Mobil-applikation til administration af webhosting-platform
Projektkursus Systemudvikling 2014
Delrapport 4

Nicklas Warming Jacobsen 230891
Christian Enevoldsen 020691
Simon Warg 180489
Robert Rasmussen 050394

04-06-2014

Instruktor: Kasper Passov

Indhold

1 Abstract

The following report explains how the development of an application for both Android and iOS is created for and with the company Surftown. Surftown is a danish web-hosting company, with mainly scandinavian (Danish, Swedish and Norwigian) customers, which offers three services: Web-hosting, E-mail hosting and domain-name registration, plus a web-based controlpanel to allow Surftown's customers to configurate their acquired services. However Surftown wishes a smartphone application, to enable their customers to access parts of the functionality in their controlpanel, mainly informational functions. One of Surftowns requirements for this project are that the application is programmed native to Android and iOS. We have therfore split the projekt group in two sub-groups containen two persons in each to focus respectively on the iOS application development and the Android application development. We use the same Surftown API and share some ressources between the Android application and the iOS application, and with this in mind we have created three git repositories; one for the iOS application development, one for the Andrioid application development and one for the shared ressources between the two.

2 Formål og rammer

Vi bruger "FACTOR" [1] analysen til at konkretiserer rammene og problem området for vores projekt. FACTOR anlysen består af seks forskellige punkter:

1. **F**: Beskrivelse af funktioner som systemet skal indeholde for at løse problemområdet.
2. **A**: Beskrivelse af de aktøre som skal bruge systemet til at, som er er indehaver af problemområdet.
3. **C**: Beskrivelse af de forhold for som systemet bliver udviklet under, samt de forhold som systemet skal bruges.
4. **T**: Beskrivelse af de teknologier som systemet bliver udviklet med, samt de teknologier som systemet i sidste ende vil køre med og kræve.
5. **O**: Beskrivelse af hovedobjekterne som problemområdet indeholder.

6. **R**: Brskrivelse af hvad systemet i sidste ende vil have ansvar for.

F

Mobil adgang til udvalgte funktioner og statusser, samt support hos Surftown. Mere konkret ville en bruger for eksempel kunne se om nogle af Surftowns servere er ude af drift, og om kundens egne services er påvirket af det.

A

Surftowns kunder som vil have mulighed for at tjekke status og lave mindre ændringer i deres Surftown services uden at have adgang til en "rigtig" computer og internet.

C

Brugerene har en hvis form for IT-kendskab dog er de ikke nødvendigvis professionelle. Surftown har ydret stærk ønske om en native applikation til iOS og Android. Udviklingsarbejdet foregår ikke på fuld tid, eftersom det er et studieprojekt.

T

Systemet skal køre på iOS og Android. Android applikationen vil blive udviklet med Google's Android SDK, med deres tilhørende Eclipse plugin. iOS applikationen vil blive udviklet med Apple's xCode development IDE. De to grupper til udviklingen af Android og iOS applikationerne har henholdsvis 2 Android og iOS telefoner hver.

O

Objekterne for problemområdet er beskrevet i sektion 3.4

R

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

3 Kravspecifikation

Sammen med Surftown har vi besluttet nogle funktionelle og ikke-funktionelle krav, samt nogle begrænsning til applikationen. Disse krav er nogle vi mener er vigtige for brugeren og samtidige er brugbare på en smartphone. Grunden til at vi udvikler applikationen native til begge operativ systemer, er fordi at Surftown har givet os den begrænsning at den skal laves nativt, dette opfylder samtidigt det ikke-funktionelle krav at applikationen skal være hurtig.

3.1 Funktionelle krav i prioteret rækkefølge

1. Surftown kontaktoplysninger.
    - Et simpelt skærmbillede hvor brugeren kan se Surftowns tlf. nummer, e-mail og adresse.
2. En indlognings funktion.
    - En funktion som giver brugeren af applikationen mulighed for at logge ind med e-mail og password.
3. Driftstatus af brugerens webhosting.
    - En informativ funktion, hvor brugeren kan se om Surftown har nogle tekniske fejl, som påvirker brugerens webhosting hos Surftown.
4. Driftstatus af brugerens E-mail hosting.
    - En informativ funktion, hvor brugeren kan se om Surftown har nogle tekniske fejl, som påvirker brugerens E-mail hosting hos Surftown.
5. Driftstatus af brugerens database hosting.
    - En informativ funktion, hvor brugeren kan se om Surftown har nogle tekniske fejl, som påvirker brugerens database hosting.
6. Driftstatus af brugerens webmail.
    - En informativ funktion, hvor brugeren kan se om Surftown har nogle tekniske fejl, som påvirker brugeren webmail(s).

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 recover sit bruger password.
    - En funktion hvor brugeren kan skrive sin E-mail ind for at få tilsendt et nyt password.
12. En oversigt brugerens fakturaer.
    - En informativ funktion, hvor brugeren kan se en oversigt over de betalte og udestående fakturaer som brugeren har hos Surftown.
13. Udløbningsdato'er for brugerens domæner.
    (a) En informativ funktion, hvor brugeren kan se hvornår hvert domæne som brugeren ejer og som er tilknyttet Surftown skal betales næste gang.
14. Løbende priser for brugerens domæner.
    (a) En informativ funktion, hvor brugeren kan se hvor meget hvert domæne som brugeren ejer og som er tilknyttet Surftown koster per betalingsperiode. (Dette er relevant da .dk, .se, .com osv. har forskellige priser)

15. Løbende priser for brugerens webhostings.

    (a) En informativ funktion, hvor brugeren kan se hvor meget hver hosting service, som brugeren har hos Surftown, koster per betalingsperiode.

16. Antal tilladte tilknyttede domæner til hver af brugerens webhostings.

    (a) En informativ funktion, hvor brugeren kan se hvor mange domæne Surftown tillade at tilknytte til brugerens hosting services hos Surftown.

17. Visning af tilkøbte supportkode

    (a) En informativ funktion, hvor brugeren kan se sine tilkøbte supportkoder hos Surftown.

Af funktionelle krav vil vi om minimum gerne implementerer til og med punkt 6, da det har højest priotet hos Surftown.

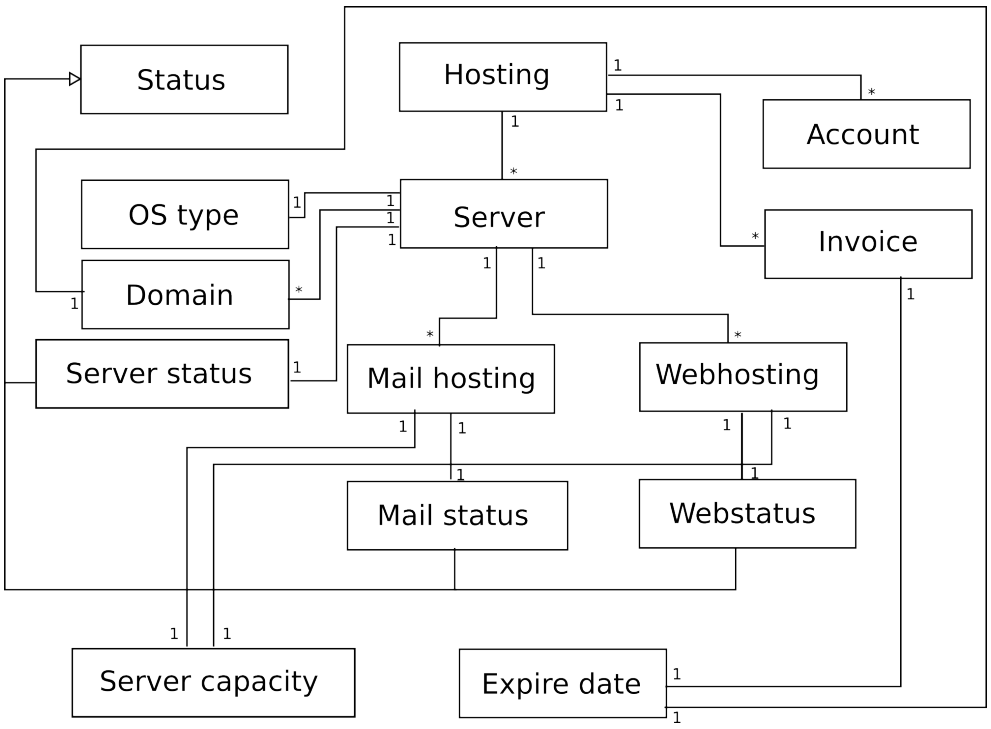
3.2 Ikke-funktionelle krav

1. Applikationen skal være nem og intuitiv at bruge.
2. Applicationen skal være hurtig.
3. Applikationen skal være stabil.

3.3 Begrænsninger

1. Applikationen skal følge Surftowns officielle design-guidline.
2. Applikationen skal nemt kunne oversættes og undestøtte forskellige sprog [implementeringskrav]
3. Applikationen skal skrives native til Android og iOS [implementeringskrav]

3.4 Klassediagram over problemområdet



Figur 1: Figur af klassediagramet over problemområdet

På figur 1 kan ses et klassediagram af vores problemområde, inspireret af metoderne beskrevet i kapitel 5 *Analysis* i [2].

1. **Hosting:** Klassen "Hosting" fungerer som hovedet klassen i diagramet, og er den klasse som repræsenterer samlet set det produkt som Surftown udbyder og som deres kunder har købt (uafhænigt af servicetypen).

2. **Account:** For hver "Hosting" kan der være adskillige "Account", som repræsenterer de bruger som er oprettet og har adgang til hostingen hos Surftown.

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

4. **Server:** En "Hosting" kan have flere forskellige "Server"'er tilknyttet til sig.

   (a) **OS type:** Da Surftown både tilbyder serverer med Linux og med Windows operativsystemer, har klassen "Server" tilknyttet en værdi der repræsenterer hvilket operativsystem serveren køre.

   (b) **Domain:** Hver server kan have flere forskellige domæner tilknyttet til sig. Domæner udløber, hvorefter de skal genkøbes. Selvom at domæner koster penge, antager vi at det er noget klassen "Hosting" tager sig af, da Surftown sender en regning ud samlet for ens services (Hvilket indbefatter at betale penge for et eller flere domain name(s))

   (c) **Web/E-mail hosting:** En server kan have flere web og e-mail -hosting. Både for web og e-mail -hosting gælder det at de har en bestemt mængde plads ("Server Capacity"), som er mængden af data der kan være på de to.

5. **Super klassen Status og Status(ser)**: I klassediagrammet på figur **??**, kan man se der eksisterer en super klasse kalder "Status", som bliver nedarvet fra henholdsvis "Server status", "Mail status" og "Web status". De tre forskellige status klasser nedarver alle fra superklassen "Status" da de grundlæggende repræsenterer den samme form for information, men variger i graden og kvaliteten (forstået som typen) af information.

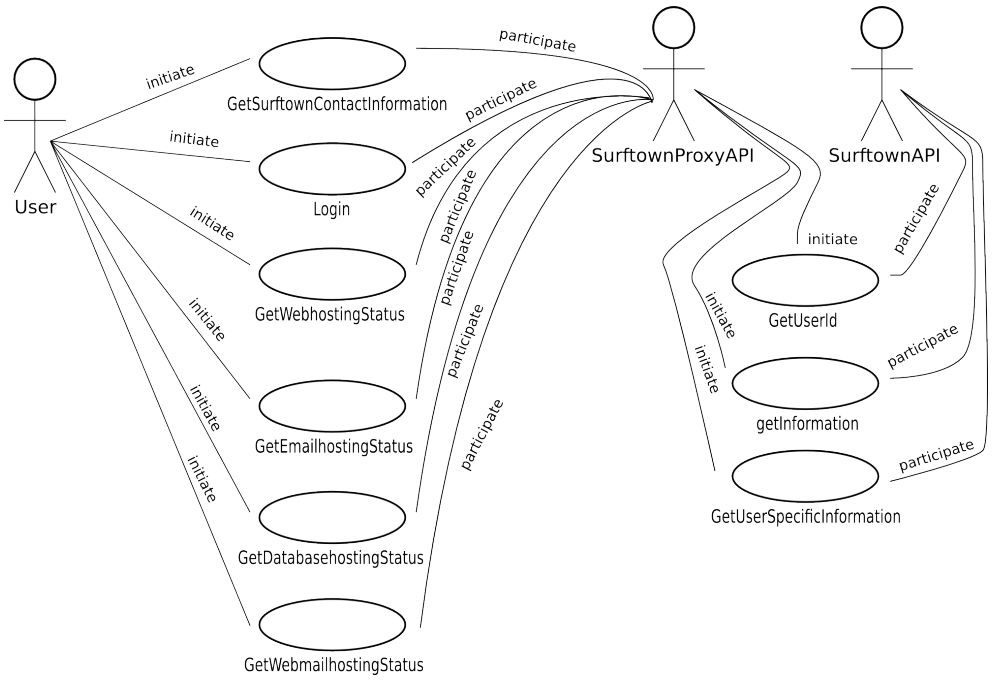
3.5 Use cases og aktører

Der indgår tre forskellige aktører i vores system; "User", "SurftownProxyAPI" og "SurftownAPI", som henholdvis er følgende:

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

2. **SurftownProxyAPI:** Et middleware der står for kommunikationen mellem applikationen og Surftown interne API.

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



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

Figur 2 viser et højniveau-diagram af de seks første funktioner beskrevet i afsnit 3.1. Som man kan se på figur 2, bliver alle funktionerne i figuren som er listet i afsnit 3.1 startet af aktøren "User", samt de har alle 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. Disse tre funktioner: "GetUserId", "GetInformation" og "GetUserSpecificInformation" er henholdvis til at: hente det unikke ID som indentificere "User"; hente generel information om Surftown; og hente information omhandlende "User".

Specificerede use cases

Som beskrevet i *Object-Oriented Software Engineering Using UML, Patterns, and JAVA*[2] kapitel 4, har vi lavet 4 forskellige use cases og deres tilhørende sekvensdiagramer, som dækker de fire mest interresante funktioner. De

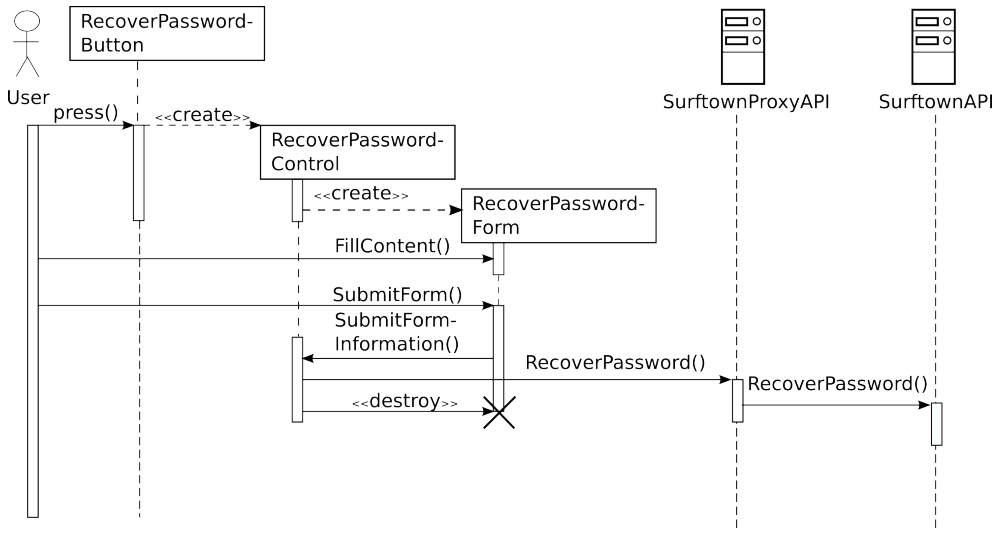
første to use cases, og deres tilførende sekvensdiagramer beskriver to forskellige specifikke funktionen, henholdsvis "RecoverPassword" (figur 3 og 4) og "Login" (figur 5 og 6). De to resterende funktioner, henholdsvis "GetPublicInformation" (figur 7 og 8) og "GetPrivateInformation" (figur 9 og 10) er mere generelle. Vi har tilladt os at fremstille det på denne måde, da proceduren for at indlæse offentlig tilgængelig information fra Surftowns server er ens uafhængigt af typen af informationerne, hvilket også gælder for at indlæse bruger-specifik (privat) information.[1]

Vi har i use case figurene [figurer 3, 5, 7 og 9], for simpeltheden og redundansen skyld, valgt at behandle aktørene "SurftownProxyAPI" og "SurftownAPI" som èn aktør. Dette har vi tilladt os, da "SurftownProxyAPI" fungerer som en udvidelse til "SurftownAPI", og det således på et ikke-teknisk niveau fungerer som èn aktør. Vi behandler dem dog som seperate aktører i sekvensdiagrammerne.

[1]Dette skal ikke forstås som at det er en og samme procedure for at indlæse privat og offentlig information

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 svare 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** angivet 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 passworded. 5. **User** modtager en e-mail, og følger vejledningen.
Resultat	**User** får nyt password

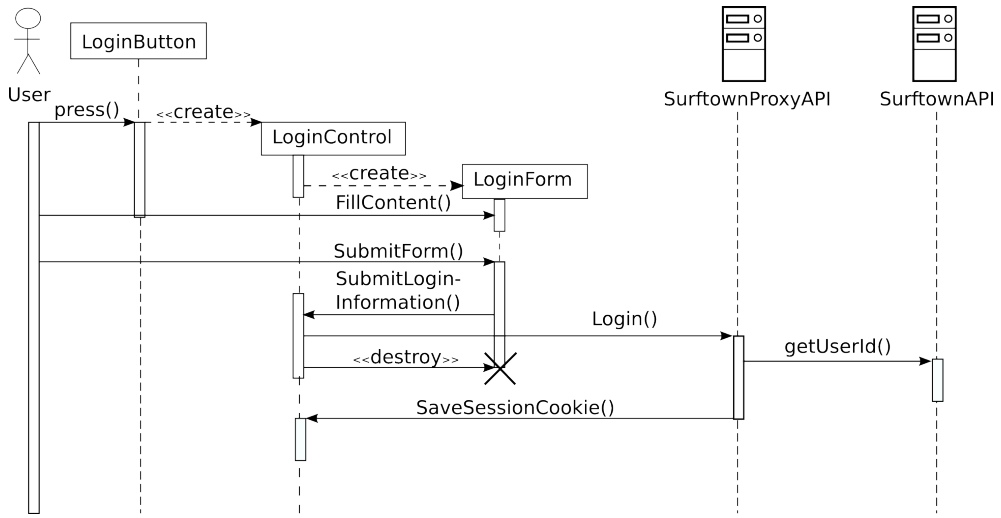
Figur 3: Figuren viser et use-case-diagram af “RecoverPassword“. ”RecoverPassword“ er en funktion som aktøren ”User“, brugeren af applikationen, kan aktiverer ved at trykke på en knap, i tilfælde af at brugeren har glemt sine loginoplysninger til Surftown. Når aktøren ”User“ aktiverer denne funktion, viser applikationen en udfyldelses-form, hvor ”User“ kan indtaste og indsende sin E-mail adresse til Surftown. I givet fald at ”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 svare 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 "getPrivat-Information". 6. Applicationen indikerer overfor **User** om 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 give aktøren "User" mulighed for at se sine private oplysninger, ved angive sin E-mail og password, som er registreret hos Surftow.



Figur 6: Figuren er et sekvensdiagram af "Login" funktionen, hvis use case er afbilledet i figur 5. I denne figur er alle tre forskellige aktøre 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 boundary'en. "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.

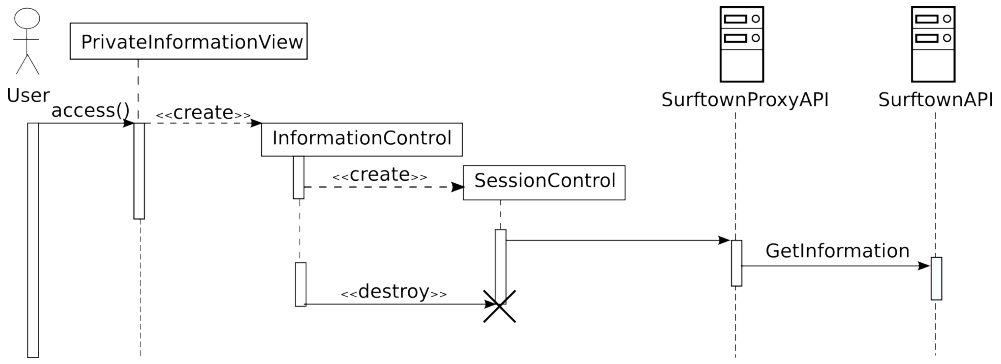
Use case navn	getPublicInformation
Deltagende aktøre	Startet af **User** Kommunikerer med **Surftown**
Event flow	1. **User** åbner applikationen på sin telefon. 2. Applikationen henter de åbne informationer ned fra **Surftown**. 3. Applikationen fremstiller informationerne for **User**.
Resultat	**User** bliver præsenteret for information

Figur 7: Figuren er et use-case-diagram af "GetPublicInformation" funktionen.

Figur 8: Figuren er et sekvensdiagram af "GetPublicInformation" funktionen. "GetPublicInformation" er en meget simpel funktion, hvor aktøren "User" trykker sig ind på et view hvor der skal vises offentlig tilgængelig information (f. eks. kontaktinformation til Surftown), hvorefter der bliver oprettet et "InformationControl" objekt. "InformationControl" objektet laver et kalde til "SurftownProxyAPI" som videresender kaldet til "SurftownAPI"et

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 9: Figuren er et use-case-diagram af "GetPrivateInformation" funktionen. "GetPrivateInformation" funktionen er den funktion som bliver kaldt når brugeren skal se ikke-offentlige tilgængelige informationer (f. eks. informationer om brugeres services hos Surftown)



Figur 10: Figuren er et sekvensdiagram af "GetPrivateInformation" funktionen. Funktionen "GetPrivateInformation" bliver kaldt når aktøren "User" tilgår et view med bruger 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.

4 Systemdesign

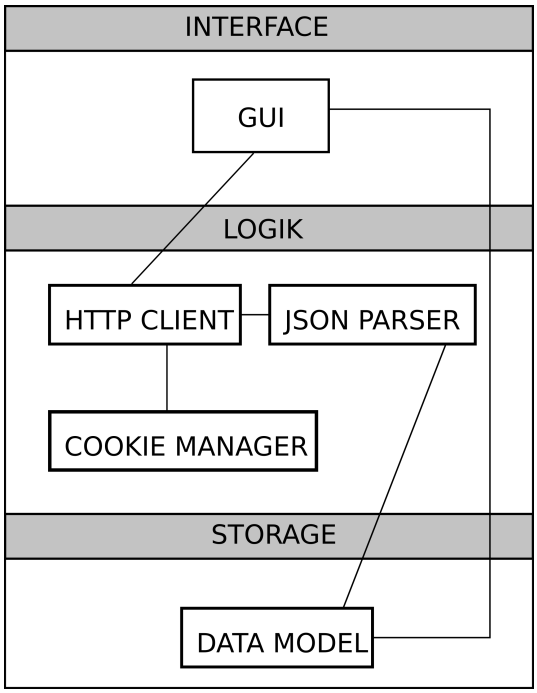
4.1 Surftown API

Surftowns API kaldes via HTTP, og via en bestem GET variable "call" kan man sætte hvilken funktion man vil kalde[2]. Men da Surftowns API er designet til at blive brugt på et lokalt netwærk, understøtter API'et ikke nogle former for handlings- eller informationsrestriktioner[3]. Vi fandt os derfor nødsaget til at lave en form for udvidelse af Surftowns API.
Resultatet blev at vi lavede en form for API proxy server, som er skrevet i Javascript og bliver kørt i NodeJS. Vores API proxy server 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 Surtowns API. Hvis en funktion som parameter kræver et bruger-id, vil API proxy serveren tjekke om der er sendt en gyldig session-cookie med i HTTP-

[2]Afhængig af hvilken funktion man kalder, kan man også sætte andre GET variabler.
[3]Dette var vi ellers blevet lovet af Surftown at de ville lave til os, i forbindelse med udviklingen af dette system

requesten, hvis det er tilfældet vil proxy API serveren sende requesten videre sammen med bruger-id'et associeret med session-cookien.

4.2 Applikationen



Figur 11: Decomposition diagram af applikationen, inspireret af *Object-Oriented Software Engineering Using UML, Patterns, and JAVA*[2]

- **GUI:** GUI'en er implementeret sådan at ved "View" skifte, dvs. ved nyt skærmbillede, requester GUI'en nye data fra HTTP CLIENT'en og fremviser herefter de data der er i DATA MODEL.
- **HTTP CLIENT:** Ved et request fra GUI'en, 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 API'et og sender de modtaget data til JSON PARSER
- **JSON PARSER:** Når JSON PARSER modtager data vil den dekode dataene og sende de dekodet 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 cookies'nes udløbstider
- **DATA MODEL:** DATAMODEL sørger for at udfylde foruddefineret datastrukturer med de nye data. Dette sørger for at data'ene forbliver konsistente overfor GUI'et, som skal fremvise dem.

Manglende implementeringer

I afsnit 3.1 udtrykke vi et ønske om at implementerer de første seks punkter som et minimum, hvilket vi ikke har nået. Vi mangler stadig at implementerer:

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

Derudover mangler vi at implementerer Suftowns officielle design, som vi har sat som en begrænsning i afsnit 3.3.

5 Programtest og resultater

For at være sikker på at vores system er nem og intuitivt at bruge, som er en af vores ikke-funktionelle krav beskrevet i afsnit 3.2, har vi lavet nogle usability test sammen med en test bruger "User"[4]. "User" har allereder en hjemmeside hos en anden hosting leverandør end Surftown, men ser ikke sig selv som en erfaren bruger.
Målet med testene er at finde fejl, hvor systemet opføre 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 idèen med usability test, ikke er at vise at systemet ikke indeholder nogle fejl eller bugs, men at forbedere måden systemet kan bruges på.

[4]Simon Warg's roomate

5.1 Resultat

Brugeren blev informeret om sin rolle og sine mål som han skulle opnå ved at bruge applikationen, som beskrevet i bilag 9.4.

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 blevet spurgt hvor mange services han kunne se. Det første svar vi fik, var et spørgsmål, hvor han spurgte hvad type af service han skulle vælge. Efter en kort forklaring blevet det klart for **User** at han havde 3 services som var listet på skærmen. brugeren nåde 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. Vi gjorde ham 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 in i "Domains" igen. I andet forsøg fandt han ud af at man kunne scrolle for at se de resterende domæne, 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.

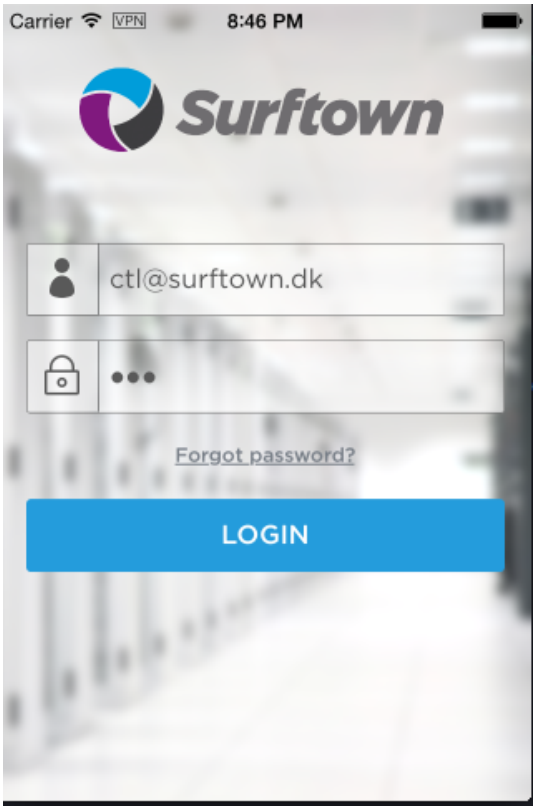
5.2 Resultat konklusion

Vi mener at 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, mener vi ikke at det er noget vi skal ændre

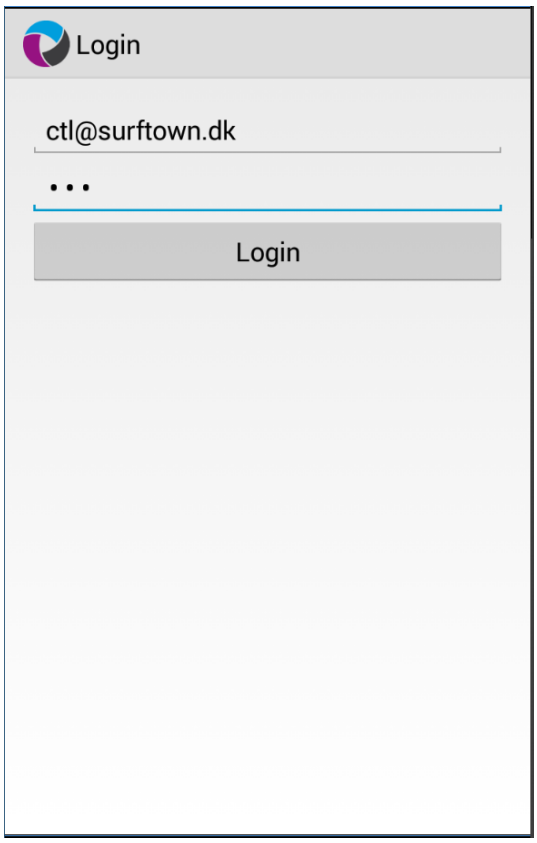
6 Brugergrænseflade og workflow

6.1 Skærmbilleder

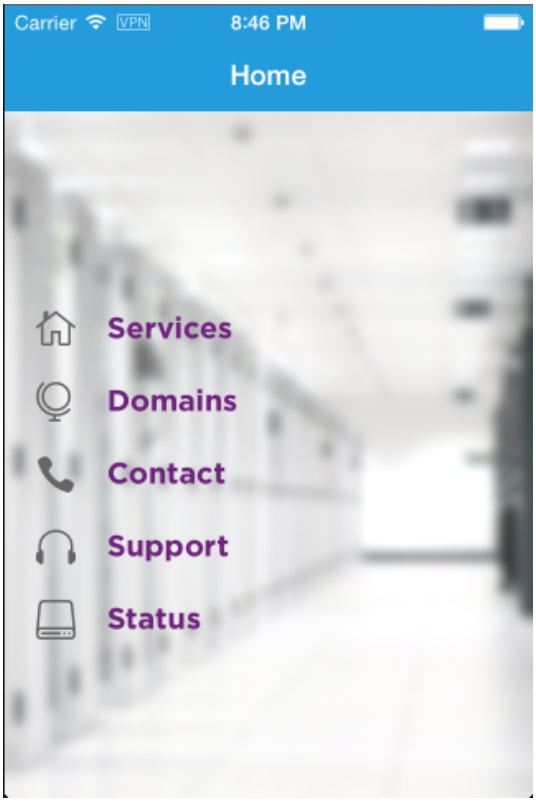
De følgende billeder, er forskellige skærmbilleder af de vigtigste dele af vores applikation på Android og iOS. Dataene som er vist på billederne er data som Surftown har oprettet til os på den test server de har stillet til rådighed for os.



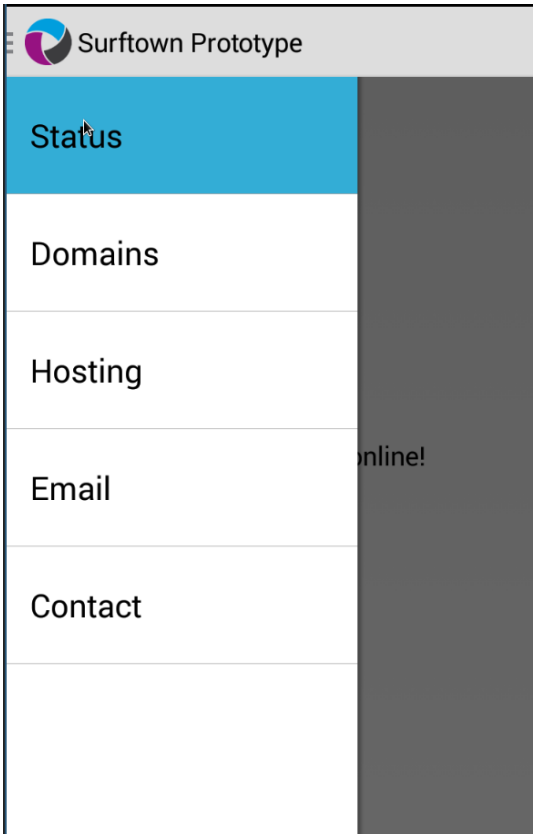
Figur 12: Loginskærm på iOS



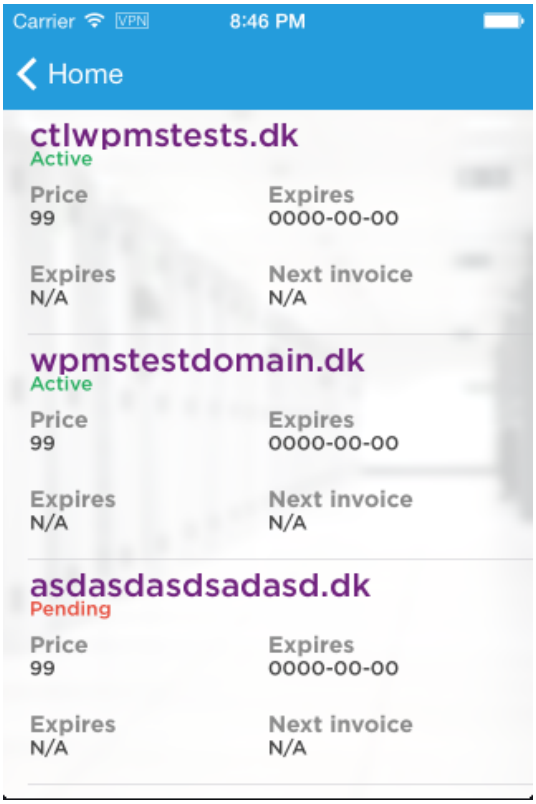
Figur 13: Loginskærm på Android



Figur 14: Menuskærm på iOS



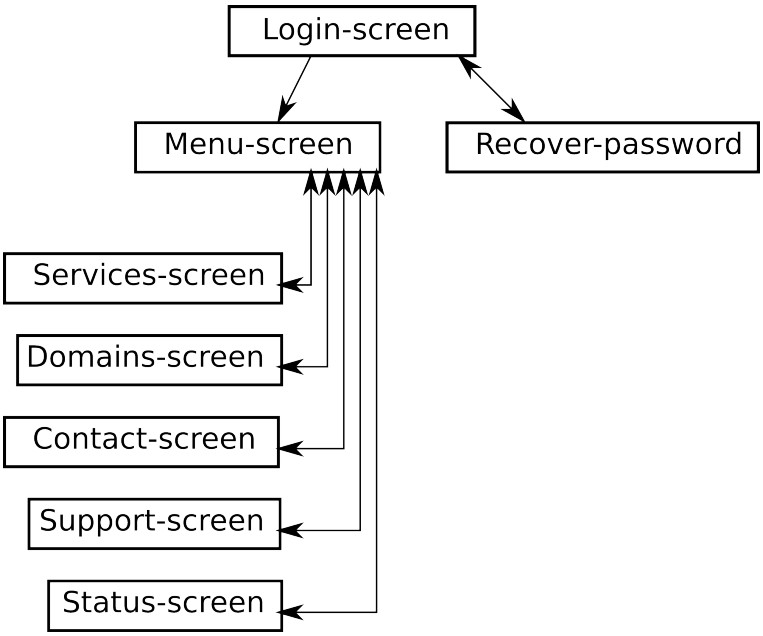
Figur 15: Menuskærm på Android



Figur 16: Domæneoversigtskærm på iOS

Figur 17: Domæneoversigtskærm på Android

6.2 Workflow



Figur 18: Flowdiagram over hovedfunktionerne i Surftown applikationen

På figur 18 kan ses et flowdiagram over hovedfunktionerne i applikationen, hvor vi tager 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 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".

6.3 Audio-visuel præsentation

Vores audio-visuelle præsentation af vores prototype kan findes på følgende link: http://youtu.be/coaXCn-iWJE

7 Versionstyring

Vores vigtigste ændring siden sidst er vores proxy API server, som løser problemet med at Surftown ikke stillede et ordenligt API til rådighed. Vi har dog brugt lidt for lang tid på det, og derved er noget af det andet arbejde blevet forsømt en smule.

7.1 Git opbygning

Vi har oprettet 3 forskellige git repositories; et iOS applikation repository, et Android applikation repository og et shared ressources repository. Det shared ressource repository bliver et sub-git repository til de to andre, så vi ikke har ressourcerne liggende redundant, men mens vi kan holde de to kodebaser adskilt.

8 Projektsamarbejdet

Projektgruppen har fungeret rigtig godt socialt. Vi har haft flere møder hvor vi har diskuteret bla. hvad der skulle laves til næste gang og hvad Surftown har kommunikeret ud[5] Vi har dog også holdt to officielle møder med Surftown igennem projektforløbet, som dog ikke har været særlig effektive, da møderne har været meget ustrukturede. Vi har i gruppen været relativ dårlige til at strukturere vores arbejde og tage referater af vores møder, hvilket har medført at vores projekt har været lidt kaotisk. Dog skal det siges at Surftowns manglende samarbejde også har været en stor faktor for kaoset.

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*

[5] Da to medlemmer af projektgruppen arbejder på Surftown, blev den naturlige måde at kommunkerer med Surftown på igennem dem.

9 Bilag

9.1 Kildekode til proxy API server

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

9.2 Kildekode til Android applikation

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

9.3 Kildekoden til iOS applikationen

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

9.4 Testspecifikationer

9.4.1 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 forklare 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.

9.4.2 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 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 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.

9.5 Git log for Android applikationen

Nicklas Warming Jacobsen Når man logger ind kommer man nu ind på den rigtige activity. Tilføjet toast popup ved login til at give besked om: 1. forkert password/email, 2. Server fejl og 3. netværksfejl.

Nicklas Warming Jacobsen Prof of concept login er nu implementeret

Nicklas Warming Jacobsen Startet på surftown API connection klasse

Nicklas Warming Jacobsen Start på at lave integration til billhost API'et

Robert Rasmussen Finished navigation drawer, and support for headers in nav drawer

Robert Rasmussen Added navigation drawer.

Robert Rasmussen Splash screen added. Started work on application drawer.

Robert Rasmussen Created new prototype with empty activity

Robert Rasmussen Removed prototype

Robert Rasmussen Added prototype project

9.6 Git log for iOS applikationen

Christian Enevoldsen Corrects spacing between navigation bar and tableview in dashboard

Christian Enevoldsen Adds tableview to STDashboardViewController

Christian Enevoldsen Few adjustments to navigationitem style - including title

Christian Enevoldsen Adds UINavigationController

Christian Enevoldsen Fixed couple of warnings. Please check log

Christian Enevoldsen Fixed conflict

Christian Enevoldsen Fixed conflict

Simon Added a TableView layout for HBServices. Extended HBService model to cover more properties

Christian Enevoldsen Fix blocking the UIAlertView

Simon Merge branch 'master' of bitbucket.org:Chrene/surftown-ios-app

Simon Created HBService classes

Christian Enevoldsen added success to HBRequest

Christian Enevoldsen Removed prepareForSegue. Not used

Christian Enevoldsen Removed the scrolling effect from the login screen

Christian Enevoldsen Adds properties for the client data

Christian Enevoldsen removes injection and parse

Christian Enevoldsen removes injection and parse

Christian Enevoldsen Lot's of changes

Christian Enevoldsen Big update.

Christian Enevoldsen Moved content offset behavior to STScrollViewController

Christian Enevoldsen Adds custom scrollview controller for custom behavior

Christian Enevoldsen Tweaks corner radii

Christian Enevoldsen Refactor color names and adds placeholder text color

Christian Enevoldsen Adds more documentation for the STButton and small refactoring

Christian Enevoldsen Adds event blocks to the button

Christian Enevoldsen Refactors button

Christian Enevoldsen Fixes the button texts frame and color

Christian Enevoldsen Adds the username and password textfields, and the submit button, to the login view

Christian Enevoldsen Displays the login screen through the app delegate now

Christian Enevoldsen Deletes storyboard. Gonna go the hardcore way

Christian Enevoldsen Links STLoginViewController to the storyboard. Dismisses keyboard when user taps outside the keyboard while editing

Christian Enevoldsen Creates STLoginViewController

Christian Enevoldsen Creates STLoginViewController

Christian Enevoldsen Changes border width of STTextField

Christian Enevoldsen initial commit