Vi har fået et godt indblik i hvordan udvikling til Android fungerer, og et indtryk af at dette er en teknologi i en rivende udvikling. En udvikling vi kan sige, efter dette projekt, gerne vil være en del af. Noget ingen af os havde så meget som snuset til inden projektets start. Ydermere har vi fået et endnu bedre kendskab til webudvikling end vi havde inden påbegyndelsen af projektet og vigtigheden i, at lave et produkt der er let for brugeren at finde rundt i, er blevet klar for os.

Vi forventer at færdiggørelsen af projektet, vil forekomme en del lettere. Da vi nu har en platform for at kunne arbejde mere effektivt. Det næste skridt er at få alt der er medtaget i dette projekt til at virke, før vi foretager os nogen form for release.

12 Litteraturliste

Websites

Navn	Adresse
w3schools	http://www.w3schools.com/
Wikipedia	http://da.wikipedia.org/
Android developers	http://developer.android.com/
Dojo	http://dojotoolkit.org/

13 Bilag

13.1 Vedlagt DVD

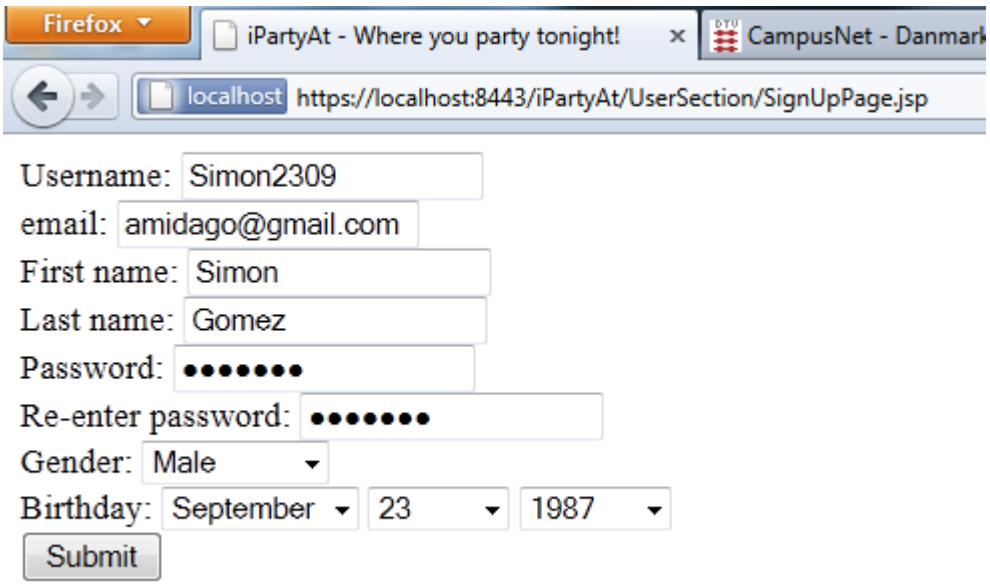
Den vedlagte DVD indeholder føglende:

- Rapport i PDF format
- Mappe indeholdende:
  - Android-SDK-Windows (development kit til android)
  - Apache – tomcat-7.0.11 (Tomcat server)
  - Eclipse (Primære udviklingsværktøj)¨
  - Workspace (indeholdende projektet og alt kildekode)
  - Mowes (Databasen + database server)
  - Jakarta – tag libs – standard – 1.1.2 (JSTL jar)
  - Jaxb (Jaxb jar)
  - MySQL – connecter – java – 5.1.15 (connecter mellem database og webserver)
  - .Keystore fil (sikkerhedsfil)
- Brugervejledning til installation af prototypen

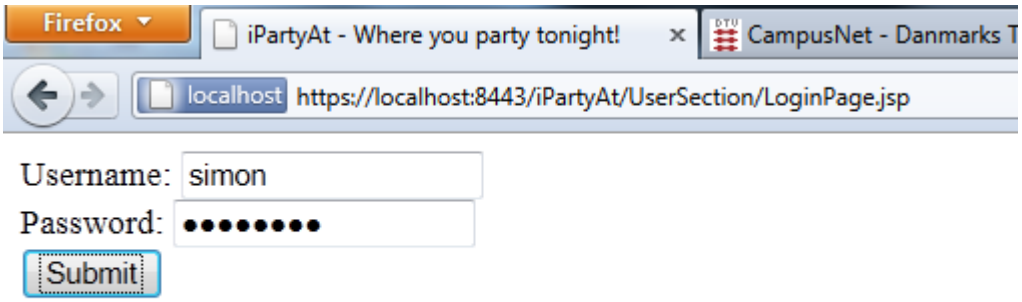
13.2 Bilag til test

13.2.1 Blackbox test

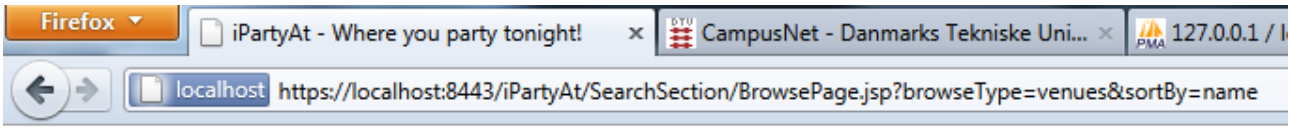
13.2.1.1 A. 1



13.2.1.2 A. 2

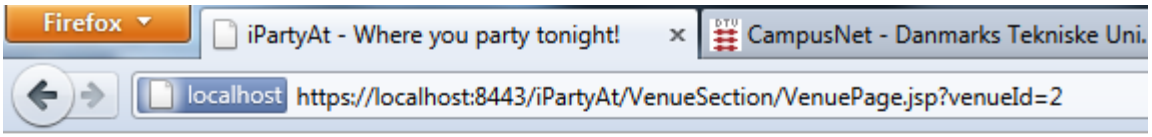


13.2.1.3 B. 1



Name	Address	Rating	Number of ratings
Bar-By	Gothersgade_22	4.88	25
DiskotekSjov	Borgergade_23	4.75	12
FeststedDanmark	testgade44	2.5000000000000001	50
FunBarDk	Testgade	3.8	5

13.2.1.4 B. 2



Frontpage
Venuename: DiskotekSjov
Address: Borgergade_23
Change venue information: Change venue information

Venue Rating: 4.769230769230769 of 13 votes
Rate this venue: 1 2 3 4 5

Calendar:

Create a new event.

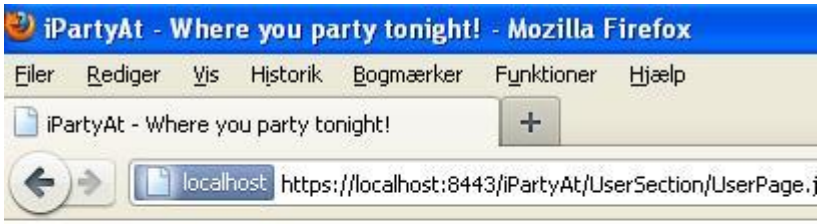
13.2.1.5 B. 3

Test test test tesdt

Participants for this event:

che
che2
ChristianE

13.2.1.6 B. 4



Frontpage
Username: ChristianE
Firstname: Christian
Lastname: Esbensen
email: ChristianE@student.dtu.dk
Change information and password: Change personal information

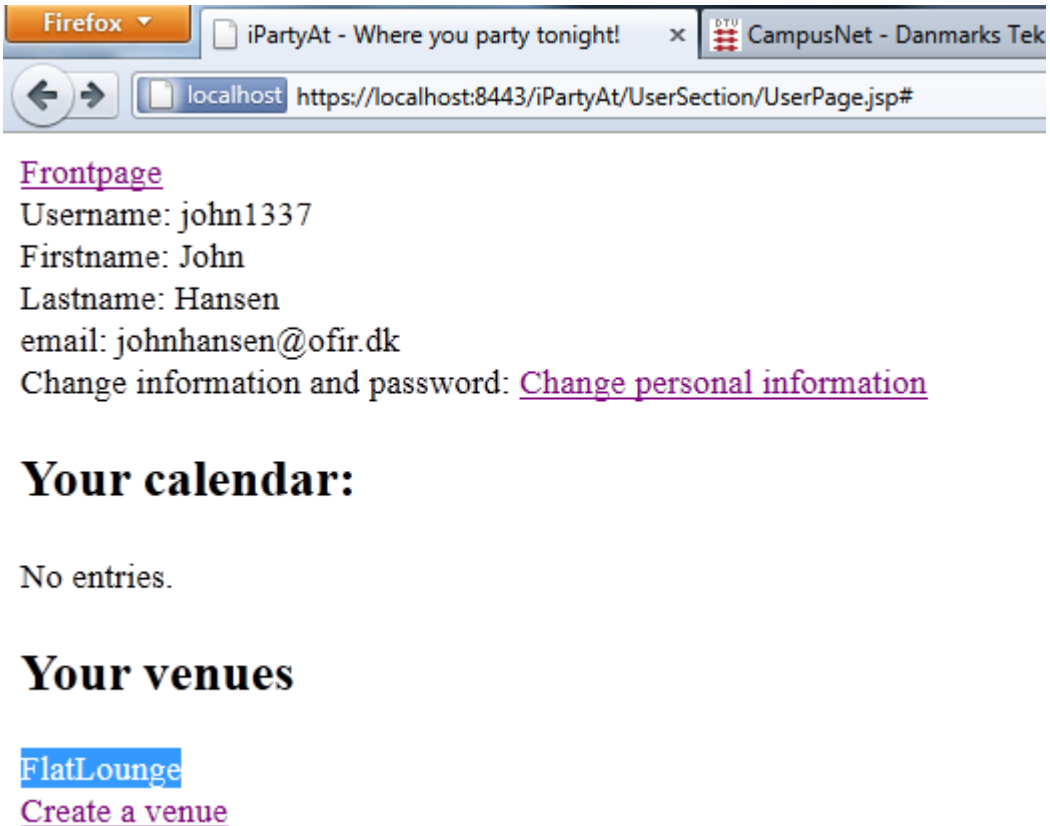
Your calendar:

No entries.

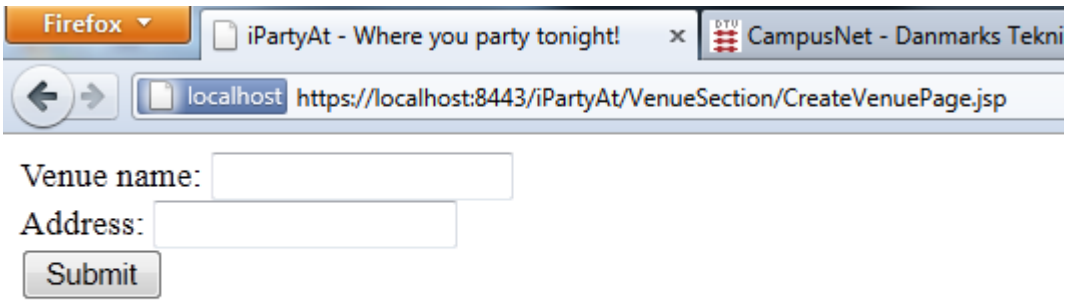
Your venues

Create a venue

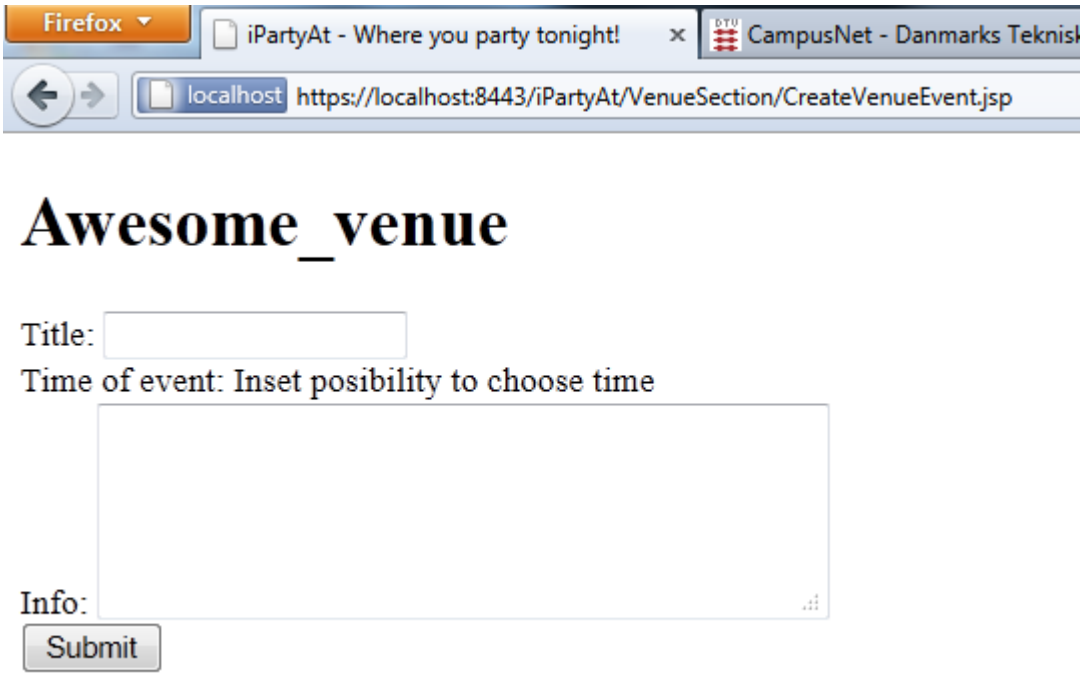
13.2.1.7 B. 5



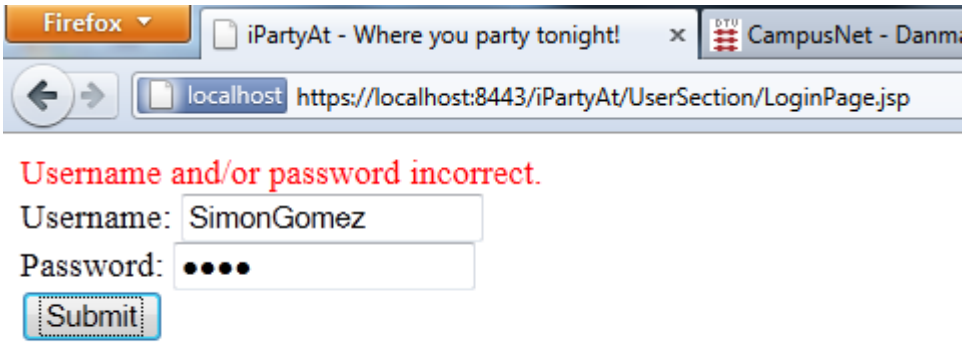
13.2.1.8 C.1



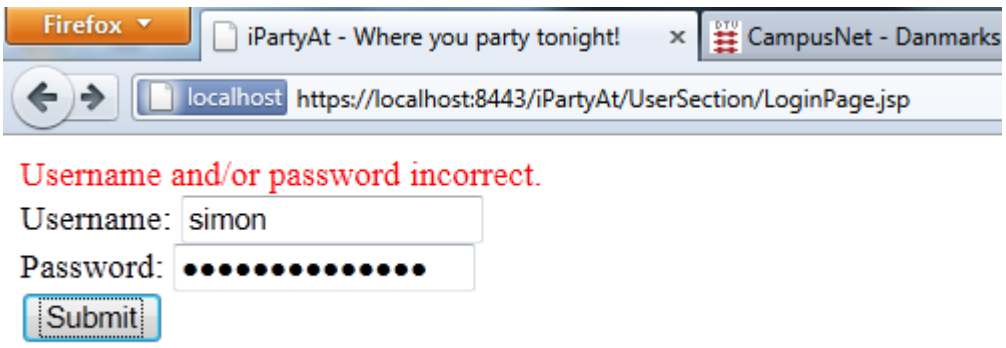
13.2.1.9 C.2



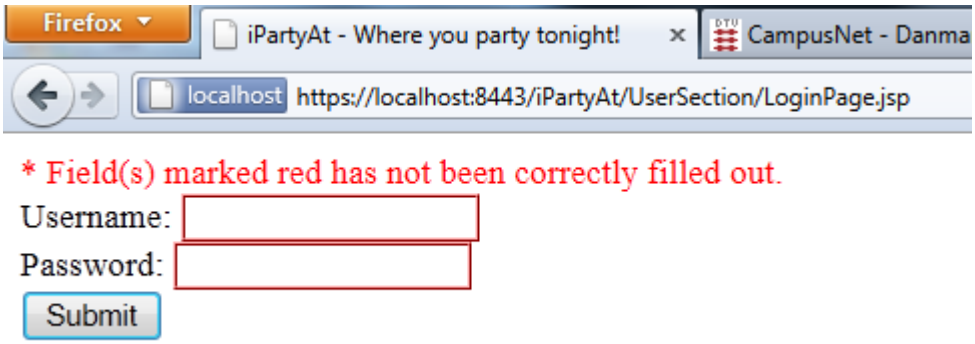
13.2.1.10 D.1



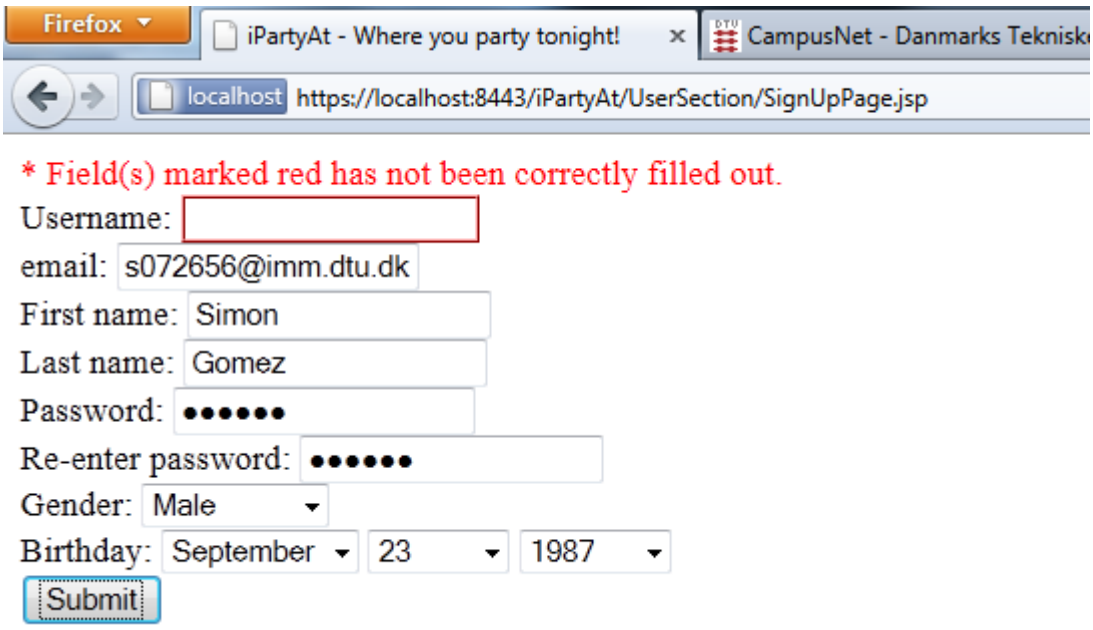
13.2.1.11 D. 2

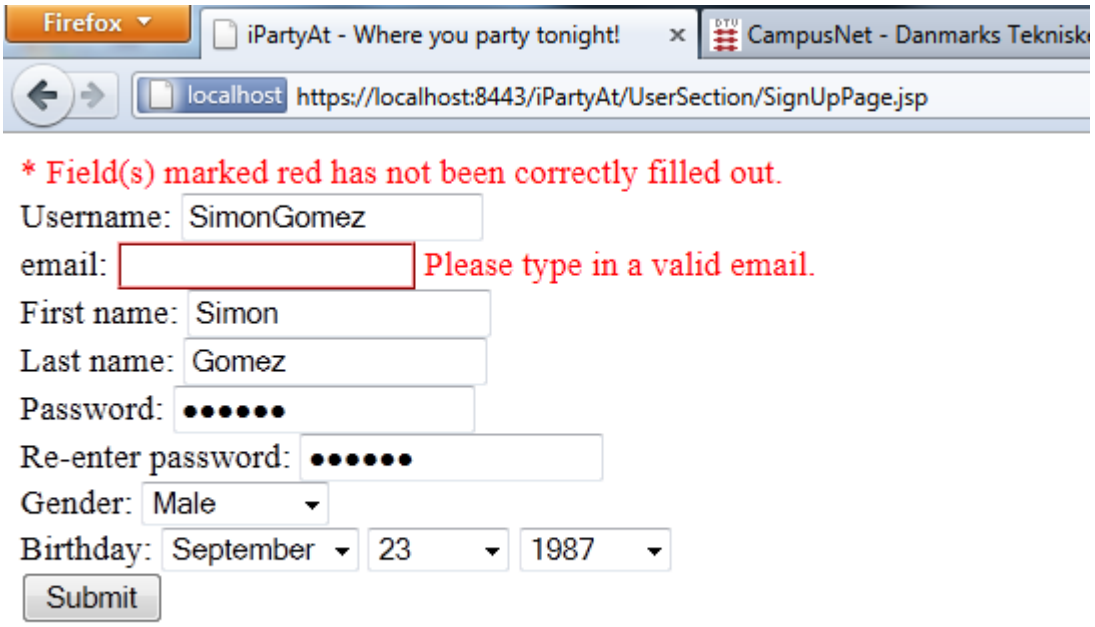


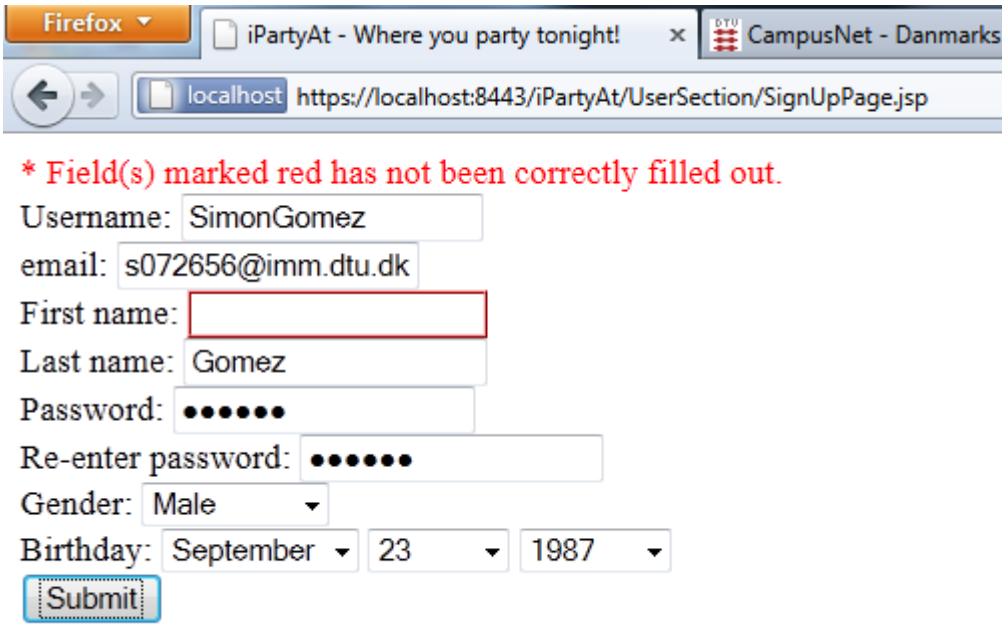
13.2.1.12 D. 3

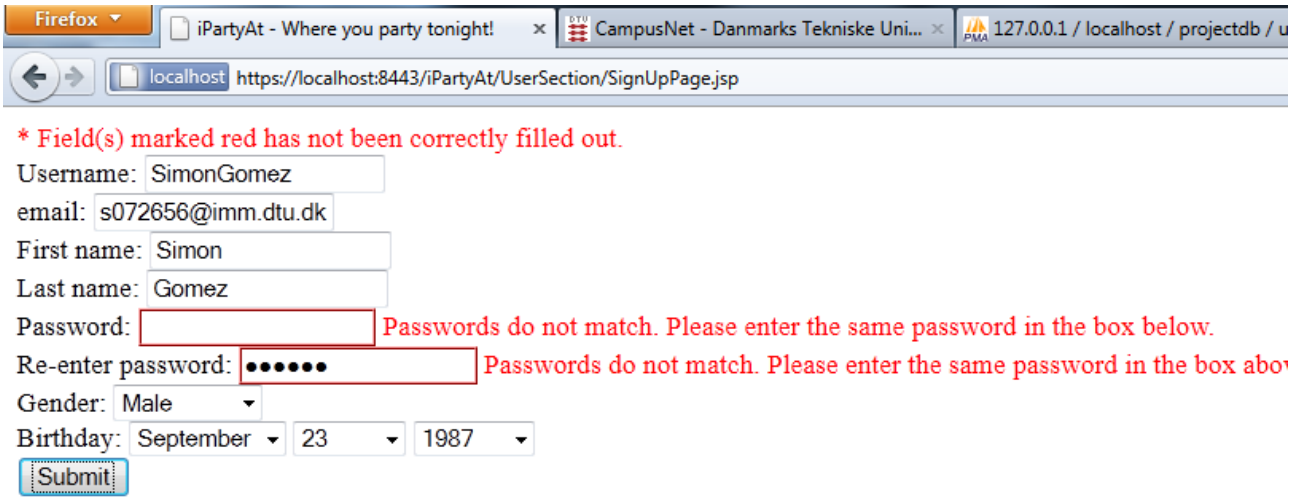


13.2.1.13 E.1

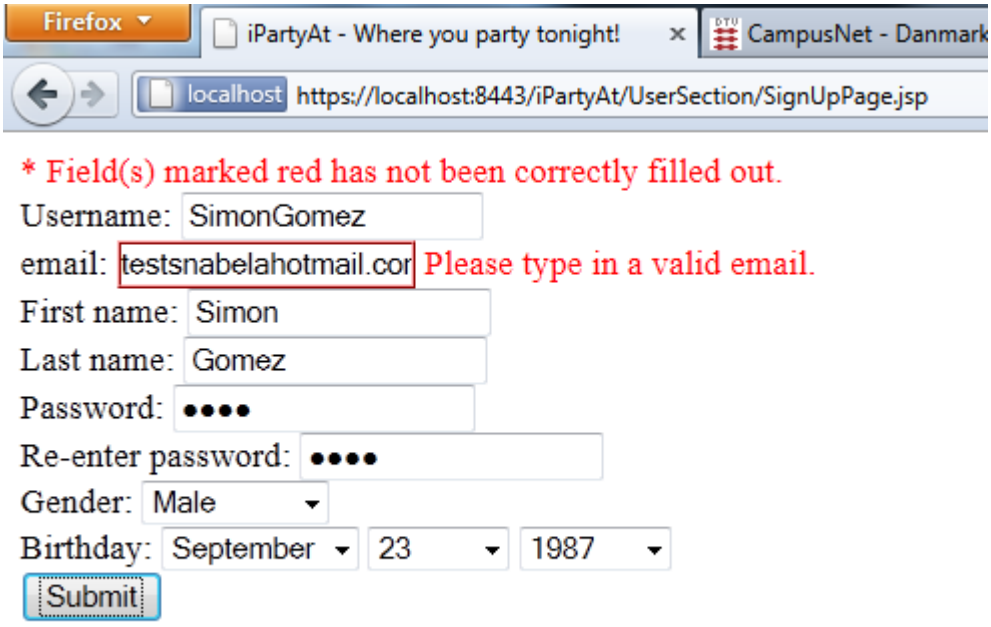



Firefox
iPartyAt - Where you party tonight!
CampusNet - Danmarks Teknisk
localhost
https://localhost:8443/iPartyAt/UserSection/SignUpPage.jsp
* Field(s) marked red has not been correctly filled out.
Username: SimonGomez
email:
Please type in a valid email.
First name: Simon
Last name: Gomez
Password: ••••••
Re-enter password: ••••••
Gender: Male
Birthday: September 23 1987
Submit

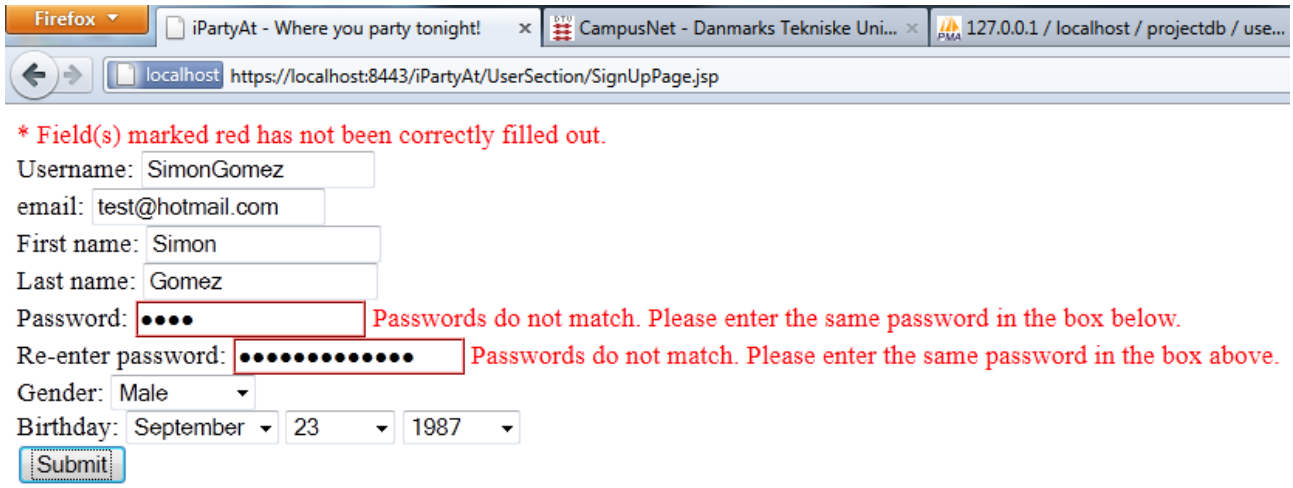

Firefox
iPartyAt - Where you party tonight!
CampusNet - Danmarks
localhost
https://localhost:8443/iPartyAt/UserSection/SignUpPage.jsp
* Field(s) marked red has not been correctly filled out.
Username: SimonGomez
email: s072656@imm.dtu.dk
First name:
Last name: Gomez
Password: ••••••
Re-enter password: ••••••
Gender: Male
Birthday: September 23 1987
Submit


Firefox
iPartyAt - Where you party tonight!
CampusNet - Danmarks Tekniske Uni...
127.0.0.1 / localhost / projectdb / u
localhost
https://localhost:8443/iPartyAt/UserSection/SignUpPage.jsp
* Field(s) marked red has not been correctly filled out.
Username: SimonGomez
email: s072656@imm.dtu.dk
First name: Simon
Last name: Gomez
Password:
Passwords do not match. Please enter the same password in the box below.
Re-enter password: ••••••
Passwords do not match. Please enter the same password in the box abo
Gender: Male
Birthday: September 23 1987
Submit

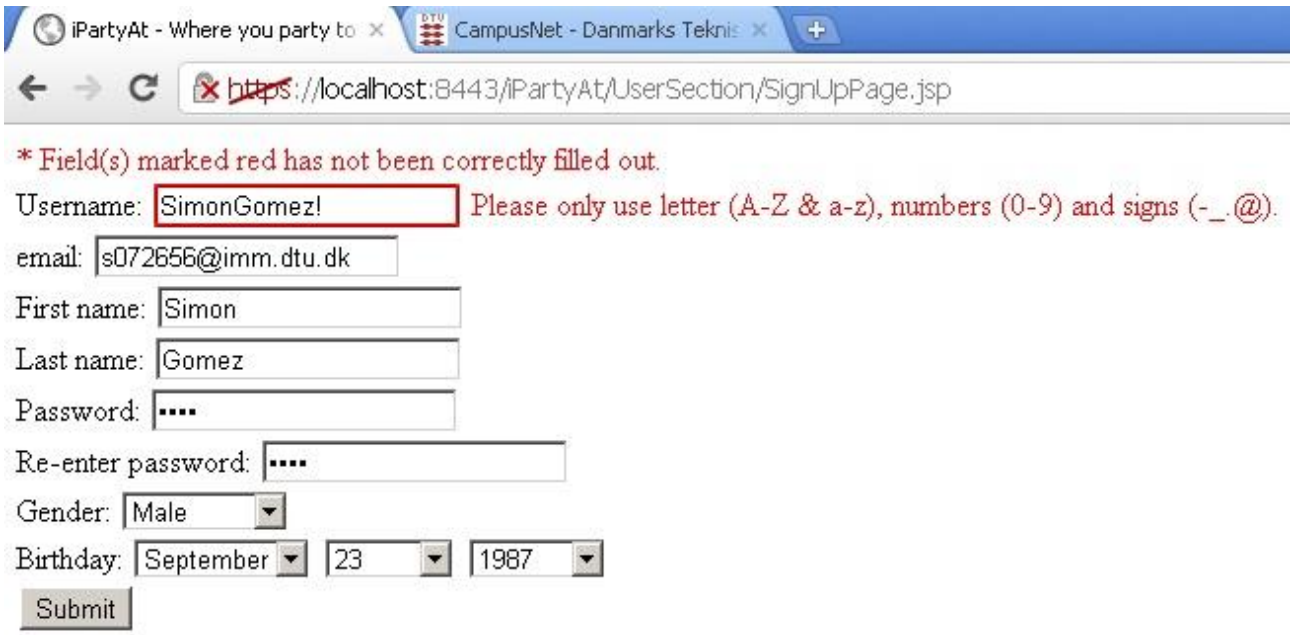
13.2.1.14 E. 2


Firefox
iPartyAt - Where you party tonight!
CampusNet - Danmark
localhost
https://localhost:8443/iPartyAt/UserSection/SignUpPage.jsp
* Field(s) marked red has not been correctly filled out.
Username: SimonGomez
email: testsnabelahotmail.cor
Please type in a valid email.
First name: Simon
Last name: Gomez
Password: ••••
Re-enter password: ••••
Gender: Male
Birthday: September 23 1987
Submit

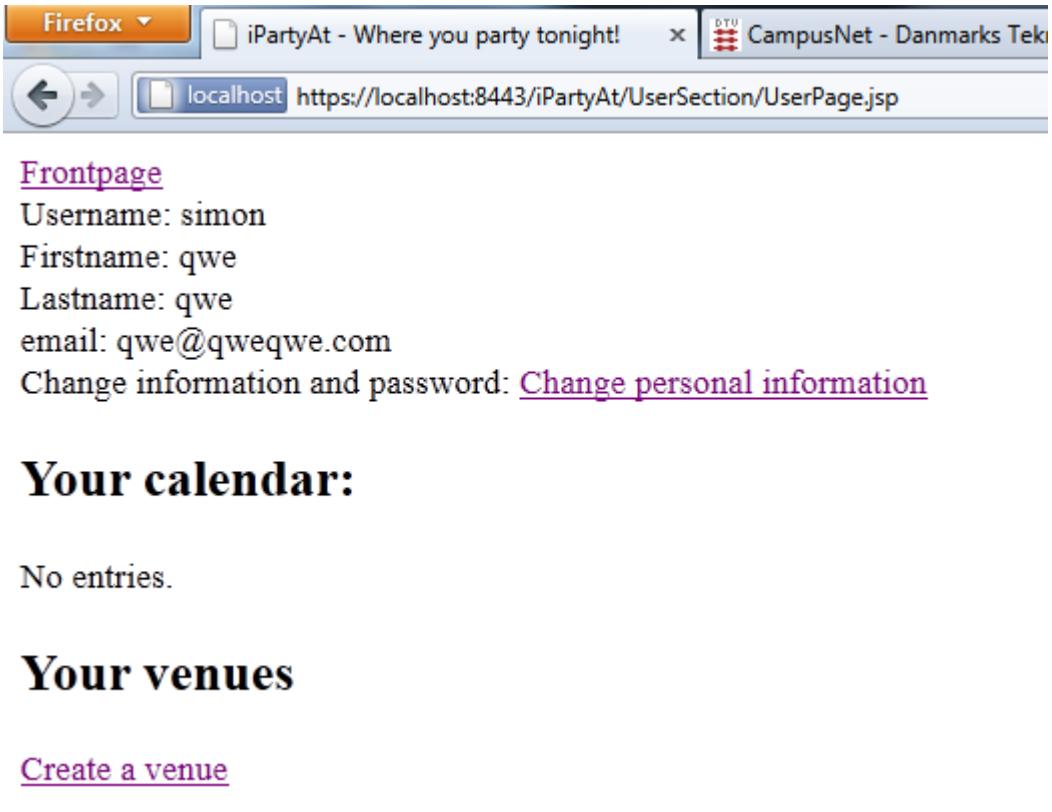
13.2.1.15 E. 3



13.2.1.16 E. 4

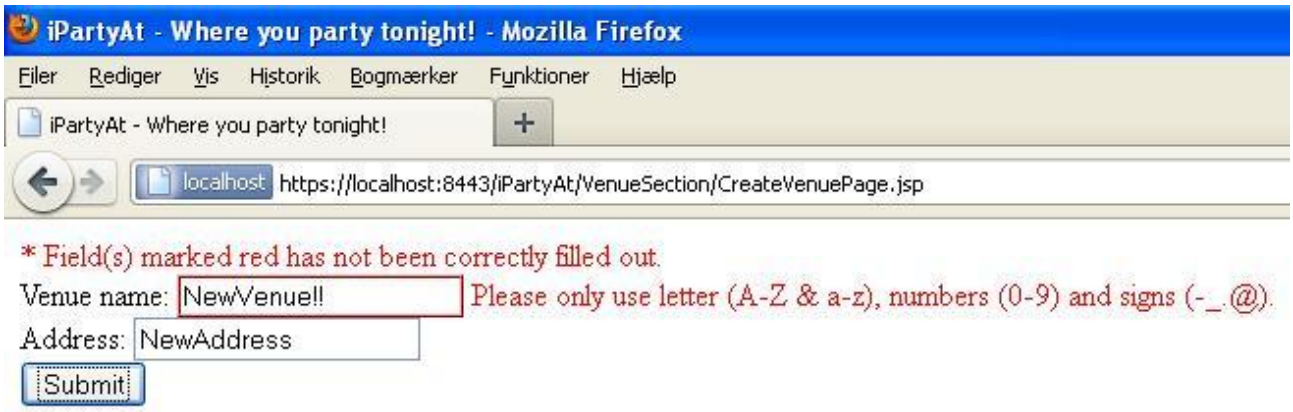


13.2.1.17 E.5



13.2.1.18 F. 1

13.2.1.19 F. 2



13.2.1.20 G. 1

13.2.1.21 G. 2

Name	Address	Rating	Number of ratings
cheVenue	cheVenue	5.0	1
venue1	venue1	4.0	1
venue2	venue2	0.0	0
venue3	venue3	3.0	1
venue4	venue4	0.0	0
venue5	venue5	2.0	1
venue6	venue6	1.0	1
megavenue	venuestreet	0.0	0

13.2.1.22 G. 3

Name	Address	Rating	Number of ratings
cheVenue	cheVenue	5.0	1
venue1	venue1	4.0	1
venue3	venue3	3.0	1
venue5	venue5	2.0	1
venue6	venue6	1.0	1
venue4	venue4	0.0	0
venue2	venue2	0.0	0
megavenue	venuestreet	0.0	0

13.2.1.23 H. 1


```
Connected to the database
Select * from user WHERE userName = 'che2' and password = 'che2';
```

13.2.1.24 H. 2


5554:my_avd
1:58 PM
MobiPartyAt
Venue Name
venue2
Address
venue2
Venue Rating
5.0 of 1 votes
Rate Venue
1 2 3 4 5
Calendar
MENU

13.2.1.25 H. 3

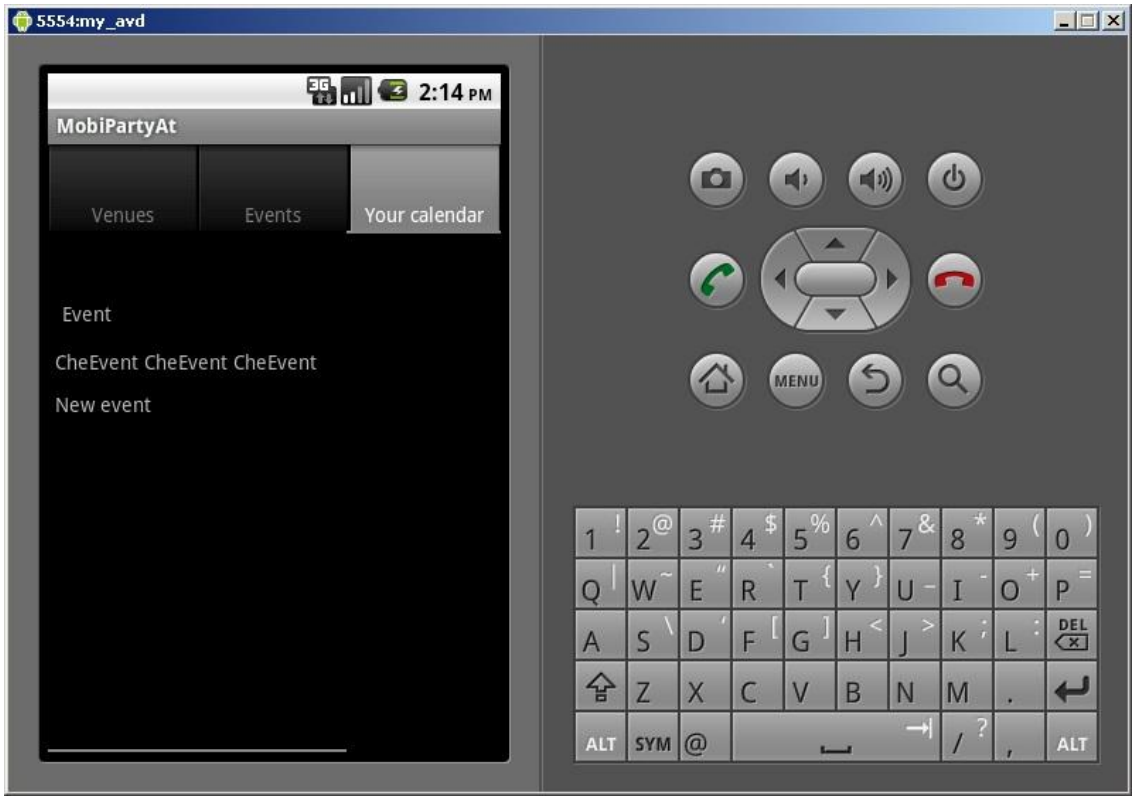

5554:my_avd
2:14 PM
MobiPartyAt
Venues
Events
Your calendar
Event
CheEvent CheEvent CheEvent
New event
MENU

13.2.1.26 H. 4


5554:my_avd
2:23 PM
MobiPartyAt
MobiPartyAt
Login
Browse
MENU

13.2.1.27 H. 5

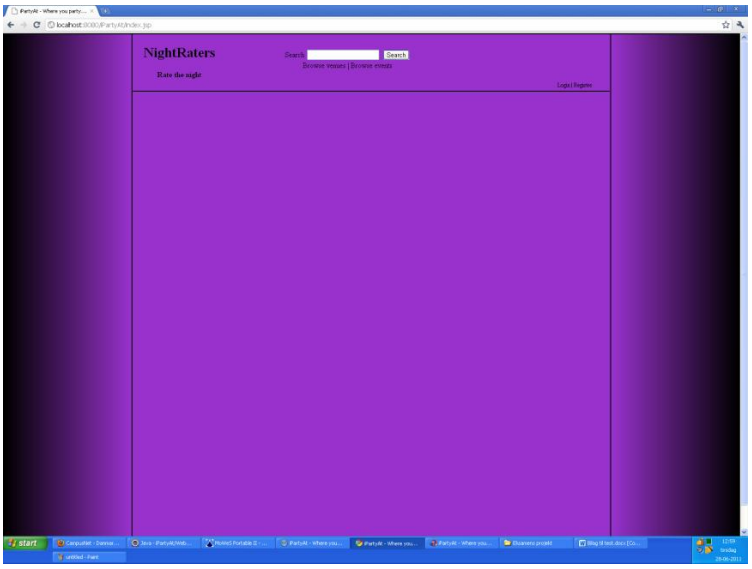
Før sign up



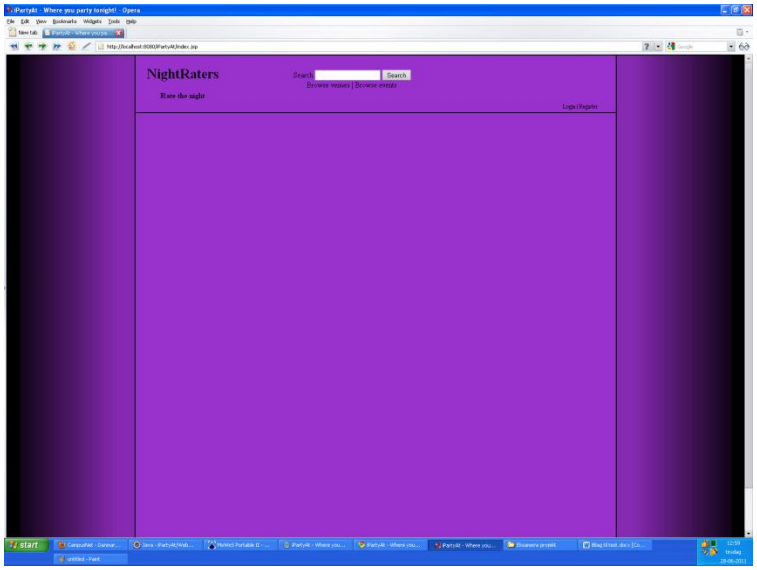
Efter sign up

13.2.2 Browser Test

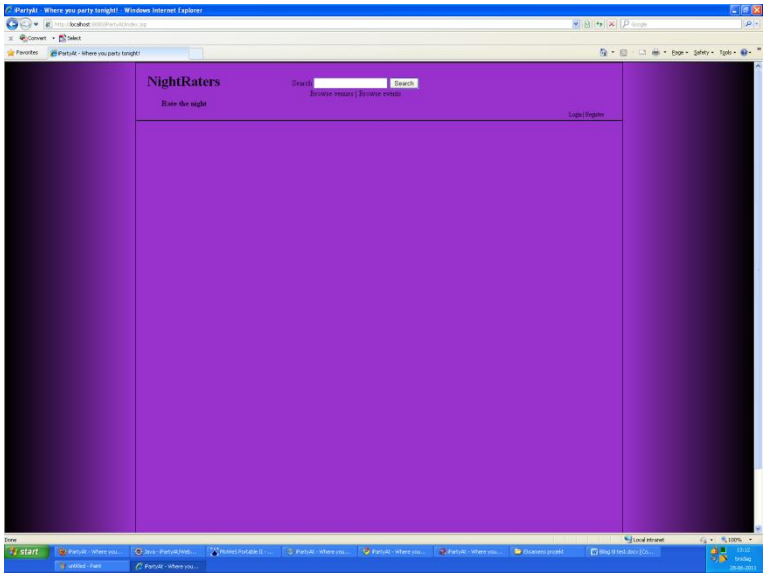
13.2.2.1 Crome



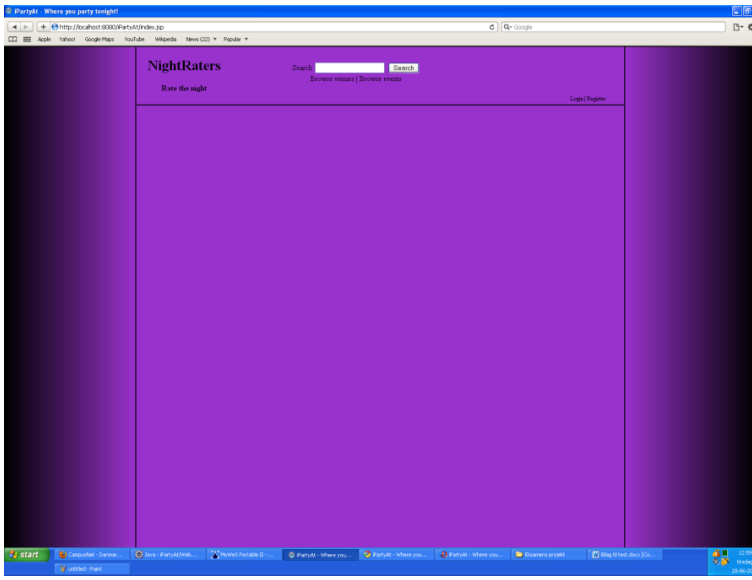
13.2.2.2 Opera



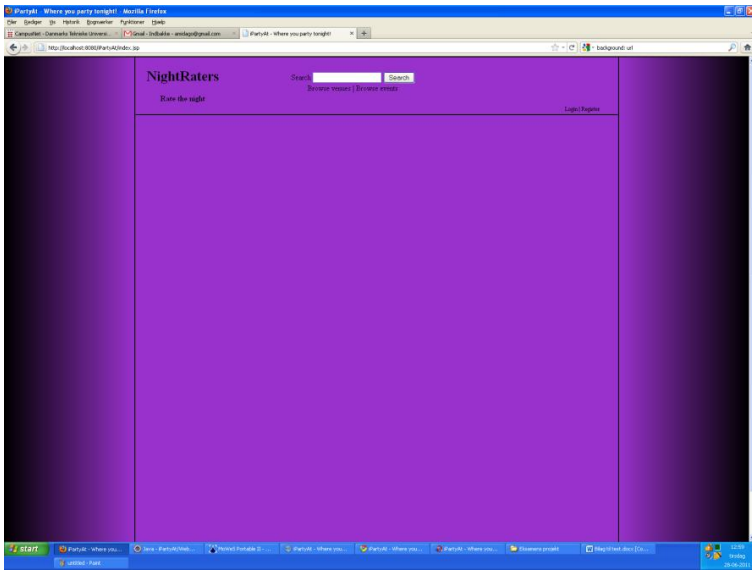
13.2.2.3 Internet explorer



13.2.2.4 Safari



13.2.2.5 Mozilla



13.3 Bilag til udskrift af XML

13.3.1 All Events XML udskrift

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
          <allcalentries xmlns="http://www.w3schools.com">
                    <venuecalentry>
                              <venuecalentryId>1</venuecalentryId>
                              <info>CheEvent CheEvent CheEvent </info>
                    </venuecalentry>
                    <venuecalentry>
                              <venuecalentryId>2</venuecalentryId>
                              <info>Test test test tesdt </info>
                    </venuecalentry>
                    <venuecalentry>
                              <venuecalentryId>3</venuecalentryId>
                              <info>hej hej hej </info>
                    </venuecalentry>
                    <venuecalentry>
                              <venuecalentryId>4</venuecalentryId>
                              <info>New event</info>
                    </venuecalentry>
                    <venuecalentry>
                              <venuecalentryId>5</venuecalentryId>
                              <info>EventMad</info>
                    </venuecalentry>
          </allcalentries>
```

13.3.2 All Venues XML udskrift

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
          <allvenues xmlns="http://www.w3schools.com">
                    <venuetype>
                              <venueId>5</venueId>
                              <venuename>cheVenue</venuename>
                              <venueaddress>cheVenue</venueaddress>
                              <venueavg>0.0</venueavg>
                              <venuecount>0</venuecount>
                    </venuetype>
                    <venuetype>
                              <venueId>6</venueId>
                              <venuename>megavenue</venuename>
                              <venueaddress>venuestreet</venueaddress>
                              <venueavg>0.0</venueavg>
                              <venuecount>0</venuecount>
                    </venuetype>
          </allvenues>
```

13.3.3 User XML udskrift

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
          <usertype xmlns="http://www.w3schools.com">
                    <userId>12</userId>
                    <username>che2</username>
                    <email>che2@che2.dk</email>
                    <firstname>che2</firstname>
                    <lastname>che2</lastname>
                    <usercalendartype>
                              <usercalendarId>1</usercalendarId>
                              <usercalentrytype>

          <usercalentryId>1</usercalentryId>

          <venuecalentryId>3</venuecalentryId>
                                        <info>hej hej hej </info>
                              </usercalentrytype>
                    </usercalendartype>
          </usertype>
```