Organic Donkey Hostel - Hjemmeside med booking
Projektkursus Systemudvikling 2015

3. juni 2015

Projektgruppe-nummer 1, gruppemedlemmer:

Jeppe Schönemann Skov 240992
Rose Sofie Greve 250493
Frederik Leed Henriksen 110394

Instruktor: Christoffer Wadum

Indhold

1 Abstract

We are making a website for a small Bed and Breakfast on Isla Margarita, Venezuela. The place is working towards becoming a sustainable and self-sufficient mini-hostel, where guests can enjoy organic greens from the garden. The frontpage on the website will contain a description of the place and the surrounding area. The website will contain a simple booking- and online payment system. The website will have a link with pictures of the houses and surrounding facilities, and a link with information on the different activities taking place at the hostel. With courtesy to the place working with ecology and sustainability, the layout of the website will be simple and inspired by the colors of nature. Further, we will make an administration part of the system, from where our costumer can see and edit the bookings. The administrator will also be able to edit the website photos and text descriptions.

2 IT-projektets formål og rammer

Her følger en beskrivelse af rammerne for vores IT-projekt, baseret på FACTOR-begrebet.

Functionality: Skal indeholde et booking- og betalingssystem, hvor administratoren har mulighed for at redigere i bookingerne. Samt redigere i de billeder og den information der fremvises på hjemmesiden.

Application domain: Skal administrere bookninger, herunder redigere og slette bookinger samt finde kontaktoplysninger på de personer der har oprettet en booking.

Conditions: Vi skal udvikle et simpelt system, da de der skal betjene systemet har ikke stor teknisk erfaring.

Technology: Systemet skal både udvikles og kunne betjenes på en billig pc med internet. Grundet hyppige strømsvigt skal databasen ligge på en webserver.

Objects: Personer der ønsker at booke værelse. Administrering af bookinger og websiden.

Responsibility: Systemet skal løse administrative problematikker i henhold til at holde styr på kontaktoplysninger af kunder, holde styr på hvilke værelser der er ledige, og hvornår disse er ledige. Systemet skal også reklamere for hostel'et.

3 Kravspecifikationer for IT-løsningen

3.1 a - Funktionelle og ikke funktionelle krav

Dette afsnit angiver de funktionelle og ikke funktionelle krav, samt de begrænsninger vi har til vores system.

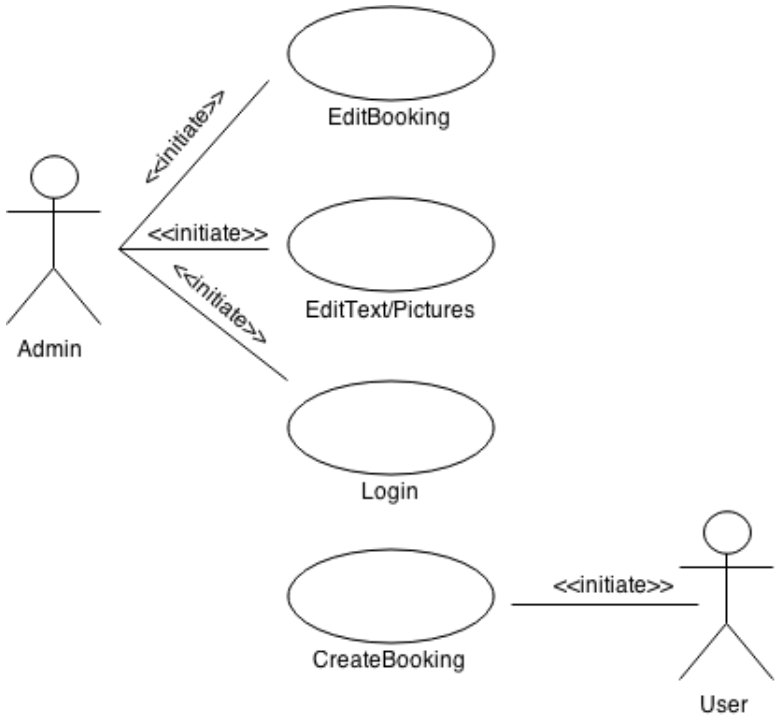
Funktionelle krav:
- Side med bookingsystem der giver brugeren mulighed for at booke senge
- Administrativ side der giver vores kunde mulighed for at redigere i bookinger, tekst og billeder på brugersiden
- Loginsystem til den administrative side
- Betalingssystem. Et eksternt system som kunden viderestilles til. Ikke et betalingssystem vi selv koder.

Ikke-funktionelle krav:
- Simpelt system som kan benyttes af folk uden særlig teknisk erfaring
- Side med information om stedet
- Side med billedegalleri
- Side med kontaktinformation
- Layout der afspejler hostelets stil og holdning til miljøet.

Begrænsninger:
Siden skal kunne vises på en billig computer og uden hurtigt internet til rådighed. Databasen med bookinger skal ligge på en webserver, da strømsvigt er hyppige i området.

3.2 b - Use case model

Dette afsnit beskriver kort det mest essentielle ved systemets use cases, og illustrerer sammenhængen mellem disse og aktørerne i et højniveau-diagram.



Figur 1: Et højniveau-diagram der illustrerer hvilke use cases de forskellige aktører har

EditBooking Administratoren kan slette og redigere i værelsesbookingerne. Login er krævet for at kunne redigere i bookinger.
EditText/Pictures Administratoren kan slette og redigere i teksten og billederne på hjemmesiden. Login er krævet for at kunne redigere i hjemmesiden.
Login Administartoren skal logge ind på Admin siden, for at se og redigere bookinger samt tekst og billeder.
CreateBooking Brugeren kan booke et værelse på hostelet gennem hjemmesiden.

3.3 c - Use cases

De følgende tre tabeller angiver mere detaljerede beskrivelser at vores tre vigtigeste use cases. Diagrammerne illusterer relationerne mellem aktørerne og systemet.

Tabel 1: Højniveau use-case-diagram over use casen Login

Use-case-name	Login
Participating actors	Administratoren
Flow of events	1. Administratoren indtaster sit brugernavn og kodeord på admin log-in siden og trykker på log-in knappen 2. Admin-databasen bliver spurgt om de indsendte oplysninger er rigtige. 3. Hvis der findes et match, bliver man videreført til admin-siden. Hvis ikke får man en fejlmeddelelse.
Entry conditions	At man ikke allerede er logget ind
Exit conditions	At man er logget ind

Tabel 2: Højniveau use-case-diagram over use casen EditBooking

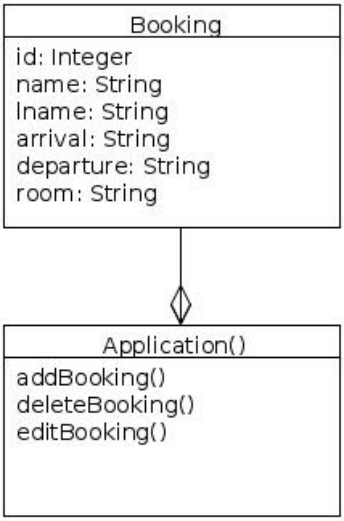
Use-case-name	EditBooking
Participating actors	Administratoren
Flow of events	1. Administratoren indtaster den ønskede ændring for en booking 2. Ændringen bliver sendt til databasen og bookingens detaljer redigeres 3. Tabellen opdateres med de nye oplysninger
Entry conditions	At man er logget ind
Exit conditions	At den ønskede ændring i bookingen er sket

Tabel 3: Højniveau use-case-diagram over use casen CreateBooking

Use-case-name	CreateBooking
Participating actors	Brugeren
Flow of events	1. Kunden indtaster sine ønskede booking datoer og kontaktinformationer 2. Kunden trykker på booking-knappen og bookinginformationerne bliver tjekket for at verificere at bookingen ikke overlapper med andre bookinger 3. Hvis datoerne for bookingen godkendes, bliver bookingen oprettet, og kunden bliver videresendt til "home"siden. Hvis der er overlap får kunden en fejlmeddelelse.
Entry conditions	At brugeren befinder sig på bookingsiden
Exit conditions	At brugerens booking gemmes i databasen

3.4 d - Klassediagram over problemområdet

Dette afsnit illustrerer vores problemområder i et klassediagram.
Figur 2 illustrerer et klassediagram over problemområdet bookingsystem. Application() tilføjer, redigerer og sletter bookinger fra bookingdatabasen. Booking definerer hvilke parametre en booking skal indeholde.



Figur 2: UML klassediagram over bookingsystemet

3.5 e - BCE-model

Dette afsnit indeholder en BCE-model over vores system.

Tabel 4: Boundary

Admin log-in side	Web-siden for administratoren som benyttes til at verificere at personen er administrator
Kunde booking web-side	Her har kunden adgang til at oprette en booking samt navigering til de andre kunde sider
Kunde "home"web-side	Her kan kunden navigere til de andre sider
Kunde galleri web-side	Her kan kunden navigere til de andre sider
Kunde kontakt web-side	Her kan kunden navigere til de andre sider
Kunde aktiviteter web-side	Her kan kunden navigere til de andre sider

Tabel 5: Control

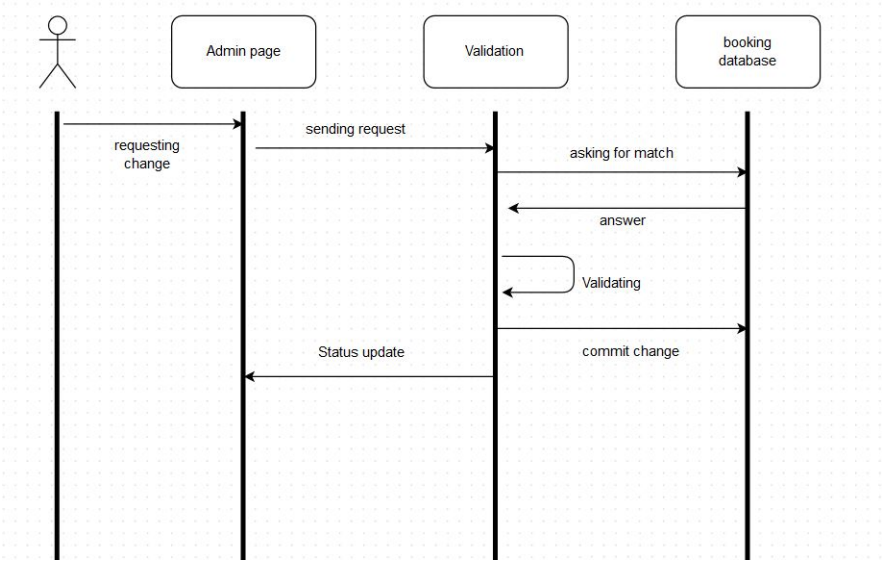
Admin log-in verificeringsknappen	Facilliterer forespørgelsen mellem admin log-in siden og admin-databasen. Hvis personen er en admin, bliver vedkommende sendt videre til admin siden. Hvis ikke, sker der ikke noget
Opret booking knap	Søger databasen for overlap i dato. Hvis der ikke er sådanne konflikter skabes en booking med de givne informationer i databasen. Hvis der er en konflikt vises en passende fejlmeddelelse
Rediger booking knap	Redigerer i en allerede oprettet booking i databasen
Rediger web-sider	Redigerer kundesidernes indhold i forhold til billeder og tekst (Vi ved endnu ikke hvordan dette skal implementeres)
"Home"link	Sender kunden til "home"web-page
Booking link	Sender kunden til booking web-page
Galleri link	Sender kunden til galleri web-page
Kontakt link	Sender kunden til kontakt web-page
Aktiviteter link	Sender kunden til aktiviteter web-page

Tabel 6: Entity

Admin databasen	Indeholder oplysninger om hvilke brugere der eksisterer i systemet
Administrator siden	Her har administrator mulighed for at tilse aktuelle bookninger og rette i databasen
Booking database	Indeholder alle bookinger på siden

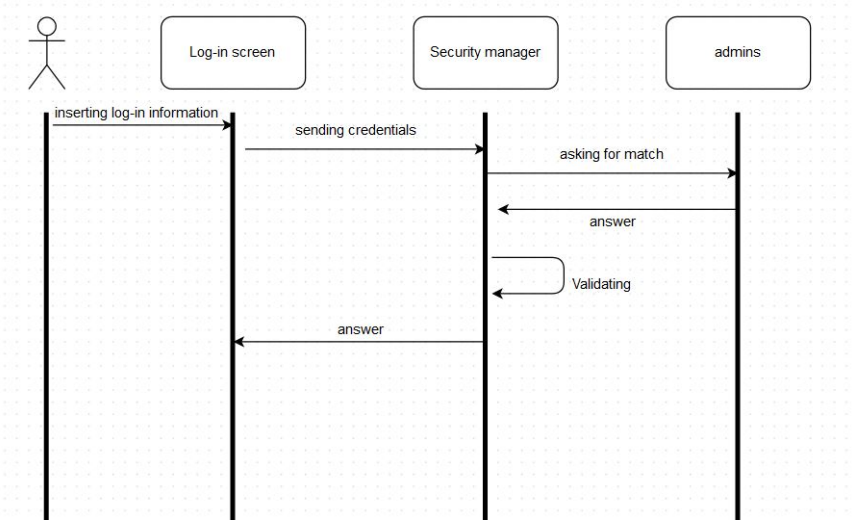
3.6 f - Sekvensdiagrammer over use cases

De følgende sekvensdiagrammer illusterer interaktionen mellem aktøren og systemet i en use case.



Figur 3: Sekvens-diagram over use-casen EditBooking

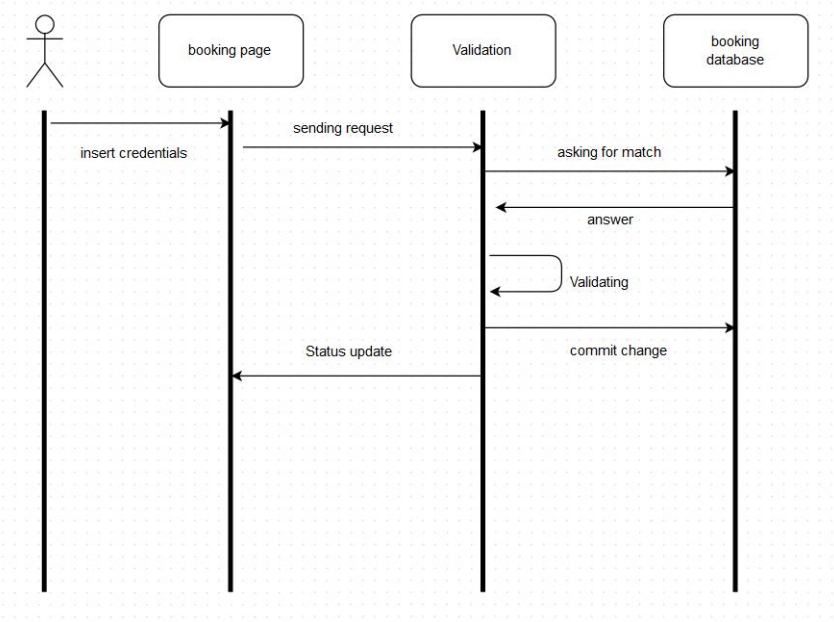
Figur 3: Brugeren forespørger om en ændring fra admin siden. Fra admin siden sendes en forespørgsel til en funktion, der validerer denne. Funktionen sender en forespørgsel til databasen, som svarer. Hvis valideringsfunktionen evaluerer til "sand", bliver ændringen i databasen gennemført. Hvis ikke bliver det skridt sprunget over. Man får en statusopdatering, der beskriver, om ændringen bliver udført eller ej.



Figur 4: Sekvens-diagram over use-casen Login

Figur 4: Brugeren indtaster sine informationer på login skærmen. Informationerne sendes til en sikkerhedsvalideringsfunktion. Funktionen forespørger om et match i admin-databasen. Denne sender

svar tilbage. Sikkerhedsvalideringsfunktionen validerer forespørgelsen. Svaret bliver sendt til login skærmen.



Figur 5: Sekvens-diagram over use-casen CreateBooking

Figur 5: Brugeren indtaster information på bookingsiden. Informationen bliver sendt til en valideringsfunktion. Denne forespørger, om der er konflikter i databasen. Databasen svarer, og valideringsfunktionen validerer forespørgelsen. Status på forespørgsel vises på bookingsiden.

4 Systemdesign sammenfatning

Systemet som det er nu: Vi har benyttet Play! frameworket til at lave en meget fin prototype af systemet. Det er nu muligt at navigere mellem alle de forskellige sider på hjemmesiden. Vi har fået implementeret en billede-slider på siden *Gallery*, som præsenterer billeder af stedet i et diasshow. På siden *Contact* har vi implementeret et google-maps kort, der viser hostelets beliggenhed. Bookingsystemet er også blevet konstrueret, og det er nu muligt at tilføje en booking fra bruger-siden, samt tilføje og slette bookinger fra admin-siden. Bookingerne gemmes i en database.

De væsentligste mangler: Vi mangler en funktion, der gør det muligt for administratoren at ændre en booking. Desuden mangler vi stadig at lave et login-system til admin-siden.
Vi skal også gøre det muligt for kunden, selv at bestemme hvilke billeder der fremvises samt redigere i teksten på de forskellige sider. Endeligt mangler vi at koble bookingsystemet til et betalingssystem.

5 Program- og systemtest

Resultatet af vores testplan er vedlagt som bilag.

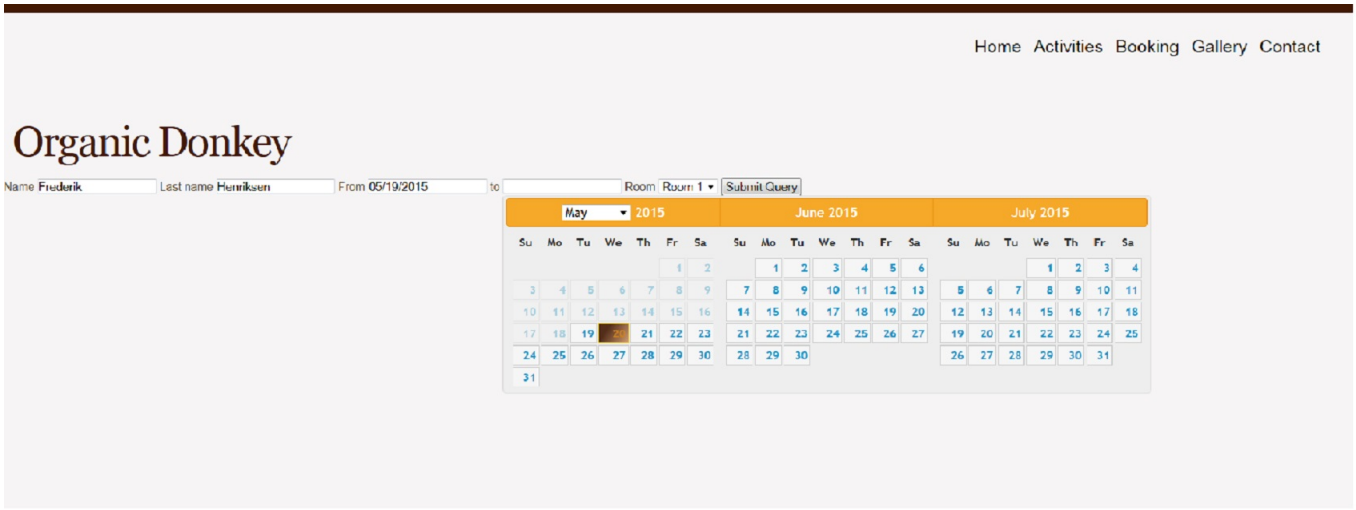
6 Brugergrænseflade og interaktonsdesign

Dette afsnit omhandler brugergrænsefladen og interaktionendesignet, for vores system.

6.1 Eksempler på brugergrænseflade

Følgende billeder viser brugergrænsefladen på hjemmesiden.

I figur 6 ses vores bookingside hvorfra man kan oprette en booking. Det forløbige design er således, at der udover den oblikatoriske menu-bar også er fire tekstfelter, en "drop-down"menu og en knap. Disse felter skal indsamle det data, der bliver til en oprettet booking (Indholdet af en booking er ikke færdigt endnu, der vil altså være andre/anderledes parametre i slutproduktet).

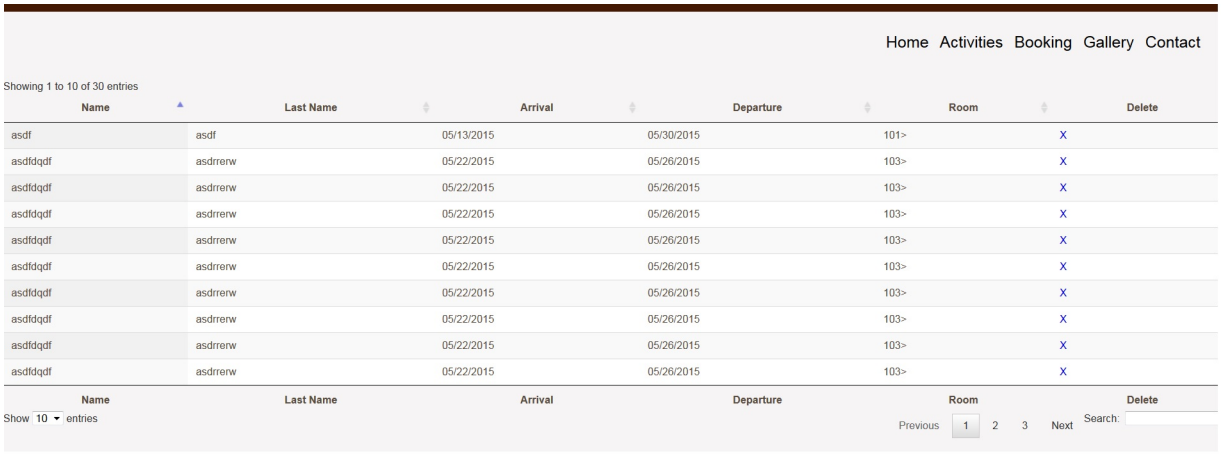


Figur 6: Opretter en booking. Her ses en bookning der er ved at blive lavet, hvor brugeren er ved at vælge dato

I figur 7 ses det layout som går igen på de fleste sider med overskrift, menu-bar og tekst.

Figur 7: Index siden. Her ses vores sides forside

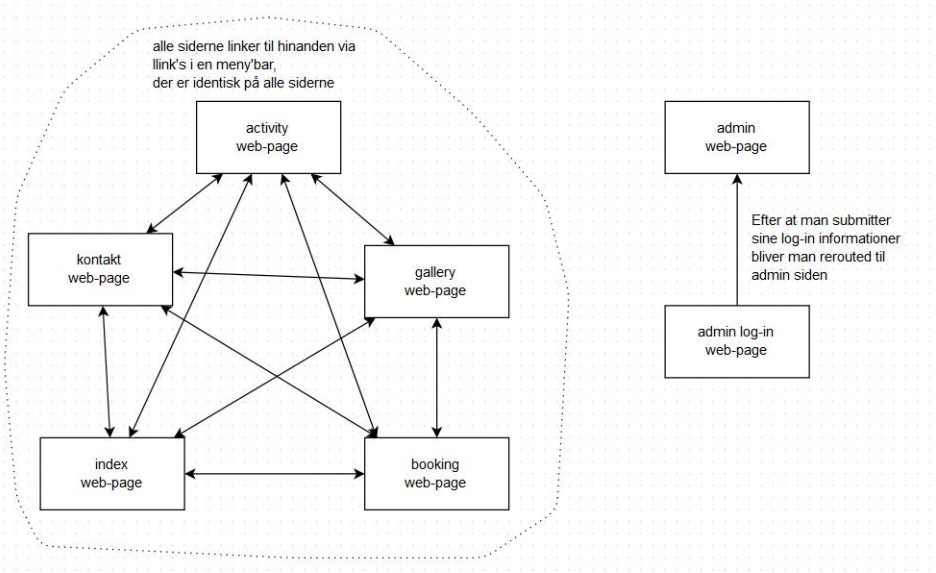
I figur 8 ses den forløbige admin side hvor administrator har overblikket over bookinger. Administratoren kan søge og slette i bookinger.



Figur 8: Oprettede bookinger. Her ses admin-siden hvor administratoren kan se og redigere bookinger

6.2 Dynamikken i brugerinteraktion

I figur 9 ses rutediagrammet over, hvordan brugere af siden kan bevæge sig imellem de forskellige views. Kunden betjener siderne gennem menuen. Administratoren skal logge ind for at administrere bookingerne.



Figur 9: Flowchart over siden. Her ses rutediagrammet over siden, som den fungerer på nuværende tidspunkt

6.3 Audio-visual præsentation

Følg linket for at se en audio-visual præsentation af brugergrænsefladen af vores seneste prototype:

`https://youtu.be/_KMsyZdtMHM`

7 Projektsamarbejdet

Internt i gruppen arbejder vi godt sammen. Cirka halvvejs inde i projektet aftalte vi at sætte mere tid af til projektet. Siden da er vi nået rigtig langt med systemet.

Vi har indtil videre haft fire møder med kunden. Møderne foregår over Skype, da kunden bor i Venezuela. På trods af at afstanden kan være upraktisk, er kunden fleksibel med dato for møder, da vi blot mødes over Skype.

Under møderne tager vi referat i form af stikord, og dokumenterer de aftalte ændringer i projektaftalen og projektplanen, i rapporten.

8 Litteraturreview

8.1 Jim Highsmith: Extreme Programming - review

Extreme Programming er en software udviklings metode, der er karakteriseret ved at være åben for tilpasninger løbende i udviklingsprocessen, som har til hensigt at gøre kvaliteten og reagerer på ændringer i kundes krav. Andre elementer ved extreme programming inkluderer, parprogrammering eller lav en stor kode gennemgang, unit testing af alt koden, og undgå at programmere features før de faktisk skal benyttes og forvent ændringer i kundens krav som tiden går.

Metoden har fire hovedprincipper

- Kommunikation
  - Brugere og udviklere skal kommunikerer løbende om systemet, så brugeren på en måde kan være med i udviklings processen.
- Simpelhed
  - Software udvikleren skal kun skrive kode der løser det problem de har her og nu og vil ikke løse problemer der muligvis kan opstå senere.
- Feedback
  - Der skal laves test, for at udvikleren kan se om det der er lavet virker som antaget.
- Mod
  - Det kræver mod at lade vær med at bruge de traditionelle udviklings metoder.

Extreme programming opererer med tolv punkter, der beskriver måden at arbejde i et extreme programming projekt i praksis.

- Planlægning
  - Kort 2-3 ugers cyklus, mange udgivelser.
- Små udgivelser
  - Alle udgivelser skal være så små som mulige, kun indeholde de mest værdifulde krav.
- Systemmetafor
  - Extreme programming bruger ordet metafor som en måde til at definere det overordnet mål med projektet.
- Simpelt design
  - Det simple design har to dele, et design for funktionaliteten. To, lave det bedste design der kan give den funktionalitet.
- Hyppige refaktorering
  - Designet bliver hyppigt lavet om da kundens krav kan ændre sig.
- Test
- Parprogrammering
- Fælles ejerskab til programkoden
- Kontiuerlig integration
- Overkommeligt arbejdstempo
- Et samlet udviklingshold
- Fælles kodestandard

I vores projekt har vi lavet lidt af en kombination af extreme programming og den mere traditionelle måde at gribe et system udviklings projekt an. Blandt andet har vi ikke implementeret ting i vores projekt før vi faktisk skulle bruge dem, vi tager en ting af gangen og prøver ikke tænke for langt fremad og kun programmere så tingene kun gør det de skal, uden nogle ekstra features. På den mere traditionelle måde har vi holdt os stramt til vores kunde specifikationer.

8.2 Review af naurProgramming as theory building:

ariklen bemærker at man kan drage fordel af at have det samme team af programmøre til at forbedre/ædre i koden til programmet, samt at benyttelse af de samme programmøre frem for nye, gør det bedre for tiden/resultatet.

Enhver situation er unikqe. Man kan ikke standardisere disse situationer og så bliver man altså nød til at tage dem som de kommer, da man ikke kan lave nogle teoretisk standarter.

Det er kun programmøren der har produceret projektet som har styr på hvilke hvilke dele gør hvad og hvorfor det gav mening at gøre dette.

Der er mange muligheder for at ændre et programs struktur, men kun den originale programmør kan læse det med det "mindset" som får det hele til at give mening. Dette er fordi at der eksistere mange forskellige måder at opnå det samme resultat, og det kun er ham der programmeret det som har den tankegang.

Et program som ikke holdes i live, altså ikke bliver opdateret mere, men som stadig har nyttige aspekter, vil måske blive rekonstrueret på et givent tidspunkt. Dette kan tage tid, hvis det ikke er de originale programmøre der udføre dette, og de nye programmøre vil være splittet mellem at være loyale over for det originale program og forbedre på dettes svagheder. Det bliver altså nemt en omkostningsfund process.

Behold programmørnede på teamet, for at fremme effektiviteten. Det er ikke særligt effektivt at skifte programmøre ud under vejs i en process.

Vores projekt i forhold til teksten: teksten handler jo i ultra korte træk om at det at bevare sit faste team af programmøre igennem hele processen kan betale sig, da de har den samlede forståelse for problemområdet, samt den komplette forståelse af ideerne og planerne med de funktionaliteter som er implementeret. Dette har altså ikke været et problem for os, da vi lavede et projekt fra bunden og ikke skulle bygge videre på det andre havde udviklet før os. På en måde er det ikke sandt, da vi igennem vores valg af framework til vores side, alligevel bygger på noget som andre har lavet, og derigennem har skulle sætte os ind i andres tankegang om hvorledes man skulle gå til den problemstilling at designe en hjemmeside.

9 Bilag

9.1 Bilag 1: Versionsstyring

Link til vores commit-log på GitHub:
https://github.com/fumpen/pksu/commits/master

9.2 Bilag 2: Changelog for projektrapporten

Jeppe Schönemann Skov - JSS
Frederik Leed Henriksen - FLH
Rose Sofie Greve - RSG

Initialer	Dato	Afsnit	Ændring
JSS, FLH, RSG	23.04.15	Alle	Release edition 1.0
RSG	07.05.15	1, 2	Stavefejl, mere præcis beskrivelse
RSG	07.05.15	3.a	Krav i stikord, mere uddybende
FLH	07.05.15	3.b, 3.c, 3.e	Omskrivning og ændring i model
JSS	07.05.15	3.d	Opdateret klassediagramet
RSG, JSS	07.05.15	3.f	Kommentarer til firgurer
RSG, JSS	07.05.15	6, 7	Ændring i tekst
RSG	11.05.15	4, 5, 7	Ændring i tekst
FLH	11.05.15	6	Flowchart, skærmbilleder, video
RSG	21.05.15	3.1 a	Ændring i tekst
RSG	27.05.15	3.1 b	Nyt diagram
RSG	27.05.15	3.1 c	Ændring af entry og exit conditions
FLH	27.05.15	3.1 f	Beskrivende tekst til diagrammerne
FLH	27.05.15	9.3 Timeline	Beskrivende tekst til diagrammerne
JSS	02.06.15	9.4 Testplan	Beskrivende tekst til diagrammerne
JSS, FLH, RSG	02.06.15	Udvidet prototype klar	Beskrivende tekst til diagrammerne

9.3 Bilag 3: Timeline

-Under første møde d. 19. februar 2015 fik vi lavet en projektaftale med kunden.
-Under andet møde d. 10. marts 2015 snakkede vi detaljer ift. bookingsystemet.
-Under tredje møde d. 15. april 2015 snakkede vi om valg af betalingssystem, samt om hvilke dele af projektet kunden helst så os nedprioritere i tilfælde af tidspres.
-Den 11 maj 2015 havde vi den første prototype. Denne var i stand til at oprette bookinger, slette bookinger, sortere bookninger og den havde næste alle de sider som der skal være i det endelige produkt. -Den 2 juni havde vi en udvidet prototype klar.

9.4 Bilag 4: Test plan

Vi har ikke til hensigt at tjekke alle typer input eller tjekke sikkerheden af vores kode.

Systemoversigt Vores program er delt op efter Model-View-Control design mønstret.

Test og mangler Det er primært vores Views der er blevet testet. De eneste tjek, vi har lavet på selve systemet eller databasen, er, om bookingerne bliver lagt over i databasen, når de er blevet submittet, og at de kan blive slettet af administratoren hvis det er ønsket. Samt tjek af login for administratoren.

Hardware og software krav

- Styresystem der kører OSX, Linux eller Windows
- Java
- Play Framework
- Sqlite
- Testet på Chrome og Firefox

Prøve plan

- Tilføj booking
- Slet booking
- Admin login
- Admin tilføj / fjern billeder
- Admin ændre tekst

Vi laver vores test løbende med at de bliver implementeret, så vi har mulighed for at rette systemet til, hvis det skulle påvirke noget af det resterende.

9.4.1 Test plan specification

Admin:

Login system

Test item

1. Login med forkert kode

Output

1. Melder en fejl

Alle brugere:

Tilføj booking

Test item

1. Tilføj en booking der starter i fortiden
2. Tilføj en booking der slutter i fortiden
3. Tilføj en booking på et værelse der allerede er fuldt booket

Output

1. Melder en fejl og beder kunden om at ændre tiden
2. Melder en fejl og beder kunden om at ændre tiden
3. Melder en fejl og beder kunden om at vælge et andet værelse

Admin:

Slet booking

Test item

1. Slet en booking

Output

1. Beder admin om acceptere eller cancel ønske om at fjerne bookingen

Admin:

Ændre gallery

Test item

1. Tilføj gyldigt billede
2. Tilføj ugyldigt billede
3. Slet billede

Input specifikation

1. Jpg, png format
2. Forkert format

Output

1. Beder admin accept / cancel
2. Melder fejl og beder admin tjekke om billedet er gyldigt
3. Beder admin accept / cancel

Admin:

Ændre tekst

Test item

1. Tilføj tekst
2. Slet tekst

Output

1. Beder admin accept / cancel
2. Beder admin accept / cancel