Mobil-applikation til administration af webhosting-platform - Surftown
Projektkursus Systemudvikling 2014

Christian Enevoldsen 020691

19. juni 2014

Gruppe: Simon Warg, Nicklas Warming, Robert Rasmusen
AV-præsentation: http://youtu.be/coaXCn-iWJE

Indhold

Indledning

Denne rapport er skrevet som dokumentation for den praktiske del af projektkurset "Systemudvikling". Der redegøres for metoder og valg foretaget under og før udvikling. Til at starte med redegøres for formålet og rammerne for projektet, hvor den analysemodel, som er brugt i projektet beskrives. Dernæst fokuseres der på kravspecifikation og begrænsninger, efterfulgt af Use cases og Systemdesign. Næst redegøres der for programtest, user tests, brugergrænsefladen og workflow og afsluttes med en gennemgang af implementeringen og udviklingsprocessen. Til sidst i rapporten er der skrevet et review af artiklen *"Challenges of migrating to Agile methodologies"*, af Shridhar Nerur, Radhakanta Mahapatra og George Mangalara.

1 IT-Projektet

1.1 Formål og rammer

Det er valgt at bruge "FACTOR" [1] analysen til at specificere rammerne og problemområdet for projektet. FACTOR analysen består af de seks nedenstående punkter:

1. **F**: Beskrivelse af systemfunktionerne til løsning af problemområdet.
2. **A**: Beskrivelse af de aktøre, som er indehavere af problemområdet.
3. **C**: Beskrivelse af forhold, som systemet bliver udviklet under, samt de forhold systemet skal bruge.
4. **T**: Beskrivelse af teknologier som systemet bliver udviklet med, samt det systemet i sidste ende vil køre med og kræve.
5. **O**: Beskrivelse af hovedobjekterne som problemområdet indeholder.
6. **R**: Beskrivelse af ansvarsområdet for systemet.

F

Mobil adgang til udvalgte funktioner, samt Surftowns offentlige info. F.eks. kan en kunde se om en server er nede, og om det påvirker kundens egne services.

A

Surftowns kunder, der mobilt får mulighed for at tjekke status på servere, samt se status på deres services.

C

Brugerne skal have IT-kendskab, men ikke nødvendigvis være professionelle eller superbrugere. Surftown har ønsket applikationer til iOS og Android. Udviklingsarbejdet foregår ikke på fuld tid, eftersom det er et studieprojekt.

T

Android applikationen vil blive udviklet med Googles Android SDK, med deres tilhørende Eclipse plugin. iOS applikationen vil blive udviklet med Apple's Xcode. Udviklingsgrupperne har hver især resourcer til kørselstest under udviklingen.

O

Objekterne for problemområdet er beskrevet i sektion 1.2

R

Administrativ værktøj for Surftowns kunder og kommunikationsmiddel mellem Surftown og deres kunder.

1.2 Kravspecifikation

Sammen med Surftown er der besluttet funktionelle og ikke-funktionelle krav, samt begrænsninger til applikationen. Disse krav er vigtige for brugeren, samt specificerede til brug på smartphones. Applikation skal udvikles nativt til iOS og android. Dette er et valg foretaget med Surftown, fordi applikationen skal være hurtig. F.eks skal brugeren ikke vente 2 sekunder på, at det første view bliver hentet, hvilket ville være tilfældet, hvis vi hentede hjemmesiden ned og viste den tilpasset til smartphonen.

Funktionelle krav i prioriteret rækkefølge

1. Surftown kontaktoplysninger.
    - Et simpelt skærmbillede hvor brugeren kan se Surftowns kontaktoplysninger.
2. En login funktion.
    - En funktion der giver brugeren mulighed for at logge ind med e-mail og password.
3. Driftstatus af brugerens webhosting.
    - En informativ funktion hvor brugeren kan se opdateringer omkring tekniske fejl hos Surftown, som påvirker brugerens webhosting.
4. Driftstatus af brugerens E-mail hosting.
    - En informativ funktion hvor brugeren kan se opdateringer omkring tekniske fejl hos Surftown, som påvirker brugerens email.
5. Driftstatus af brugerens database hosting.
    - En informativ funktion hvor brugeren kan se opdateringer omkring tekniske fejl hos Surftown, som påvirker brugerens database hosting.
6. Driftstatus af brugerens webmail.
    - En informativ funktion hvor brugeren kan se opdateringer omkring tekniske fejl hos Surftown, som påvirker brugerens webmails.
7. Oversigt over brugt/ledigt plads på brugerens hostings.
    - En informativ funktion hvor brugeren kan se hvor meget ledigt plads, der er tilbage på brugerens forskellige hostingsservices.
8. Status af brugerens domæner.
    - En informativ funktion hvor brugeren har mulighed for at se en oversigt over alle brugerens domæner, som er tilknyttet Surftown, samt relevant information om domænerne.

9. OS type (Windows/Linux) for brugerens webhostings.
    - En informativ funktion hvor brugeren kan se hvilke operativsystem, som brugerens hostingservices køre.
10. En guide til at sætte Surftowns e-mail oplysninger op på en smartphone.
    - En simpel side hvor der er en guide i tekst- og billedeform, som beskriver hvordan man sætte en Surftown E-mail konto op på en Android/iPhone telefon.
11. Mulighed for at nulstille sin kode.
    - En funktion hvor brugeren kan skrive sin E-mail ind for at få tilsendt et nyt password.
12. En oversigt over brugerens fakturaer.
    - En funktion hvor brugeren kan se en oversigt over de betalte og udestående fakturaer, som brugeren har hos Surftown.
13. Udløbningsdatoer for brugerens domæner.
    (a) En funktion, hvor brugeren kan se hvornår dens domæner, tilknyttet til Surftown skal fornys.
14. Løbende priser for brugerens domæner.
    (a) En funktion hvor brugeren kan se, hvor meget dens domæner koster per betalingsperiode.
15. Løbende priser for brugerens webhostings.
    (a) En funktion hvor brugeren kan se faktureringsprisen for hver hosting service, som brugeren har hos Surftown.
16. Antal tilladte tilknyttede domæner til hver af brugerens webhostings.
    (a) En funktion hvor brugeren kan se hvor mange domæner, Surftown tillader at tilknytte til brugerens hosting services.
17. Visning af tilkøbte supportkoder

(a) En funktion hvor brugeren kan se evt tilkøbte supportkoder.

Af de funktionelle krav prioriteres implementeringen af de første 6 punkter højest. Det er valgt fordi Surftown prioriterer dem højest.

Ikke-funktionelle krav

1. Applikationen skal være brugervenlig og intuitiv. Her er det vigtigt at sætte fokus på brugerne. De skal føle sig hjemme, når de bruger applikationen. F.eks på iOS-versionen er det valgt at bruge UINavigationController til navigation i applikationen. Det er standard user experience på iOS applikationer. Det kendetegnes f.eks. ved "< Back "knappen, som alle iOS brugere ved, hvad gør.

2. Applikationen skal være hurtig. Som skrevet før kan Surftowns hjemmeside blot hentes ned og vises pænt, hvilket vil være langsomt og irriterende. Derfor implementeres applikationerne nativt til deres styresystem. Der fokuseres også på data-lagring (caching), så anvenderen ikke skal hente de samme data ned fra web-serveren, hver gang anvenderen går til et view. F.eks. lagres domæner i en række (array), så det hurtigt kan vises, næste gang domæne-skærmen (view) bliver vist.

3. Applikationen skal være stabil. Det vil sige ingen eller kun få tider, hvor webserveren er nede og ingen crashes eller fejl.

Begrænsninger

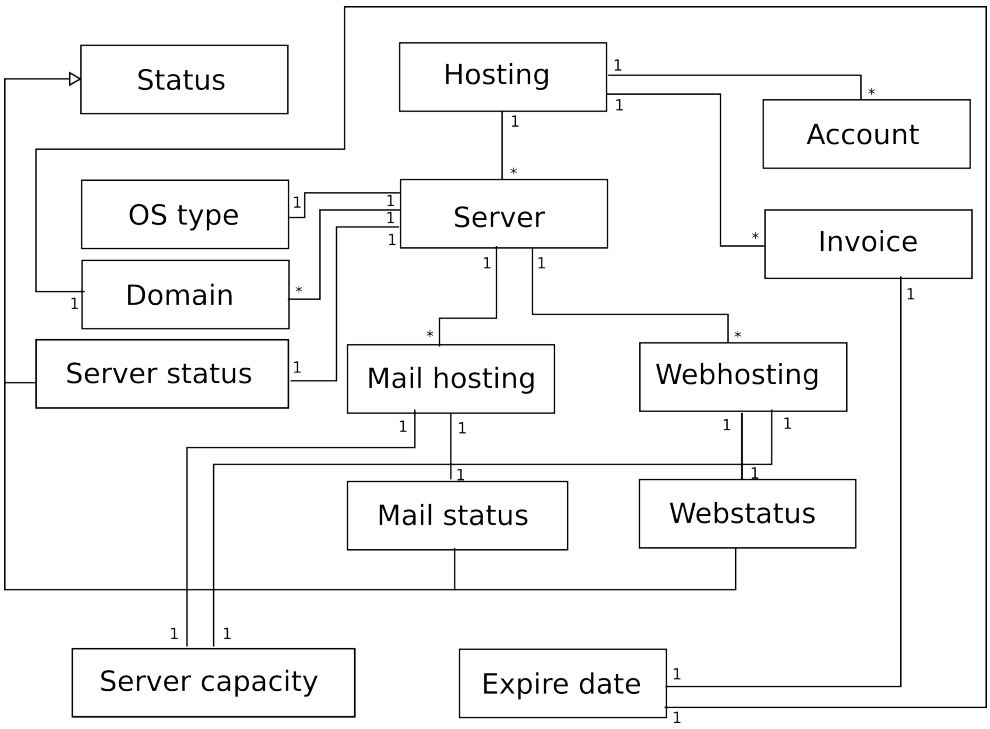
1. Applikationen skal følge Surftowns officielle design-guidline. Den er ikke tilgængelig for offentligheden

2. Applikationen skal nemt kunne oversættes og understøtte forskellige sprog [implementeringskrav]

3. Applikationen skal skrives nativt til Android og iOS [implementeringskrav]

Klassediagram over problemområdet

Figur 1 illustrerer et klassediagram af problemområdet inspireret af metoderne beskrevet i kapitel 5 *Analysis* i [2].



Figur 1: Figur af klassediagramet over problemområdet

1. **Hosting:** Klassen "Hosting" fungerer som hovedklassen i diagrammet, og er klassen, som repræsenterer det produkt Surftown udbyder, som deres kunder har købt (uafhænigt af servicetypen).

2. **Account:** For hver "Hosting" er der adskillige "Account", som repræsenterer de brugere, som er oprettet, og har adgang til hosting hos Surftown.

3. **Invoice:** Da Surftown ikke tilbyder nogle gratis produkter, er der minimum en "Invoice" (oprettelsesgebyr) klasse tilknyttet en "Hosting" klasse. "Invoice" klassen repræsenterer en faktura fra Surftown.

4. **Server:** En "Hosting" kan have tilknyttet flere forskellige "Server"'e.

   (a) **OS type:** Da Surftown både tilbyder servere med Linux og Windows som operativsystem, har klassen "Server" tilknyttet en værdi, der repræsenterer operativsystemet på serveren.

(b) **Domain:** Hver service kan have flere domæner tilknyttet. Domæner udløber, hvorefter de skal fornyes. Selvom domæner koster penge, antages det at klassen "Hosting" tager sig af det, da Surftown sender en samlet regning ud for hver service

(c) **Web/E-mail hosting:** En server kan have flere web og e-mail -hosting. Både web og email hosting har en ("Server Capacity"), som er mængden af data, der kan være på dem.

5. **Super klassen Status og Status(ser)**: I klassediagrammet på figur 1, eksisterer superklassen "Status", som bliver nedarvet fra hhv. "Server status", "Mail status" og "Web status", da de grundlæggende repræsenterer den samme form for information, men dog varierer i graden og kvaliteten (forstået som typen) af information.

Use cases og aktører

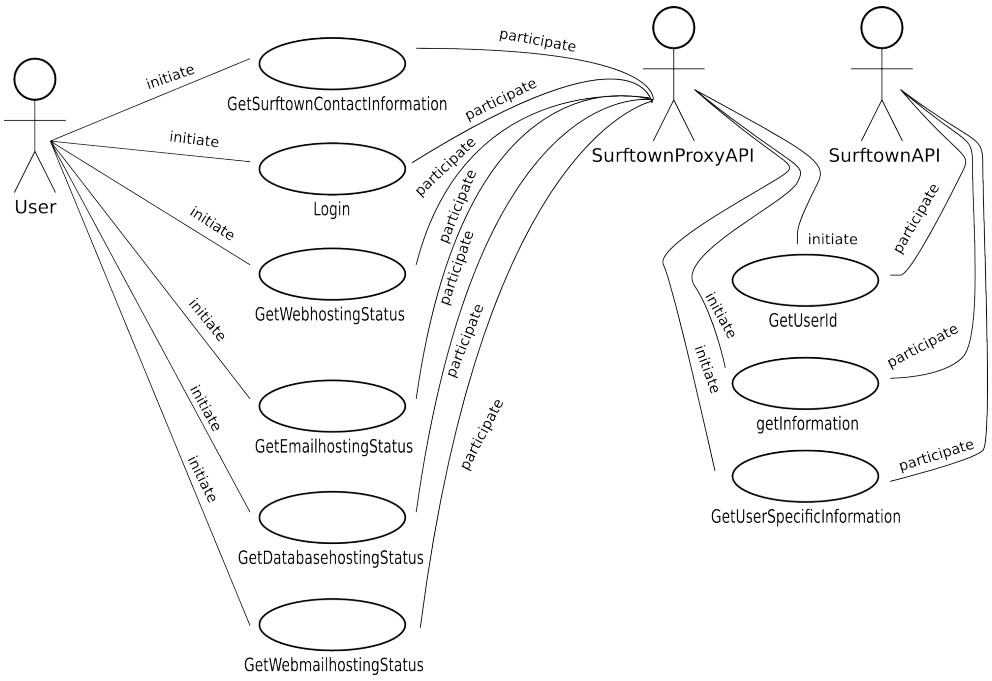
Der indgår tre forskellige aktører i systemet; "User", "SurftownProxyAPI" og "SurftownAPI", som hhv. er følgende:

1. **User:** Brugeren af applikationen.

2. **SurftownProxyAPI:** En mellemmand der står for kommunikationen mellem applikationen og Surftowns interne API.

3. **SurftownAPI:** Surftowns interne API som leverer informationer om brugeren og Surftown.

Figur 2 viser et højniveau-diagram af de seks første funktioner beskrevet i afsnit 1.2. Som man kan se på figur 2, bliver alle funktionerne i figuren listet i afsnit 1.2 startet af aktøren "User". De har alle sammen aktøren "SurftownProxyAPI" som participater, da alt information fra og til "User" går igennem denne aktør.
Aktøren "SurftownProxyAPI" kan starte tre forskellige funktioner, som alle har "SurftownAPI" som participater. De tre funktioner er: "GetUserId", "GetInformation" og "GetUserSpecificInformation". De er hhv. til at hente et unikt ID, som identificerer "User", hente generel information om Surftown og hente information omhandlende "User".



Figur 2: Højniveau-diagram af systemet som beskrevet i *Object-Oriented Software Engineering Using UML, Patterns, and JAVA*[2] kapitel 4.

Specificerede use cases

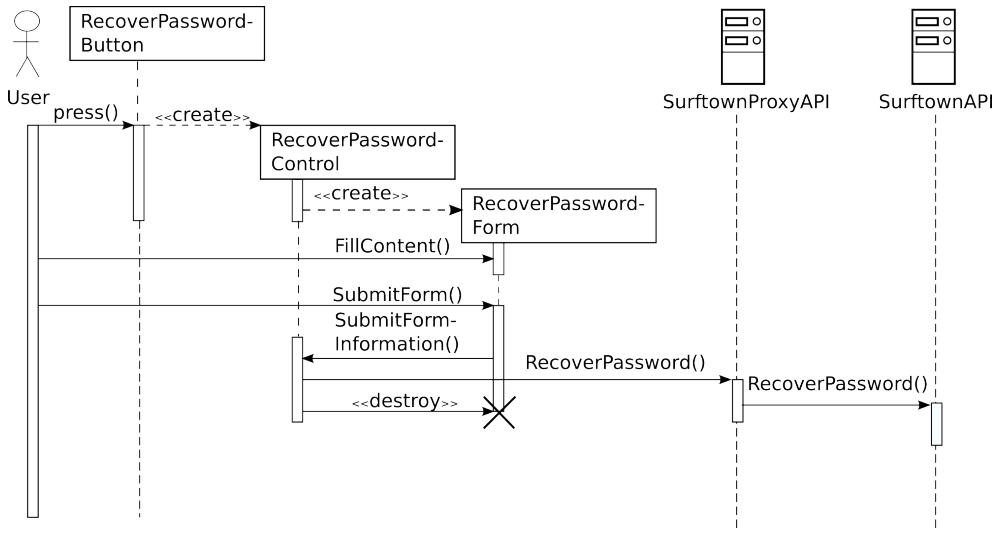
Som beskrevet i *Object-Oriented Software Engineering Using UML, Patterns, and JAVA*[2] kapitel 4, laves 3 forskellige use cases og deres tilhørende sekvensdiagrammer, som dækker de 3 mest interessante funktioner. De første to use cases og deres tilhørende sekvensdiagrammer, beskriver to forskellige specifikke funktioner, hhv. "RecoverPassword" (figur 3 og 4) og "Login" (figur 5 og 6). Den resterende funktion "GetPrivateInformation" (figur 7 og 8) beskriver funktionaliteten, når brugeren er logget ind.

I use case figurerne [figurer 3, 5, 7], er det for simpelthedens skyld valgt at behandle aktørerne "SurftownProxyAPI" og "SurftownAPI" som én aktør. Dette er gjort, da "SurftownProxyAPI" fungerer som en udvidelse til "SurftownAPI", og det således på et ikke-teknisk niveau fungerer som én aktør. De behandles dog som separate aktører i sekvensdiagrammerne.

Use case navn RecoverPassword

Deltagende aktøre	Startet af **User** Kommunikerer med **Surftown**
Event flow	1. **User** aktiverer funktion “Password recovery” i applikationen på sin telefon. 2. Applikationen svarer igen ved at vise **User** en form. Formen består af ét felt, hvor **User** kan angive sin e-mail adresse. Når **User** har angivet sin e-mail adresse, indsender **User** formen. 3. Applikationen modtager formen, og sender **User**s angivede e-mail adresse til **Surftown**. 4. **Surftown** modtager e-mail adressen, og sender en e-mail til e-mail adressen med en vejledning til at genskabe kodeordet. 5. **User** modtager en e-mail, og følger vejledningen.
Resultat	**User** får nyt password

Figur 3: Figuren viser et use-case-diagram af “RecoverPassword“. ”Recover-Password“ er en funktion som aktøren ”User“, brugeren af applikationen, kan aktivere ved at trykke på en knap, i tilfælde af at brugeren har glemt sine loginoplysninger til Surftown. Når aktøren ”User“ aktiverer denne funktion, viser applikationen en udfyldelses-form, hvor ”User“ kan indtaste og indsende sin E-mail adresse til Surftown. Hvis ”User“'s indtastede E-mail adresse er genkendt af Surftown, vil Surftown sende nye loginoplysninger til aktøren ”User“.

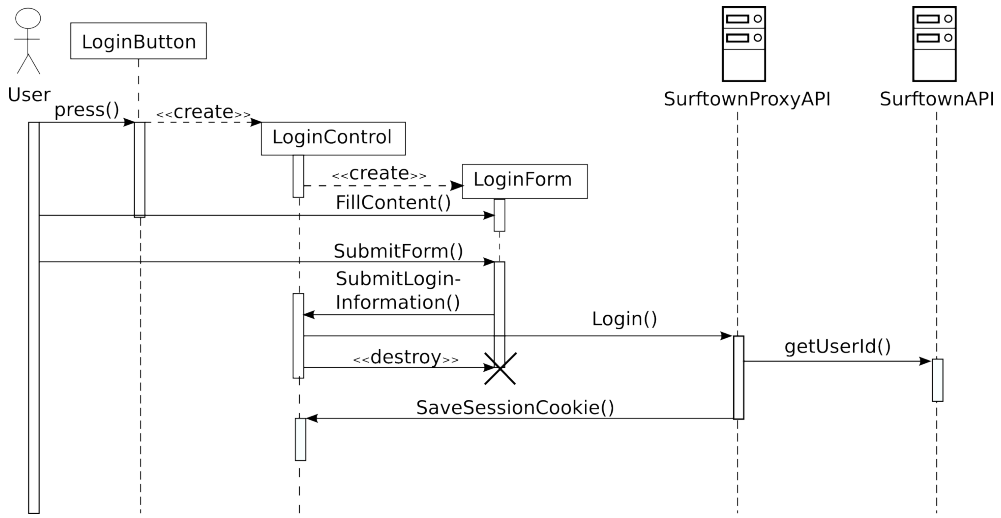


Figur 4: Figuren viser et sekvensdiagram af "RecoverPassword" funktionen, hvis use case er afbilledet i figur 3. I denne figur er alle tre forskellige aktører vist. Det starter med at aktøren "User" aktiverer funktionen "RecoverPassword" ved at trykke på en knap. Herefter bliver der oprettet et "RecoverPassword" objekt, som opretter en boundary, i form af en udfyldelses form, hvor "User" kan indtaste information og indsende det. Når "User" har indsendt informationen, modtager "RecoverPassword" objektet informationen og destruerer den sidste boundary. Herefter sender "RecoverPassword" objektet informationen videre til aktøreren "SurftownProxyAPI", som sender det til "SurftownAPI", hvorefter "SurftownAPI" sørger for at sende nye logininformationer til "User".

Use case navn	Login
Deltagende aktøre	Startet af **User** Kommunikerer med **Surftown**

Event flow	1. **User** aktiverer funktion "Login" i applikationen på sin telefon. 2. Applikationen svarer igen ved at vise **User** en form. Formen består af to felter, hvor **User** kan angive sin e-mail adresse og password. Når **User** har angivet sin e-mail adresse og password, indsender **User** formen. 3. Applikationen modtager formen, og sender **User** angivet e-mail adresse og password til **Surftown**. 4. **Surftown** modtager e-mail adressen og passworded, og tjekker gyldigheden af e-mail adressen og passworded. Hvis gyldigt event 5 ellers event 6. 5. Applikationen logger **User** ind. **User** henter sine private informationer fra **Surftown**, med use casen "getPrivatInformation". 6. Applicationen indikerer overfor **User**, at **User** ikke er blevet logget ind, og sender **User** til event 2 i use case "Login".
Resultat	**User** er logget ind

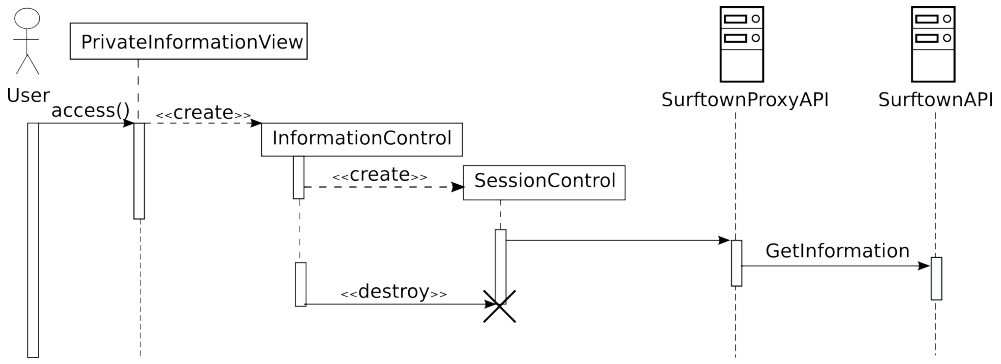
Figur 5: Figuren viser et use-case-diagram af Login. "Login" er en funktion, som giver aktøren "User" mulighed for at se sine private oplysninger, ved at angive sin E-mail og password, som er registreret hos Surftown.



Figur 6: Figuren er et sekvensdiagram af "Login" funktionen, hvis use case er afbilledet i figur 5. I denne figur er alle tre forskellige aktører vist. Aktøren "User" aktiverer "Login" funktionen, hvorefter et "Login" objekt bliver dannet. "Login" objektet opretter en login-form boundary, hvor brugeren kan udfylde sine login oplysninger og indsende dem. Når "User" indsender oplysningerne, modtager "Login" objektet dem, hvorefter "Login" objektet sender oplysningerne til aktøren "SurftownProxyAPI", samt destruerer login-form boundaryen. "SurftownProxyAPI" sender oplysningerne til "SurftownAPI" og modtager et user-id, hvorefter "SurftownProxyAPI" laver en session-cookie som "Login" objektet gemmer. Brugeren er nu logget ind.

Use case navn	getPrivatInformation
Deltagende aktøre	Startet af **User** Kommunikerer med **Surftown**
Event flow	1. **User** beder om sine private informationer. 2. Applikationen tjekker om **User** er logget ind. Hvis **User** er logget ind event 3 ellers event 4 3. Applikationen henter de private informationer fra **Surftown** og fremstiller dem for **User**. 4. Applikationen kalder use case "Login"
Start betingelse	**User** er logget ind.
Resultat	**User** bliver præsenteret for fortrolig information

Figur 7: Figuren er et use-case-diagram af "GetPrivateInformation" funktionen. "GetPrivateInformation" Denne funktion bliver kaldt, når brugeren skal se ikke-offentlige tilgængelige informationer (f. eks. informationer om brugerens services hos Surftown)



Figur 8: Figuren er et sekvensdiagram af "GetPrivateInformation" funktionen. Funktionen "GetPrivateInformation" bliver kaldt når aktøren "User" tilgår et view med privat information. Når funktionen bliver kaldt bliver et "InformationControl" objekt oprettet, som opretter et "SessionControl" objekt. "SessionControl" objektet sender en session-cookie (som er optaget via "Login" funktionen) til aktøren "SurftownProxyAPI". "SurftownProxyAPI" beder om de private informationer for "User" via et user-id som er associeret med session-cookien.

1.3 Systemdesign

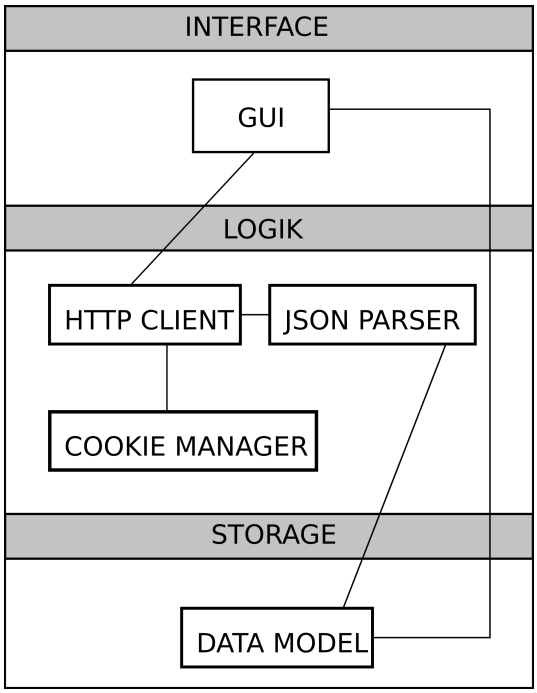
Surftown API

Surftowns API kaldes via HTTP og via en bestemt GET variable "call", kan man sætte hvilken funktion, man vil kalde[1]. Men da Surftowns API er designet til at blive brugt på et lokalt netværk, understøtter API'et ikke nogle former for handlings- eller informationsrestriktioner[2]. Det blev derfor nødvendigt at lave en udvidelse af Surftowns API.
Resultatet blev til en API proxy server, som er skrevet i Javascript og bliver kørt i NodeJS[3]. API proxy serveren understøtter handlings- og informationsrestriktioner med bruger-login og cookie-session, via HTTP protokollen. Det fungerer ved, at API proxy serveren kun tillader udvalgte funktioner at blive kaldt, og videresender informationerne til Surftowns API. Hvis en funktion som parameter kræver et bruger-id, vil API proxy serveren tjekke, om der

[1] Afhængig af hvilken funktion man kalder, kan man også sætte andre GET variabler.
[2] Dette var ellers blevet lovet af Surftown at de ville lave, i forbindelse med udviklingen af dette system
[3] En javascript-baseret server, bygget på Google Chrome's V8 motor

er sendt en gyldig session-cookie med i HTTP-requesten, hvis det er tilfældet vil proxy API serveren sende requesten videre sammen med bruger-id'et associeret med session-cookien.

Applikationen



Figur 9: Decomposition diagram af applikationen, inspireret af *Object-Oriented Software Engineering Using UML, Patterns, and JAVA*[2]

- **GUI[4]:** GUIen er implementeret sådan at ved "View" skifte, dvs. ved nyt skærmbillede, forespørger GUIen nye data fra HTTP CLIENTen, og fremviser herefter de data der er i DATA MODEL.
- **HTTP CLIENT:** Ved et request fra GUIen laver HTTP CLIENT en request til COOKIE MANAGER, for at få evt. cookies fra en tidligere HTTP request, herefter sender HTTP CLIENT en HTTP request til Surftown APIet og sender de modtaget data til JSON PARSER

[4]Graphical User Interface

- **JSON PARSER:** Når JSON PARSER modtager data vil den afkode dataene og sende det afkodet data til DATA MODEL.
- **COOKIE MANAGER:** COOKIE MANAGER sørger for at gemme eventuelle cookies fra tideligere HTTP request, og leverer dem til HTTP CLIENT ved afsendelse af en ny HTTP request. Derudover kontrollerer COOKIE MANAGER cookienes udløbstider
- **DATA MODEL:** DATAMODEL sørger for at udfylde foruddefineret datastrukturer med de nye data. Dette sørger for at dataerne forbliver konsistente overfor GUIet, som skal fremvise dem.

Manglende implementeringer

I afsnit 1.2 udtrykkedes et ønske om at implementere de første seks punkter som et minimum, hvilket der ikke blev nået. Der mangler stadig at blive implementeret:

1. Driftstatus af brugerens E-mail hosting.
2. Driftstatus af brugerens database hosting.
3. Driftstatus af brugerens webmail

Derudover mangles at implementere Surftowns officielle design, som er en begrænsning i afsnit 1.2.

1.4 Programtest og resultater

For at være sikker på at systemet er nemt og intuitivt, er der blevet foretaget nogle usability test sammen med en test bruger "User"[5]. "User" har allereder en hjemmeside hos en anden hosting leverandør end Surftown, men ser ikke sig selv som en erfaren- eller superbruger.
Målet med testene er at finde fejl, hvor systemet opfører sig anderledes end forventet specificeret ud fra use case'ene.
Fra *Object-Oriented Software Engineering Using UML, Patterns, and JAVA*[2] kapitel 11, er det beskrevet at idéen med usability test, ikke er at vise at systemet ikke indeholder nogle fejl eller bugs, men at forbedre måden systemet bruges på. Derfor er der også foretaget programtest af login, domæne

[5]Simon Warg's sambo

og service funktionaliteterne samt user-experience. Der er ikke foretaget nogle unit-tests. Dette skyldes at udviklere har prioriteret det fra, da kørselstest er rigeligt for de funktioner, der er implementeret. I kørsel bliver funktionerne testet kronologisk, og det ville være dobbeltarbejde at køre unit-tests først, efterfulgt af samme procedure, samtidig med at man tester for semantiske, grafiske og user-experience fejl.

Resultat

Brugeren "operatøren" bliver informeret om sin rolle og sine mål, som han skal opnå ved at bruge applikationen, som beskrevet i bilag 4.4.

Logge ind:	Brugeren tastede sit brugernavn og password ind, og klikkede på "Login" knappen. Brugeren nåede målet.
Finde services:	Brugeren klikkede på "Services" i menuen. Han blev spurgt, hvor mange services, han kunne se. Det første svar på spørgsmålet om, hvad type af service, han skulle vælge lød. Efter en kort forklaring blev det klart for **User**, at han havde 3 services, som var listet på skærmen. Brugeren nåede sit mål
Finde domains:	Brugeren klikkede på "Home" fra den nuværende "Service" og klikkede bagefter på "Domains" i hovedmenuen. Han blevet spurgt hvor mange domæner han kunne se. Han kunne se 3 domæner, hvilket ikke var korrekt. Han blev gjort opmærksom på, at der burde være flere domæner, og spurgte hvad han vil gøre for at finde de resterende. Han klikkede på "Home" og gik ud i hovedmenuen, hvorefter han gik ind i "Domains" igen. I andet forsøg fandt han ud af, at man kunne scrolle for at se de resterende domæner, og han svarede 7, hvilket var korrekt. Brugeren nåede sit mål
Password reset:	**User** klikkede på "Forgot password" på login-skærmen, hvorefter et nyt stykke papir blev fremlagt på bordet. **User** "tastede" sin e-mail ind og klikkede på "Send". Brugeren nåede sit mål.

Resultat diskusion

Det er væsentligt at nævne, at der skulle have været foretaget flere user-tests før at resultatet ville være overbevisende. Udover den kandidat vi har valgt,

ville det være åbentlyst også at have inkluderet nogle udvalgte kunder hos Surftown, som i sidste ende er dem, der kommer til at bruge applikationen.

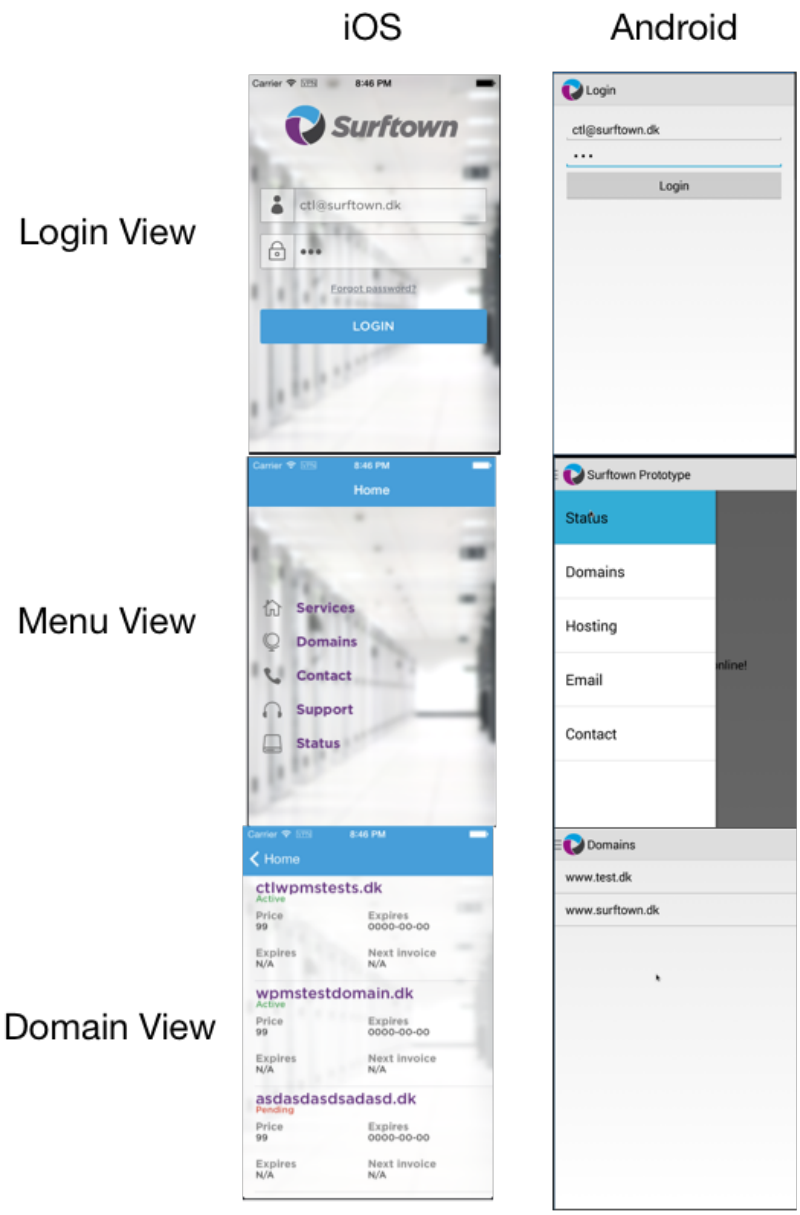
Resultat konklusion

Testene af systemet gik overordnet godt, og brugeren havde ikke de store problemer med at finde de forskellige informationer og funktioner. Dog var der enkelte terminologiske problemer for brugeren, men da terminologien er taget direkte fra Surftowns eksisterende kontrolpanel, er det besluttet, at det ikke skal ændres.

1.5 Brugergrænseflade og workflow

Skærmbilleder

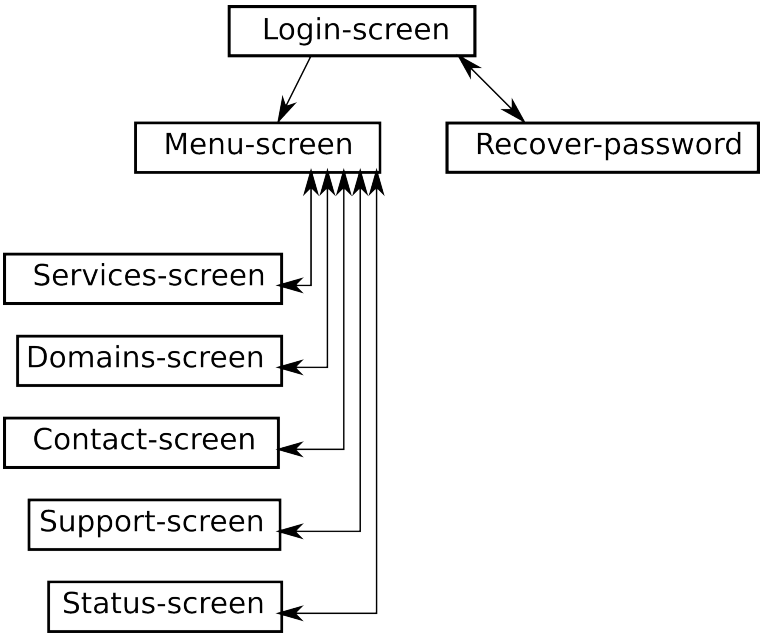
De følgende billeder er forskellige skærmbilleder af de vigtigste dele af applikationen på Android og iOS. Dataene som er vist på billederne er data som Surftown har oprettet, på den test server, de har stillet til rådighed til projektet.



Figur 10: Loginskærm på iOS

På figur 10 ses udvalgte skærmbilleder fra kørsler på hhv. iOS og Android miljø. Øverste billeder er af login skærmen, efterfulgt af menuen og til sidst domæneoversigt af skærmen. På domæneoversigten kan man se, hvad jeg refererede til tidligere omkring "< Back"knappen. Fra menuen, "Home", klikkes på Domains, hvorefter domæneskærmen vises. Nu står der så en pil tilbage i venstre hjørne med titlen Home. Dermed ved brugeren, hvordan man let kommer tilbage til menuen Home.

Workflow



Figur 11: Flowdiagram over hovedfunktionerne i Surftown applikationen

På figur 11 ses et flowdiagram over hovedfunktionerne i applikationen, hvor der tages udgangspunkt i, at man kommer fra "Login" skærmen.

- **Login-screen:** Her kan man enten logge ind i applikationen, eller man kan prøve at genskabe sit password.
  - **Recover-password:** Her kan man genskabe sit password og gå tilbage til "Login" skærmen.

- **Menu-screen:** Her har man mulighed for at komme ind på forskellige "views", for at se sine oplysinger. For alle underpunkter gælder det, at man kan komme tilbage til menuen for at vælge et andet "view".

1.6 AV præsentation

AV præsentation af prototypen, kan findes på følgende link: http://youtu.be/coaXCn-iWJE

1.7 implementeringsarbejde

Proxyen blev implementeret på baggrund af et sikkerhedshul i Surftowns interne system (Hostbill (HB)). HB's API ses som et internt system, som ikke er egnet til brug online. Sikkerhedshullet blev fundet under implementering af login-funktionen. Proxyen bruger sessions og cookies til at validere, om en bruger er logget ind. I HB laver man istedet en GET request med sine login-informationer, sammen med en API-nøgle, hvorefter man får et client-id. Dette client-id bruges f.eks. til at slette en bruger. Derfor blev proxyen implementeret, så en session-cookie er nødvendig for at lave diverse requests. Efter en validering sætter proxyen en session-cookie og gemmer derefter brugerens ID. Dvs. at det kun er proxyen, der kender den autoriserede brugers id, og derfor er informationer kun tilgængeligt for den omtalte bruger.

2 Systemudviklingsproces

2.1 Projekt oversigt

Begivenheder	**Dato Fra**	**Dato Til**
Kontakt med Surftown	10/2-2014	10/2-2014
Første møde med Surftown	18/2-2014	18/2-2014
Analyse af Hostbill API og implementeringsmuligheder	19/2-2014	16/3-2014
Andet møde med Surftown	17/3-2014	17/3-2014
Design af Applikation og Grafisk Design	18/3-2014	1/4-2014
Implementering af prototyper	1/4-2014	1/5-2014
Adgang til Developer Box	1/5-2014	1/5-2014
Implementering af proxy	1/5-2014	6/5-2014
Implementering af Applikationer (Det der blev nået)	7/5-2014	10/6-2014

Tabel 1: Skematisk oversigt over projektets fremgang fra start til slut

Som det ses i tabel 1, blev der kun holdt to møder med Surftown. Det skyldes, at der ikke har været stort behov for at afholde flere, da Simon og Christian arbejder hos Surftown, og derfor medbragte spørgsmål og svar på arbejde. Man kan diskutere, om et ekstra møde skulle afholdes, da det kunne have sat de sidste ting på plads, f.eks ændringer i kravene el. forvirringer i HB. HB er uoverskuelig og ikke veldokumenteret.

2.2 Forhold mellem parterne

Surftowns marketingschef, som også fungerer som brugerrepræsentant, har oftest været meget kritisk og uformel i sine idéer og meninger omkring kravene. Der er blevet taget kontakt til ham og HR-chefen, hvor HR-chefen, modsat ham, har været positiv over for idéer. Udviklerne har sat fokus på, at applikationen skulle være sikker og hurtig, og har derfor ikke været fokuserede på grafikken til start. Det har skabt forvirring hos Surftown, da de har været mere interesseret i det visuelle end det funktionelle. Dette har været en begrænsning for udviklerne, og har haft en demotiverende konsekvens, og har til dels nedsat udviklingsprocessen.

2.3 Samarbejdet internt

Projektgruppen har fungeret rigtig godt socialt. Der har været flere møder, hvor der er blevet diskuteret, hvad der skulle laves til næste gang, og hvad Surftown har kommunikeret ud[6] Der er holdt to møder med Surftown igennem forløbet. Gruppen har været dårlige til at strukturere arbejdet og tage referater af møderne, hvilket har kompliceret projektforløbet.

2.4 Tilrettelæggelse og samspil mellem systemudviklingsaktiviteterne under projektforløb

Systemets udvikling er dokumenteret med versionsstyresystemet Git. Det har været en styrke under prototyping, da man hurtigt og nemt kan lave en branch, og teste noget nyt. Det var især vigtigt da problemområdet blev ændret ift. implementering af proxy-serveren. Alle API kald skulle skrives om, så de fulgte et RESTful design. Det var svært at få noget konkret for anvendelsesområdet, pga besværligheden ved at installere beta-versioner af applikationer på smartphones. Det resulterede således også i at projektdomænet kun fik foretaget en enkel user test. Det var svært at få et godt samarbejde mellem de to grupper hhv. Android og iOS, da det kun er proxy-serveren, samt kravsspecifikationen, der var til fælles. Selve designet på applikationerne og implementering foregår i to forskellige perspektiver og guidelines fra hhv. Google og Apple. Dette medfører en begrænsning mellem de to grupper, da der rent implementeringsmæssigt ikke kunne genbruges meget så som resurser eller kildekode. Der var dog den styrke, at begge platformer benytter Model-view-controller designmønstret, hvilket gjorde, at man kunne skrive den samme model, evt. i c++, som begge platforme understøtter.

2.5 Forbedringer

Det første og det vigtigste er at få styr på problem- og anvendelsesområdet. Udviklere og projektledere kan styrke sine brugeres oplevelse, ved at få en klar afklaring af målet. Dette kan gøres ved user tests, for at se om brugeren opfatter målsystemet efter deres opfattelse, eller om han/hun opfatter det som et andet objektsystem. Altså om brugeren har forstået, hvad systemet skal bruges til, og hvordan det skal anvendes, eller om han/hun har sin egen

[6]Da to medlemmer af projektgruppen arbejder hos Surftown, er det naturligvis den primære kommunikationsvej.

opfattelse. I bund og grund er planlægning vigtig. Det er en forudsætning at få klare udmeldinger ud omkring systemets funktionalitet, og hvem der skal bruge det.

Konklusion på udviklingsproces

Konsekvensen af omstændighederne har medført at udviklingsgruppen, ikke har nået at lave applikationen færdig. I starten af projektet blev projektet taget seriøst, og der blev arbejdet godt sammen, hvilket resulterede i hurtige gode resultater, men på baggrund af ovenstående uoverensstemmelserne mellem parterne, medførte det et dårligt resultat. Kigger man tilbage nu ville man kunne forbedre vores forløb ved at lave flere user- og programtest og have planlagt og strukturet forløbet bedre. Udviklingsprocessen har derfor været både god og dårlig.

3 Review af "Challenges of migrating to Agile methodologies" [3]

3.1 Resumé

Artiklen handler om de fordele og ulemper, der er ved at skifte til "Agile"metoder. Den beskriver forudsætningerne for anvendelsen, hvilke menneskelige-, organisations-, proces- og teknisk relaterede problemer, der findes og bl.a hvorfor man skal overveje at anvende metoderne. Til at starte med kommer artiklen ind på forskellen mellem planlægningsbaserede løsninger og Agile metoder. Den redegør derefter, for hvad Agile metoder egentlig er og deres konsekvenser samt den internationale popularitet og tilgang til metoderne.

3.2 Analyse og diskussion

Artiklen nævner forskellen på Agile- og traditionelle metoder. I systemudvikling er det en god idé at bruge traditionelle metoder for nogle systemer. Artiklen nævner f.eks. at alt skal planlægges under et projekt, og der bliver tildelt roller. Det ses f.eks. ved store systemer, som Microsofts Windows og Apples OSX, hvor der findes projektdomæne, brugerrepræsentanter, analytikere, ingeniører, driftpersonale, grafikere, ledere osv. I modsætning har vi

Agile metoder, hvilket efter min opfattelse af artiklen, giver mening til mindre og ikke-risikable projekter, så som vores mobile applikation. Her har vi været flittige til at fokusere på vores kreativitet frem for planlægning. Det ses f.eks. ved implementeringen af vores Proxy servere. Havde vi planlagt og analyseret dybere i den indledende fase, havde vi hurtigt set det sikkerhedshul, der gjorde os nødsaget til at lave proxyen. I projektgruppen findes heller ikke en leder, som tager beslutningen, men istedet foretages de i fællesskab på basis af tankegange og prototyper. Det som er væsentligt er at Agile metoder fungerer i praksis til små projekter, som folk ikke er afhængige af, og som hurtigt kan opdateres, hvor den traditionelle planlægningsmetode med UML-diagrammer og OOAD[7] skal anvendes til store API [8] og operativsystemer. Som artiklen nævner, kan det være en god teknik at blande de to ender. Det er vigtigt at få styr på kravspecifikation, begrænsninger, kundetests og analyse af problem- og anvendelsesområde. Er der ikke foretaget disse basale steps, vil man ikke vide, hvad der skal laves og hvorfor. I modsætning vil man med disse informationer og aftaler kunne bruge Agile metoder efterfølgende.

Artiklen er velskrevet og godt dokumenteret. Den er skrevet på basis af et objektivt syn på problemområderne med en smule mere fokus på Agile metoder, hvilket giver mening, da det er grundlaget for artiklen.

[7] Objekt-orienteret analyse og design
[8] Application programming interface

4 Bilag

4.1 Kildekode til proxy API server

Koden til proxy API server er vedhæftet som en zipfil med navnet proxy.zip

4.2 Kildekode til Android applikation

Koden til Android koden er vedhæftet som en zipfil med navnet android.zip

4.3 Kildekoden til iOS applikationen

iOS koden er vedhæftet som en zipfil med navnet ios.zip

4.4 Testspecifikationer

4.4.1 RecoverPassword

Da funktionen "PasswordRecovery" ikke er implementeret endnu, vil testen blive gennemført ved at vise test brugeren nogle stykker papir, der viser hvordan funktionen ville se ud, hvorefter brugeren forklarer, hvad han/hun ville gøre.

Mål med testen	At **User** kan genskabe sit password
Deltagende aktøre	**User**

Event flow

1. **User** bliver før testen starter, fortalt at han/hun håndterer sine domæner og web hosting ved hjælp af Surftowns kontrolpanel. Han/hun skal nu bruge applikationen til at genskabe sit password
2. Foran **User** bliver der fremlagt det første stykke **skærmpapir** der forestiller login-skærmen, hvor der findes en "glemt-password" knap.
3. Hvis **User** trykker på et interaktivt område i billedet, så vil test manageren skifte **skærmpapir**et til det efterfølgende skærmbillede.
4. Testen afsluttes når **User** har opnået målet, eller hvis **User** ikke kan komme videre.

4.4.2 Login- og navigationstest

Denne test kan udføres igennem en iOS simulator.

Mål med testen	At **User** kan logge ind At **User** kan finde sine hosting services At **User** kan finde sine registrerede domæner
Deltagende aktøre	**User**

Event flow

1. **User** bliver fortalt, at han/hun håndterer sine domæner og web hosting ved hjælp af Surftowns kontrolpanel. Han/hun skal nu bruge applikationen til at logge ind for at finde sine domæner og services
2. Foran **User** bliver der sat en computer med en iOS simulator kørende og login-skærmen er blevet initialiseret.
3. Via musen på computeren kan **User** navigere og interagere med applikationen.
4. Testen afsluttes når **User** har opnået målet, eller hvis **User** ikke kan komme videre.

Litteratur

[1] L. Mathiassen, A. Munk-Madsen, P. A. Nielsen og J. Stage. *Object-Orinted Analysis & Design.* Marko Publishing House, 2000.

[2] B. Bruegge og A. H Dutoit. *Object-Oriented Software Engineering*

[3] Shridhar Nerur, Radhakanta Mahapatra og George Mangalara *Challenges of migrating to Agile methodologies*