CHALMERS

Sportdate för mobila enheter

Sportdate for mobile devices

Examensarbete inom Data- och Informationsteknik

KRISTOFFER JOHANSSON

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2018

EXAMENSARBETE

Sportdate för mobila enheter
Sportdate for mobile devices

KRISTOFFER JOHANSSON

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2018

Sportdate för mobila enheter
Sportdate for mobile devices
KRISTOFFER JOHANSSON

Examinator: Peter Lundin

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
41296 Göteborg
Telefon: 031-7721000

Institutionen för Data- och Informationsteknik
Göteborg 2018

SAMMANFATTNING

Sportdate är en mötesplats på Internet för personer med ett tränings- och friskvårdsintresse. Webbtjänsten drivs tillsammans av företagen Sportdate AB och SoftIT AB. Tidigare har åtkomsten till tjänsten skett genom en traditionell och en mobilanpassad webbplats. För användare på mobila enheter har det här inneburit att man inte erbjudits en automatisk autentisering. Man har inte heller kunnat ta del av information med hjälp av den mobila plattformens system för notiser. Det befintliga systemet begränsar också möjligheterna att introducera ny typ av funktionalitet som utnyttjar den mobila enhetens GPS-mottagare. För åtgärda problemen med det befintliga systemet och på så sätt kunna erbjuda en förbättrad användarupplevelse har en mobilapplikation utvecklats. Applikationen är konstruerad på ett sätt som möjliggör delning av information med den mobilanpassade webbplatsen för att underlätta underhållet av de båda delsystemen. Ursprungligen var det planerat att det här projektet skulle innefatta utvecklingen av en applikation för både Android och iOS, men då det ganska tidigt under arbetet blev tydligt att detta inte skulle hinnas med inom tidsramen för projektet gjordes valet att det här projektet skulle prioritera färdigställandet av en applikation för iOS. Den här rapporten fokuserar därför enbart på applikationsutveckling för denna plattform. Förutom ett par arbetsdagar på plats i uppdragsgivarens lokaler genomfördes merparten av arbetet med projektet på distans. Applikationen för iOS som färdigställdes under projektet finns idag tillgänglig på App Store, Apples marknadsplats för applikationer under namnet ”Sportdate”.

Nyckelord: iOS, Mobil, Applikation

ABSTRACT

Sportdate is an online meeting place for people with an interest in health and fitness. The web service is run by the two companies Sportdate AB and SoftIT AB. Previously, access to the service has been provided through a traditional and a mobile website. For mobile device users, this has meant that no automatic authentication is offered. You also have not been able to receive information using the mobile platform's notification system. The existing system also limits the possibilities of introducing a new type of functionality that utilizes the mobile device's GPS receiver. To solve the issues with the existing system and offer an improved user experience, a mobile application has been developed. The application is designed in such a way that it allows sharing of information with the mobile site to facilitate the maintenance of the two subsystems. Originally, this project was intended to include the development of an application for both of these platforms, but quite early during the work, it became apparent that this would not be possible within the timeframe for the project, it was then decided that this project would prioritize the completion of the application for iOS. This report, therefore, focuses exclusively on application development for this platform. With the exception of a few days working at the client's office, most of the work on the project was carried out remotely. The iOS application that was completed during the project is available today on the App Store, Apple's Marketplace for Applications under the name ”Sportdate".

Keywords: iOS, Mobile, Application

FÖRORD

Det här projektet utfördes som examensarbete på programmet Datateknik 180 hp på Chalmers Tekniska Högskola under vårterminen 2017. Omfattningen på projektet var 15 hp och arbetet var planerat över 10 veckor. Projektets uppdragsgivare var företaget Sportdate AB tillsammans med SoftIT AB.

Ett stort tack till Jonas Ånestrand på Sportdate AB och Micael Wäxby på SoftIT AB för att ha gett mig möjligheten att utföra det här projektet. Jag vill också tacka Joachim von Hacht för all hjälp och stöd som han har erbjudit i rollen som handledare under arbetet med projektet.

INNEHÅLLSFÖRTECKNING

TERMINOLOGI 1
1. INLEDNING 2
1.1 Bakgrund 2
1.2 Syfte 2
1.3 Mål 2
1.4 Avgränsningar 3
2. METOD 4
2.1 Informationsinhämtning 4
2.2 Implementation 4
2.3 Utvecklingsverktyg 4
3. TEKNISK BAKGRUND 5
3.1 iOS 5
3.2 Swift 5
3.2 Model-View-Controller 6
3.3 Representational State Transfer 7
4. GENOMFÖRANDE 8
4.1 Kravspecifikation 8
4.2 Applikationens integrering i befintligt system 8
4.3 Arkitektur 9
4.4 Design 10
4.4.1 Autentisering 10
4.4.2 Notiser 10
4.4.3 Webbinnehåll 11
4.4.3 Användarflöde 11
4.4.4 Moduler 12
4.5 Lokal webbtjänst 13
4.6 Implementation 14
4.6.1 Autentisering 14
4.6.2 Notiser 14
4.6.3 Webbinnehåll 14
4.6.4 Språkval 15
4.7 Testning och verifiering 16
5. RESULTAT 17

5.1 Autentisering 17
5.2 Notiser 18
5.3 Språkval 20
5.4 Lansering av applikationen 21
6. SLUTSATS 22
7. FÖRSLAG TILL FORTSATT ARBETE 24
7.1 Platsrelaterad funktionalitet med hjälp av GPS-mottagare 24
7.2 Notiser med hjälp av Push 24
7.3 Erbjuda köp av guldmedlemskap i mobilapplikationen 24
REFERENSER 25

TERMINOLOGI

- App Store - En marknadsplats för applikationer på Apples olika plattformar.
- Hemskärm - Den vy som visas för användaren på en upplåst mobil enhet, oftast innehållande ikoner för olika applikationer.
- iTunes Connect - En webbtjänst som används för administration av befintliga samt lanseringen av nya applikationer på App Store.
- Keychain - Ett system som används för att spara lösenord krypterat i iOS.
- Native-applikation - En applikation som är utvecklad för en specifik plattform.
- TestFlight - Ett system som används för beta-testande av iOS-applikationer.
- Xcode - Är den Integrated Development Environment (IDE) som används för att utveckla applikationer för samtliga av Apples plattformar.

1. INLEDNING

1.1 Bakgrund

Sportdate är en mötesplats på Internet för personer med ett tränings- och friskvårdsintresse. Webbtjänsten drivs tillsammans av företagen Sportdate AB och SoftIT AB. Åtkomsten till tjänsten har tidigare skett genom en traditionell och en mobilanpassad webbplats. Som medlem på webbplatsen erbjuds man olika sätt att få kontakt med andra medlemmar. Det finns också funktionalitet för att man skall kunna skapa sin egen blogg. Man kan också få hjälp med kost- och träningsrådgivning.

För att kunna logga in och använda tjänsten från en mobil enhet krävs det att man först besöker webbplatsen i en webbläsare och där anger sina inloggningsuppgifter. Vilket medför att flödet för att få tillgång till tjänsten är uppdelat i flera steg. Som användare erbjuds man inte heller direkt tillgång till applikationen från enhetens hemskärm. Den nuvarande lösningen möjliggör inte heller visning av information med hjälp av den mobila plattformens system för notiser. Man kan inte heller använda enhetens eventuella GPS-mottagare för platsrelaterad funktionalitet.

1.2 Syfte

Syftet med det här projektet är att utveckla en mobilapplikation som åtgärdar de problem som finns med det nuvarande systemet samt möjliggöra en utökning med nya typer av platsrelaterad funktionalitet med hjälp av GPS.

1.3 Mål

Målet är att utveckla en mobilapplikation som innehåller den efterfrågade funktionaliteten och som efter avslutat projekt finns tillgänglig för allmänheten.

Applikationen ska innehålla funktionalitet för att kunna erbjuda användaren en automatisk autentisering.

Applikationen ska kunna visa notiser för användaren vid olika typer av händelser med hjälp av den mobila plattformens system för notiser.

Användaren ska erbjudas möjligheten att se andra användare som befinner sig i närheten på en karta i applikationen.

Ett ytterligare mål med projektet är att undersöka om man kan underlätta underhållet av information i de båda delsystemen (webbplatsen och mobilapplikationen) genom delad information.

1.4 Avgränsningar

Inom projektet kommer enbart en applikation för iOS att utvecklas innehållande den ovan nämnda funktionaliteten. Ursprungligen var det planerat att projektet skulle innefatta utvecklingen av en applikation för både Android och iOS, men då det tidigt under arbetet blev tydligt att det inte skulle hinnas med inom tidsramen för projektet gjordes vissa omprioriteringar. Eftersom uppdragsgivaren hade tillgång till egen personal med kompetens gällande utveckling för Android beslutades det att det här projektet skulle prioritera färdigställandet av en applikation för iOS.

2. METOD

Det här kapitlet beskriver de metoder som användes för att genomföra projektet.

2.1 Informationsinhämtning

Arbetet med projektet inleddes med en analys av det befintliga systemet för att få en bild av hur mobilapplikationen skulle integreras. En kravspecifikation för mobilapplikationen inhämtades från uppdragsgivaren. Innan utvecklingsarbetet påbörjades genomfördes efterforskningar kring hur man utvecklar en applikationen av den här typen enligt praxis. För det här användes den officiella dokumentationen för iOS i så stor utsträckning som möjligt. Under arbetet lästes sedan den dokumentation som krävdes för implementationen av applikationen in kontinuerligt.

2.2 Implementation

Implementationen genomfördes med hjälp av en standardiserad objektorienterad programutvecklingsprocess. Arbetet bedrevs iterativt där små delmål sattes upp och mobilapplikationen arbetades fram stegvis. Det genomfördes även regelbundna möten med uppdragsgivaren för att säkerställa att slutprodukten uppfyllde deras krav. I slutskedet av utvecklingsarbetet genomfördes ett beta-test av mobilapplikationen av en mindre grupp användare för att säkerställa kvaliteten på slutprodukten.

2.3 Utvecklingsverktyg

Utvecklingen av mobilapplikationen skedde i språket Swift och med hjälp av utvecklingsmiljön Xcode. För versionshantering inom projektet användes Git.

3. TEKNISK BAKGRUND

Det här kapitlet ger den nödvändiga tekniska bakgrunden för att kunna förstå den övriga rapporten.

3.1 iOS

iOS är Apples operativsystem för företagets mobila plattformar. Det används idag på iPhone, iPad och finns även som specialanpassade versioner för Apple Watch i form av watchOS och för Apple TV i form av tvOS. iOS bygger likt macOS i grunden på en kärna av Unix-operativsystemet Darwin [1]. Darwin är ett komplett operativsystem med öppen källkod men iOS innehåller ytterligare delar som är patentskyddade.

Cocoa Touch heter den applikationsmiljö i iOS som man utvecklar native-applikationer emot [2]. Ett annat exempel på en applikationsmiljö som finns tillgänglig är Java. Cocoa Touch innehåller en exekveringsmiljö och standard-ramverken som man har tillgång till som utvecklare.

De två vanligaste standard-ramverken som man använder sig av när man utvecklar för iOS heter Foundation och UIKit. Foundation innehåller olika typer av basfunktionalitet som exempelvis datalagring, trådning och nätverksanslutningar. UIKit är det ramverk som används för att utveckla användargränssnitt. Det innehåller olika vy-klasser för exempelvis knappar, textfält och bilder. UIKit innehåller även en klass som heter UIViewController som används för att skapa så kallade “view controllers”. En “view controller” är en klass som hanterar en vy-hierarki samt agerar som en mellanhand mellan användar-input och det underliggande modell-lagret [3]. WebKit är ett ramverk som används för att visa webbinnehåll för användaren.

3.2 Swift

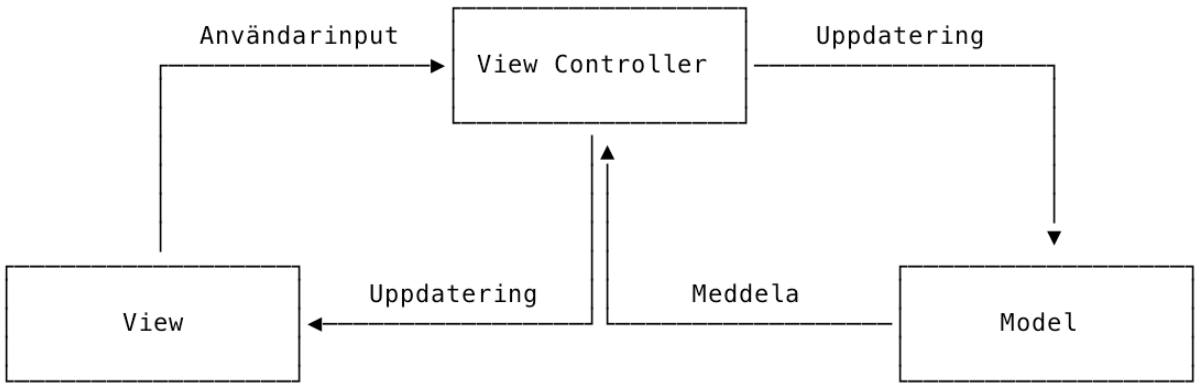
Tidigare har utvecklingen för iOS huvudsakligen skett med hjälp av språket Objective-C. Objective-C är i grundet C med ett extra lager för objektorientering ovanpå. På samma sätt som man kan använda pekare i C som referens för strukturer använder man pekare för objekt i Objective-C. I Objective-C använder något som kallas för “Dynamic Binding” för att få fram vilken implementation som ska användas när en viss metod anropas [4]. Det här genomförs dynamiskt vid exekveringen av ett program och inte vid kompileringen. Sammantaget gör det Objective-C till ett flexibelt och dynamisk språk men det kan också leda till svårupptäckta buggar.

Swift är ett programmeringsspråk som lanserades 2014 vid Apples WWDC. Tanken är att Swift skall ersätta Objective-C som ett säkrare språk. Mycket fokus ligger på typsäkerhet och att kompilatorn skall hjälpa utvecklaren att undvika många problem innan programmet körs. Exekveringsmiljön i iOS är till för Objective-C och alla standard-ramverk är fortfarande skrivna i Objective-C. Därför har man fokuserat mycket på samspelet mellan de två språken för att göra övergången till Swift så enkel som möjligt.

Swift delar en hel del funktionalitet med C# och Java som exempelvis typsäkerhet och generiska klasser. Tillskillnad från de här två språken använder sig inte Swift av någon virtuell maskin som standard utan kompileras till maskinkod. Swift använder inte heller "garbage collection" (GC) för minneshantering. Man använder istället "automatic reference counting" (ARQ). Generellt kan man säga att skillnaden mellan dessa två är att GC körs dynamiskt vid exekvering medan ARQ körs vid kompilering.

3.2 Model-View-Controller

Model-View-Controller (MVC) är ett arkitekturmönster som många av iOS standard-ramverk förväntar sig att man strukturerar sina applikationer efter. Mönstret rekommenderas dessutom av Apple [5].



Figur 3.1. MVC-mönstret i iOS-applikationer.

Ett centralt begrepp inom iOS-utveckling är det som kallas för "view controller". En "view controller" är en klass som hanterar en vy-hierarki samt agerar som en mellanhand mellan användar-input och det underliggande modell-lagret. Allt som syns på skärmen i en iOS-applikation hanteras av en "view controller". Som nämnt ovan i kapitlet om iOS är alla "view controller" subklasser till klassen UIViewController som återfinns i standard-ramverket UIKit. När en applikation startas anger man vilken "view controller" som skall agera "root view controller", vilket är den som visas först. När man sedan navigerar till andra vyer i applikationen "läggs" dessa ovanpå denna första "view controller". Man kan tänka sig det som en sorts stack. Vid en navigation bakåt i stacken försvinner vyn och klassen avallokeras. En "view controller" kan även hantera så kallade "child view controllers".

3.3 Representational State Transfer

Representational State Transfer (REST) beskriver hur kommunikationen skall ske mellan två enheter på Internet. En webbtjänst sägs vara RESTful när den tillhandahåller ett API som är strukturerat enligt REST. REST bygger huvudsakligen på följande principer [6]:

- **Client/Server**: Ansvaret skall vara tydligt separerat mellan klient och server.
- **Stateless**: Servern skall inte behöva lagra någon information relaterad till klienten.
- **Cache**: Servern ska returnera data på ett sådant sett att klienten kan veta om den sedan kan återanvända denna eller inte.
- **Uniform Interface**: Servern skall erbjuda klienten ett API med ändpunkter som är implementerade likt en katalogstruktur. För att motta, skapa, uppdatera och ta bort data ska standard-metoderna för HTML (GET, POST, PUT, DELETE) användas.
- **Layered System**: Klienten skall inte kunna avgöra om kommunikationen sker direkt med servern eller ett mellanliggande lager.

4. GENOMFÖRANDE

I det här kapitlet redogörs för genomförandet av projektet.

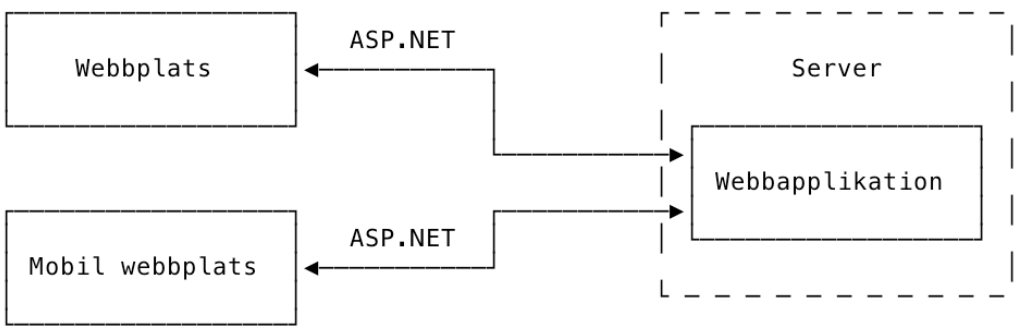
4.1 Kravspecifikation

Funktionaliteten som ska ingå i mobilapplikationen var avsedd för att lösa problemen som nämns ovan i kapitel 1.1. Det ska finnas funktionalitet för att spara användaruppgifter för att kunna erbjuda användaren en automatisk autentisering. Man ska också dra nytta av iOS system för notiser för att kunna meddela användaren om olika saker, och även använda den inbyggda GPS-mottagaren för att både kunna dela med sig av sin egen position och visa positioner för andra användare som befinner sig i närheten. Mobilapplikationen ska också gå att använda på samtliga språk som erbjuds på den nuvarande webbplatsen. Vilket för närvarande är svenska, engelska, norska, danska, finska och tyska. Slutligen ska mobilapplikation efter färdigställandet finnas tillgänglig för allmänheten på App Store, Apples marknadsplats för applikationer.

Uppdragsgivaren vill också att så mycket som möjligt av innehållet i mobilapplikationen ska återanvändas från den mobila webbplatsens webbsidor. Det här begränsar samtidigt arbetet som krävs för att färdigställa projektet då i princip allt innehåll som är synligt för användaren kan återanvändas. Det finns stöd i iOS för att bygga in webbsidor i applikationer som går att använda för det här ändamålet.

4.2 Applikationens integrering i befintligt system

Webbapplikationen är baserad på Microsoft .NET och kommunikationen mellan denna och mobilapplikationen var tänkt att ske genom ett REST-API. Vid inledningen av arbetet med projektet var det här API:et ännu inte färdigställt, utan det var planerat att implementeras i samband med det här projektet. Det som fanns tillgängligt var en specifikation över hur det var tänkt att det skulle se ut när det var färdigt (BIL.1). De ändpunkter som var inkluderade i specifikationen var en som används för användarautentisering och en för hämtning av en användares meddelanden. Tanken var att denna skulle användas för att visa notiser beroende på om en användare hade mottagit ett nytt meddelande. Ändpunkter för att kunna skicka en användares position och hämta positioner för andra användare som befinner sig i närheten var planerade att läggas till i specifikationen vid ett senare tillfälle.

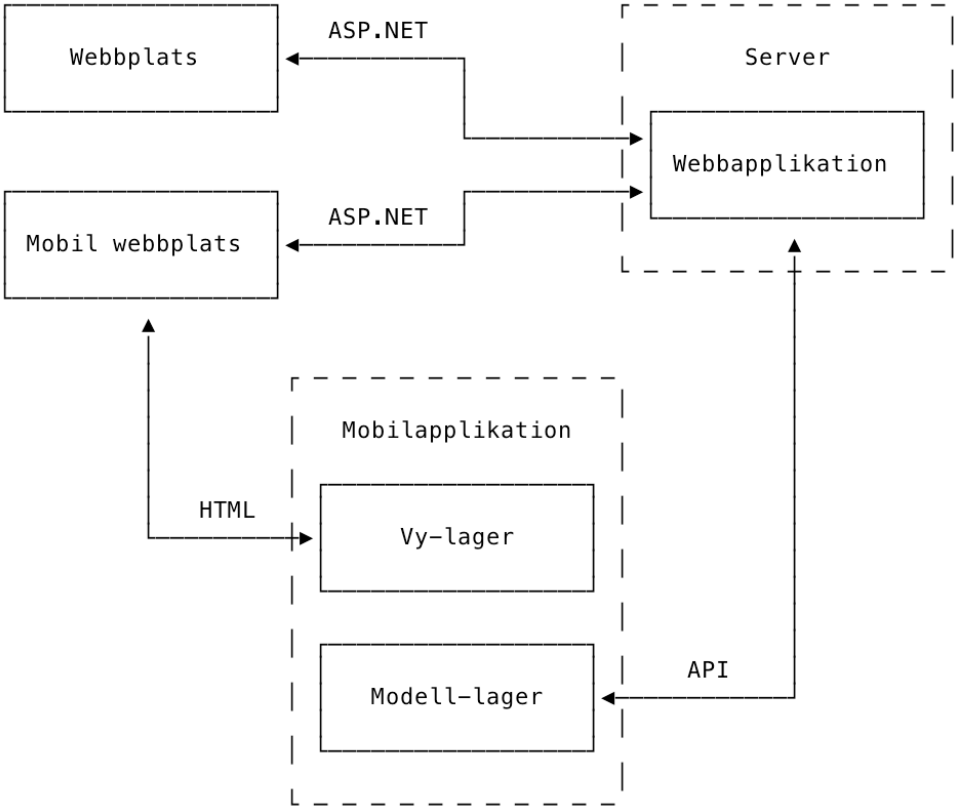


Figur 4.1. Översikt över befintlig webbapplikation.

4.3 Arkitektur

Ett mål med mobilapplikationen är att återanvända och fortsätta använda så mycket som möjligt av innehållet från den mobilanpassade webbplatsen. Samtidigt ska användaren uppleva att man använder en native-applikation och inte att man besöker en webbplats. Det här sammantaget bestämde sedan mycket av de val som gjordes gällande arkitekturen av mobilapplikationen.

Mobilapplikationen är uppdelad i tre olika lager enligt MVC-mönstret. I modell-lagret finns affärslogiken relaterad till kommunikation med webbapplikationens API som används för autentisering, notiser och platsrelaterad-funktionalitet. Här finns också funktionalitet för att spara data permanent mellan exekveringar. Vy-lagret innehåller allting som är synligt på skärmen för användaren. Det här inkluderar både native-komponenter och webbinnehåll som hämtas från den mobila webbplatsen. Funktionalitet som återfinns i det sista lagret, kontroller-lagret används för kommunikationen mellan de två övriga lagren.



Figur 4.2 Webbapplikation med mobilapplikationen adderad.

4.4 Design

I det här kapitlet beskrivs designen av de olika delarna av mobilapplikationen och ett användarflöde för att illustrera samverkan mellan dessa.

4.4.1 Autentisering

Eftersom i stort sett all funktionalitet i applikationen är beroende av att en användare är autentiserad var det mest logiskt att påbörja arbetet med denna delen först.

Säkerhet är en viktig aspekt när det kommer till autentisering både när det gäller lagringen av inloggningsuppgifter på enheten men även kommunikationen mellan mobilapplikationen och servern behöver ske på ett säkert sätt.

För att kunna lagra inloggningsuppgifter säkert finns det funktionalitet i iOS för att kunna lagra data i krypterad form. När det kommer till kommunikationen mellan applikationen och servern stod valet mellan att kryptera uppgifterna innan de skickades till servern eller att förlita sig på Hypertext Transfer Protocol Secure (HTTPS). I inledningen av arbetet med projektet erbjöds inte krypterad åtkomst genom HTTPS till varken servern eller webbplatsen. Uppdragsgivaren hade planer på att implementera det vid ett framtida tillfälle. Apple har indikerat att man någon gång framöver kommer att kräva att all nätverkskommunikation i applikationer för iOS sker över HTTPS [7]. Att implementera en egen säker lösning för kryptering av kommunikationen var också något som skulle innebära mycket extra arbete. Det här sammantaget med att man från uppdragsgivarens sida gick med på att ha funktionaliteten som krävdes för att kunna erbjuda åtkomst till servern över HTTPS färdig innan applikationen skulle lanseras för allmänheten, gjorde att valet hamnade på att förlita sig på HTTPS när det kommer till säker kommunikation mellan mobilapplikationen och servern.

4.4.2 Notiser

För att implementera ett system som levererar en notis till en iOS-applikation när någonting sker på en server använder man ett system som heter Apple Push Notification service (APNs). Det här kräver en del arbete på serversidan då man behöver sammankoppla denna med Apples servrar som i sin tur sedan sköter leveransen av en notis till en specifik enhet [8]. Då det här inte var någonting som uppdragsgivaren ansåg sig att ha tid att implementera inom tidsramen för projektet valdes en annan lösning för det här ändamålet.

iOS erbjuder funktionalitet för att hämta data från Internet i bakgrunden. Det här innebär att man kan registrera en applikation för den här funktionaliteten, operativsystemet väljer sedan lämpliga tidpunkter för att “väcka” applikationen i bakgrunden och den ges sedan en begränsad tid för att hämta data från Internet och processa denna [9]. Applikationen kan på så sätt hämta en användares meddelande från servern när den befinner sig i bakgrunden och finns det ett nytt meddelande visar den en notis på enheten.

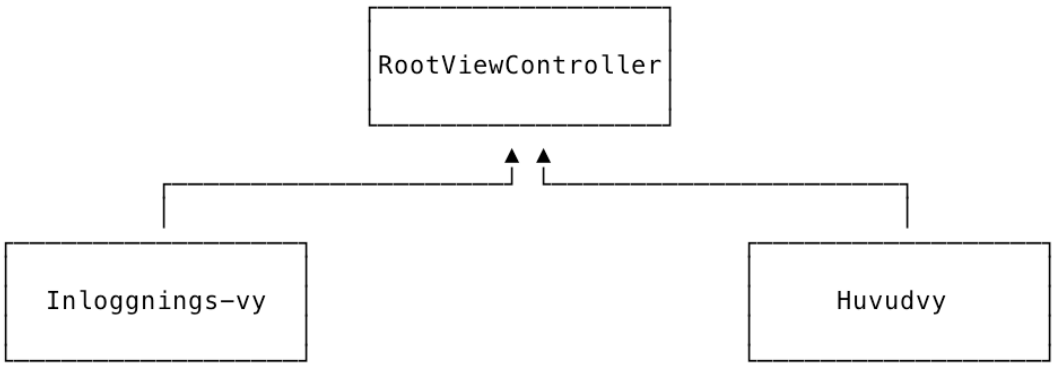
4.4.3 Webbinnehåll

En vanlig layout för en iOS-applikation är att man har en meny i nederkant innehållande knappar för att navigera mellan olika delar av applikationen. I toppen har man en vy innehållande titel och eventuella knappar på sidorna för att navigera framåt eller bakåt i vy-hierarkin i den aktuella delen. Det största området i mitten upptas sedan av själva innehållet i applikationen. Även om det mesta av innehållet i den här mobilapplikationen hämtas från den mobila webbplatsen var ett mål med applikationen att den ska upplevas som en native-applikation. Därför gjordes valet att använda den ovan nämnda layouten för applikationen, men istället för byta mellan olika vyer beroende på menyvalet används en webb-vy där man byter vilken URL som ska visas istället.

En av fördelarna med det här är att eftersom innehållet delas mellan mobilapplikationen och webbplatsen behöver ändringar bara göras på ett ställe. Man kan också uppdatera innehållet i mobilapplikationen utan att behöva kompilera om denna och lansera en ny version på App Store. Uppdragsgivaren blir inte heller beroende av en iOS-utvecklare för att kunna göra ändringar av innehållet.

4.4.3 Användarflöde

När mobilapplikationen startas visas en “root view controller” som kan befinna sig i två tillstånd “inloggad” eller “utloggad”. Tillståndet sätts beroende på om det finns några lagrade inloggningsuppgifter eller ej. Är tillståndet lika med “inloggad” visas en “view controller” innehållande huvudvyn i applikationen. Är det istället lika med “utloggad” då visas en “view controller” innehållande inloggnings-vyn. På samma sätt så skiftas det sedan mellan dessa två vyer när en användare befinner sig i huvudvyn och väljer att logga ut eller tvärtom vid en korrekt autentisering vid vyn innehållande inloggningen.



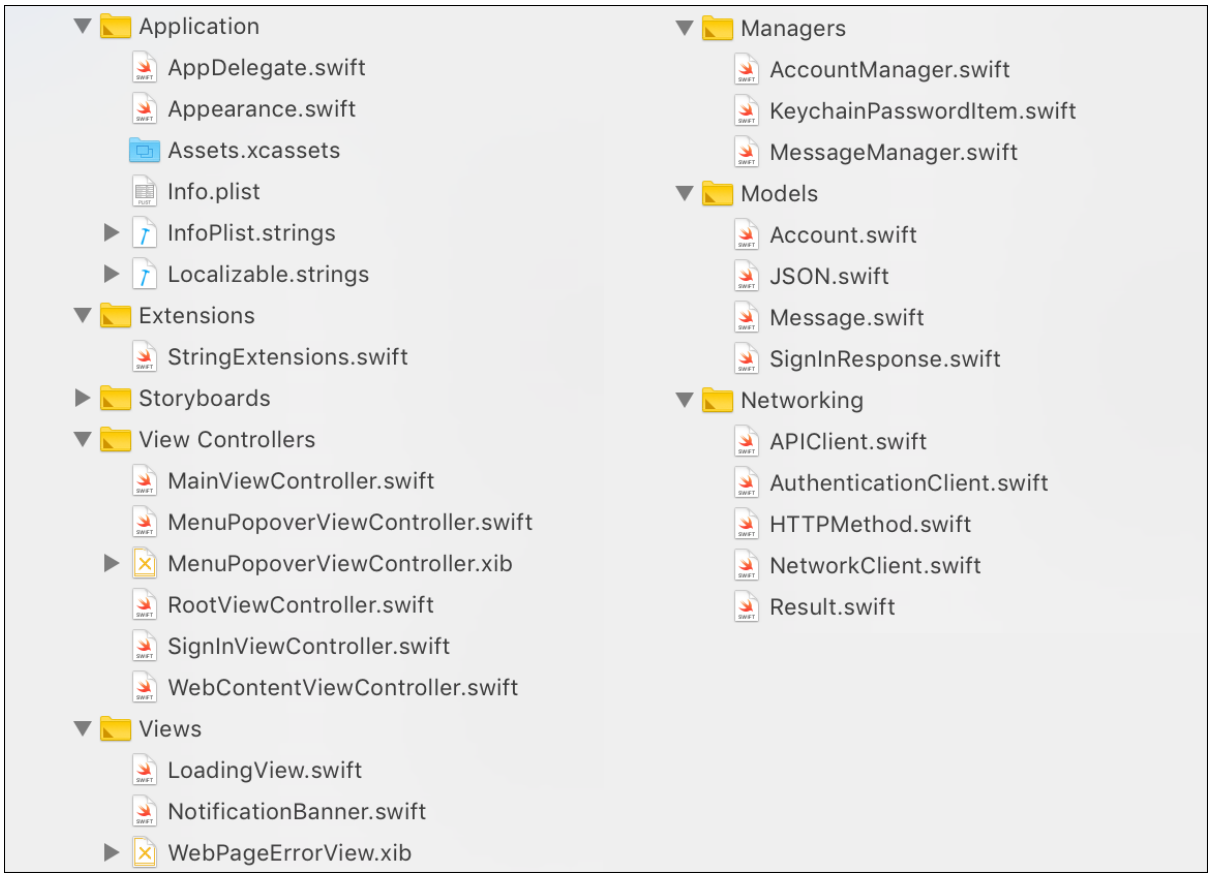
Figur 4.3

Befinner sig användaren vid inloggnings-vyn och väljer att logga in kontrolleras de ifyllda uppgifterna mot webbapplikationens API. Om de är korrekta sparas de på enheten och huvudvyn visas för användaren. Väljer sedan användaren att logga ut raderas inloggningsuppgifterna och inloggnings-vyn visas igen.

När mobilapplikation befinner sig i bakgrunden på enheten kontaktar den med jämna mellanrum servern för att kontrollera om den inloggade användaren har mottagit några nya meddelanden. Har den det visas en notis på enheten.

4.4.4 Moduler

Figuren ned visar modulerna som den färdiga applikationen innehåller. Modulerna till vänster tillhör vy-lagret medan de till höger tillhör modell-lagret i applikationen.



Figur 4.4

4.5 Lokal webbtjänst

Eftersom applikationen behövde kunna kommunicera med webbtjänstens API och det inte var tillgängligt när utvecklingsarbetet skulle påbörjas gjordes valet att sätta upp en lokal webbtjänst med ett API baserat på specifikationen (BIL.1). Det här var också tänkt att underlätta den lokala utvecklingen genom att man varken var beroende av en anslutning till internet eller webbtjänsten för att kunna arbeta med projektet. Man får också möjligheten att testa mobilapplikationen mot en miljö som man kontrollerar innan man gör anrop mot den riktiga webbtjänsten som används i produktion, vilket eliminerar risken att denna på något sätt skulle kunna påverkas negativt.

För att sätta upp den lokala webbtjänsten användes Node.js tillsammans med ramverket Express.js. Valet av denna teknik gjordes på grund av att det fanns tidigare erfarenheter av att ha arbetat med denna och eftersom att webbtjänsten enbart skulle användas för testning under utvecklingsarbetet var det viktigaste att det skulle gå snabbt och vara enkelt att komma igång med.

4.6 Implementation

Det här kapitlet beskriver implementationen av de olika delarna i mobilapplikationen.

4.6.1 Autentisering

För att lagra inloggningsuppgifterna gjordes valet att spara lösenordet krypterat med hjälp av iOS standardsystem för det här ändamålet, kallat Keychain [10]. Användarnamnet och andra inte lika känsliga uppgifter lagras med hjälp av UserDefaults som är en klass i standard-ramverket Foundation. UserDefaults erbjuder ett enkelt gränssnitt för att spara så kallade nyckel-värdespar permanent mellan exekveringar av en applikation.

Mobilapplikationen behöver kunna kommunicera med fler API-ändpunkter än bara den avsedd för autentisering. För att inte behöva upprepa mycket av samma kod på flera ställen skapades en basklass som inkapslar funktionaliteten relaterad till nätverksanslutningar. Sedan skapades en subklass av denna som innehöll funktionalitet som är specifik för ändpunkten som används för autentisering.

4.6.2 Notiser

Hämtningen av meddelande genomförs med hjälp av en subklass av den tidigare skapade basklassen som nämns ovan.

Från uppdragsgivaren sida ville man ha möjligheten att kunna visa två olika typer av notiser. Den ena är för att visa notiser när en användare har mottagit ett nytt meddelande från en annan medlem. Den andra typen är till för att administratörerna på webbplatsen ska ha möjligheten att skicka ut ett meddelande till alla användare vid exempelvis ett planerat underhåll av tjänsten och liknande.

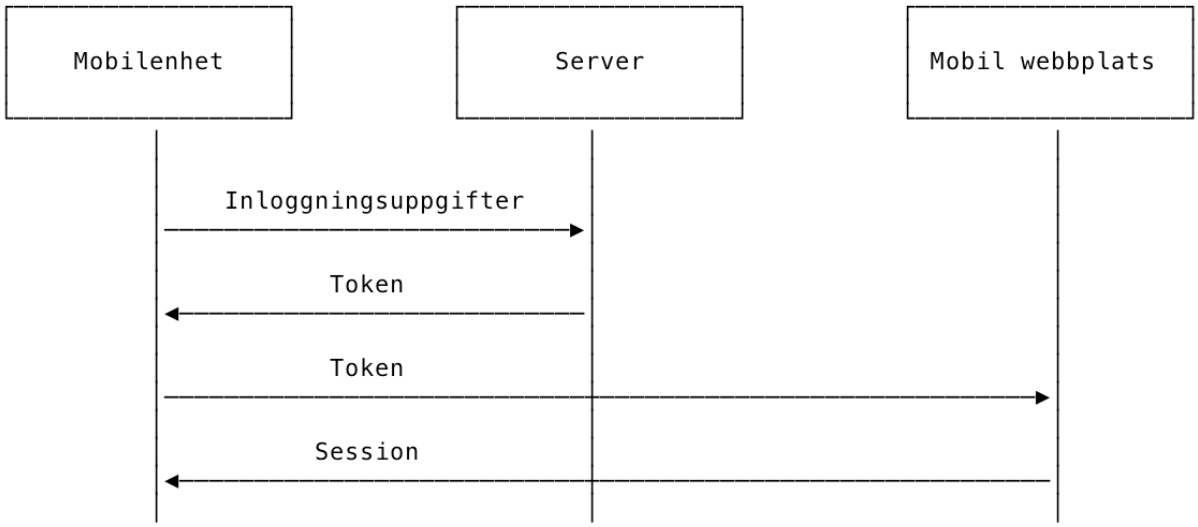
För den första typen första typen används iOS standardsystem för notiser. Den andra typen av notiser visas med hjälp av ett eget implementerat system. När man har mottagit ett meddelande av den här typen sparas innehållet med hjälp av klassen UserDefaults. Vid nästa tillfälle som mobilapplikationen startas så visas sedan innehållet för användaren med hjälp av en egenskriven vy-klass i nederkant av skärmen.

4.6.3 Webbinnehåll

För att visa webbaserat innehåll i en iOS-applikation använder man sig av en vy-klass WKWebView som återfinns i standard-ramverket WebKit. WKWebView fungerar i princip som en webbläsare på så sätt att man tillhandahåller en URL som man vill visa och denna laddas sedan in och visas för användaren. Man erbjuds också möjligheten att skapa metoder för olika händelser som exempelvis när innehållet har laddats färdigt eller om man dirigeras om till en annan URL.

En av utmaningarna med den här lösningen var relaterade till autentiseringen. Autentiseringen i mobilapplikationen kontrollerar om inloggningsuppgifter är korrekta med hjälp av web-

bapplikationens API. Webbplatsens autentisering är istället sessions-baserad vilket gör att det krävs en giltig session för att kunna hämta och visa de relevanta webbsidorna för en autentiserad användare. Det här löstes genom att man vid en lyckad autentisering i applikationen motar en token som sedan bifogas med en specifik URL när webb-vyn visas för första gången. Är det en giltig token så ges användaren en session som sedan är aktiv.



Figur 4.5 Flöde över autentiseringen.

En annan utmaning var att webbsidorna som hämtas från den mobila webbplatsen redan innehåller en titel och meny. Eftersom det i mobilapplikationen används native-komponenter för den här funktionaliteten behövdes de här tas bort på något sätt. Uppdragsgivaren ville undvika att ha specifika webbsidor enbart avsedda för mobilapplikationen, man ville kunna dela innehållet mellan den mobila webbplatsen och mobilapplikationen. Lösningen på det här blev att använda funktionalitet i webb-vyn som används som möjliggör att man kan exekvera JavaScript på innehållet som visas. Efter det att en specifik webbsida har lästs in körs ett enkelt JavaScript som döljer den del i webbsidan som innehåller menyn och titeln.

4.6.4 Språkval

Översättningen av texten i en applikation för iOS fungerar på det sättet att man utvecklar i ett så kallat bas-språk, som i det här fallet är engelska. Sedan inkluderar man specifika filer för de önskade språken där man anger vad olika strängar i bas-språket ska bytas ut mot. Operativsystemet visar sedan applikationen automatiskt på det språket som användaren har inställt på sin mobila enhet.

4.7 Testning och verifiering

Efter ca halva projektiden hade gått genomfördes ett halvtidsmöte tillsammans med uppdragsgivaren. En tidig version av applikationen visades upp för att ge en bild av resultatet vid den här tidpunkten.

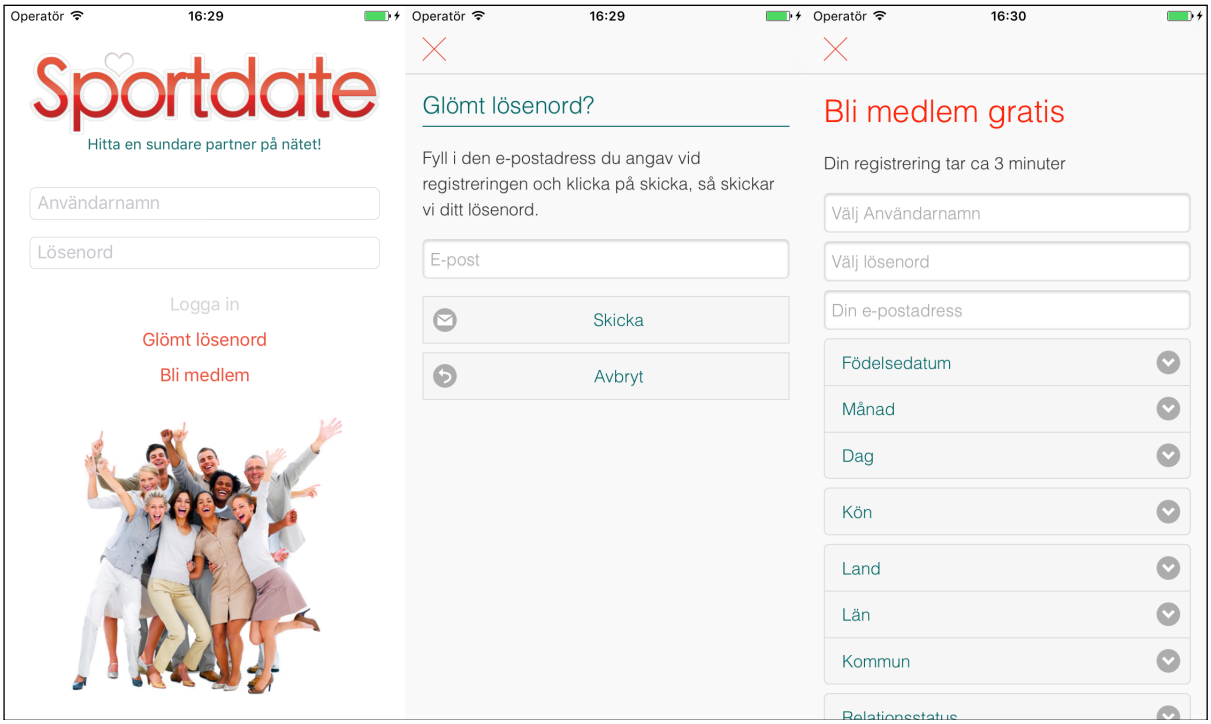
I slutskedet av utvecklingsarbetet genomförde uppdragsgivaren test av en betaversion av applikationen på deras egna mobila enheter. Det här för att säkerställa kvaliteten på slutprodukten innan den lanserades för allmänheten. För att genomföra testen användes ett system som heter TestFlight. TestFlight är ett verktyg som ingår i Itunes Connect, Apples webbtjänst för att lansera och administrera applikationer på App Store. Arbetsgången för testet såg ut på följande sätt. Testarna gav återkoppling på funktionalitet i applikationen, var det något som behövde ändras på åtgärdas det och en ny version skickades ut. Så här fortsatte det sedan tills det att samtliga parter var nöjda med funktionaliteten i applikationen.

5. RESULTAT

En applikation för iOS färdigställdes under projektet och finns idag tillgänglig på App Store under namnet ”Sportdate”. I det här kapitlet redogörs mer specifikt för funktionen av den.

5.1 Autentisering

När man som användare startar applikationen för första gången så möts man av en inloggnings-vy där man kan ange sitt användarnamn och lösenord för att logga in. Denna vy erbjuder också användaren valet att registrera ett nytt konto om man inte är medlem sedan tidigare samt att få sitt lösenord skickat till sig.

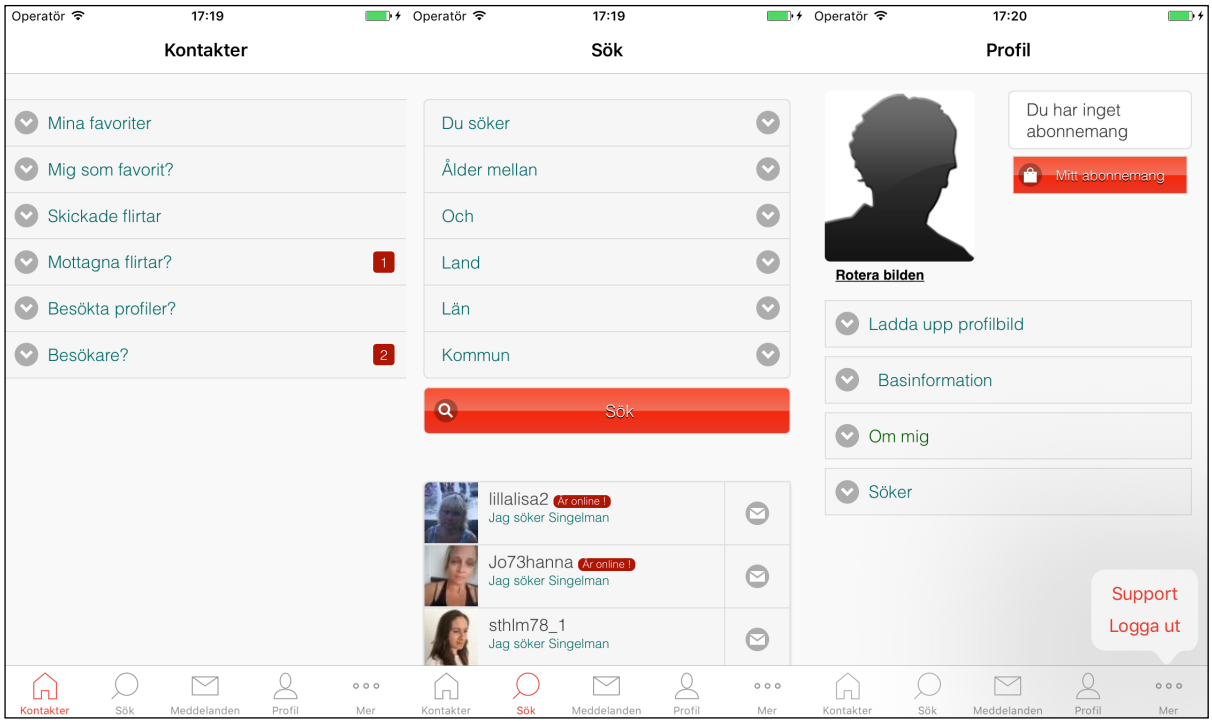


Figur 5.1. De val som är tillgängliga första gången applikationen startas.

Vid en lyckad inloggning så sparas inloggningsuppgifter på enheten och huvudvyn visas. Nästa tillfälle när applikationen startas loggas användaren in automatiskt med hjälp av de sparade uppgifterna och huvudvyn i applikationen visas direkt. Det här fortsätter tills det att antingen användaren väljer att logga ut varefter de sparade inloggningsuppgifterna raderas från enheten eller att inloggningsuppgifterna slutar att vara korrekt, vilket exempelvis kan ske om lösenordet har ändrats genom webbplatsen.

Huvudvyn i applikationen består av tre stycken vyer, en meny i nederkant för navigation, en vy för visande av innehåll i mitten och en vy innehållande titeln för det aktuella innehållet. När man som användare väljer ett av menyvalen hämtas det relaterade innehållet från den mobila webbplatsen och visas sedan i innehålls-vyn. De menyval som är tillgängliga är följande:

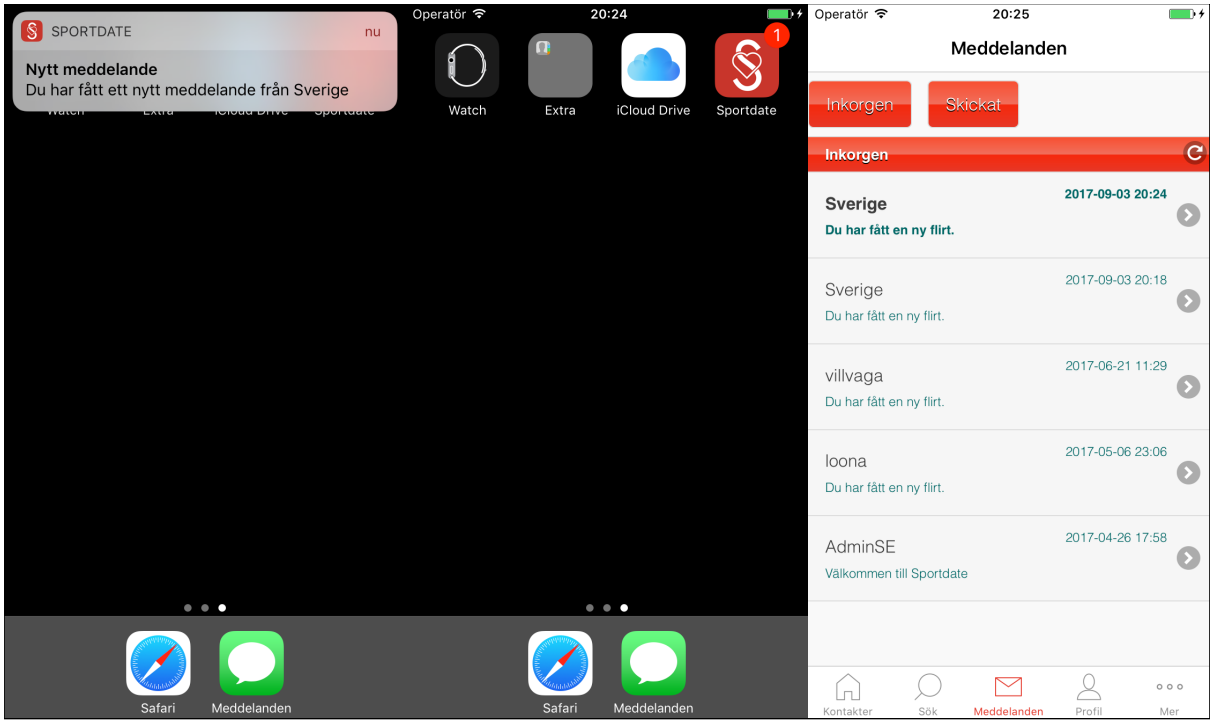
- **Kontakter**: visar olika typer av listor relaterade till en användares interaktion med andra användare.
- **Sök**: visar en vy där man erbjuds möjligheten att söka efter andra användare.
- **Meddelanden**: visar användarens inkorg för meddelanden på tjänsten.
- **Profil**: visar användarens profil. Här går det även att uppdatera profilen och ändra profilbild.
- **Support**: visar ett formulär som går att använda för att kontakta personalen bakom tjänsten.
- **Logga ut**: loggar ut användaren och raderar samtidigt hans eller hennes sparade inloggningsuppgifter.



Figur 5.2. Applikationens huvudvy.

5.2 Notiser

Applikationen innehåller funktionalitet för att kunna visa två typer av notiser. Den första typen är när en användare mottar ett meddelande från en annan medlem. För den här typen av notiser använder sig applikationen av iOS standardsystem för notiser. En notis visas och det läggs även till ett rött märke på applikationens ikon på hemskärmen för att indikera att man har en ny notis. Väljer användaren att röra vid notisen öppnas applikationen och man tas automatiskt till vyn innehållande inkorgen med meddelanden.

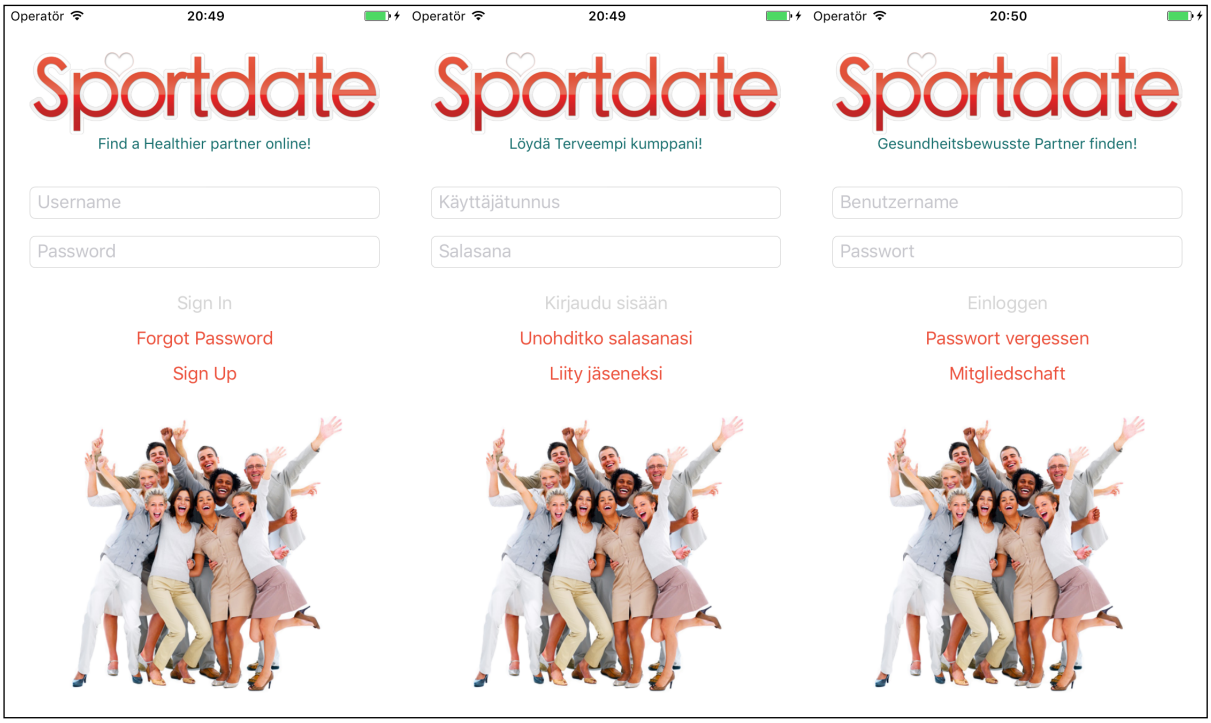


Figur 5.3. Visande av notis med hjälp av standardsystem i iOS.

Den andra typen av notiser innehåller meddelanden som administratörerna på webbtjänsten kan välja att skicka ut till samtliga användare. Ett exempel på ett sådant meddelande kan vara att man planerar underhåll av tjänsten och den kommer inte vara tillgänglig vid en viss tidpunkt. Notiser av den här typen visas för en användare nästa gång som applikationen startas med hjälp av en vy i nederkant av skärmen som försvinner antingen genom att man rör vid den eller efter ett visst tidsintervall.

5.3 Språkval

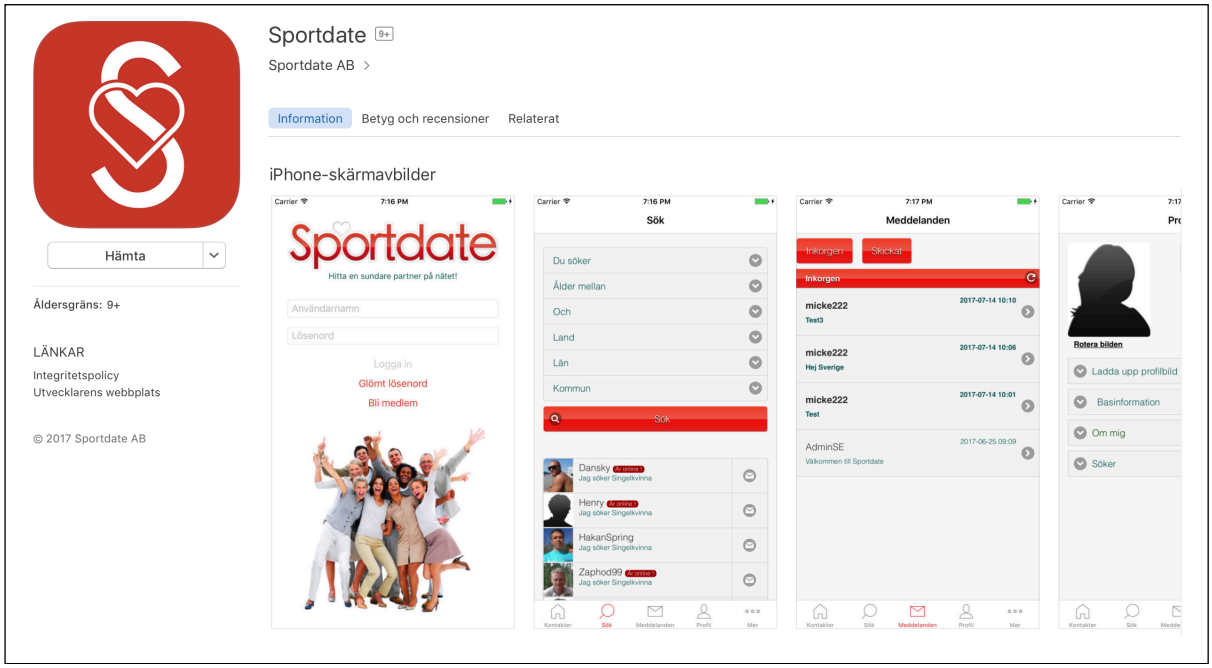
Den slutgiltiga versionen av applikationen innehåller stöd för att kunna användas på samtliga av de språk som webbplatsen stöder, svenska, engelska, norska, danska, finska och tyska.



Figur 5.4. Inloggnings-vyn med den mobila enhetens språk inställt på engelska, finska och tyska.

5.4 Lansering av applikationen

Applikationen har genomgått en granskning av Apple varefter den blev godkänd och är sedan den 24 augusti 2017 tillgänglig för allmänheten på App Store.



Figur 5.4. Applikationens sida på App Store.

6. SLUTSATS

Syftet med det här projektet var att utveckla en mobilapplikation innehållande funktionaliteten från kravspecifikationen tillgänglig för allmänheten på App Store. Den slutliga mobilapplikationen innehåller all funktionalitet från kravspecifikationen som var möjlig att implementera i systemet inom tidsramen för projektet. Funktionaliteten relaterad till GPS gick inte att inkludera eftersom uppdragsgivaren inte hade implementerat de ändpunkter som krävs i webbtjänstens API. Mobilapplikationen är idag tillgänglig på App Store. Det här gör att projektet sammanfattningsvis bör ses som lyckat.

När det kommer till den funktionalitet som är inkluderad fungerar den sett utifrån förutsättningarna väl. Autentiseringen av användare är implementerad på ett enkelt och säkert sätt. Användarens lösenord krypteras på enheten och all kommunikation med webbtjänsten relaterad till autentiseringen sker över en krypterad anslutning med hjälp av HTTPS. Som uppdragsgivaren implementerat stöd för i slutfasen av projektet.

I nuläget använder sig applikationen av ett system för så kallade “bakgrundshämtningar” för att visa notiser när användaren har mottagit ett nytt meddelande. Mobilapplikationen kontaktar med jämna mellanrum webbtjänsten för att se om den inloggade användaren har mottagit några nya meddelanden och har den det visas en notis. Det här gör att mobilapplikationen måste vara kvar i bakgrunden, d.v.s. att om användaren har avslutat den levereras det inte längre några notiser. Det går inte heller som utvecklare att styra när eller hur ofta de här hämtningarna sker, utan operativsystemet väljer själv lämpliga tider för att genomföra det här beroende på olika faktorer. Det här leder till att tiden mellan det att ett meddelande har mottagits och att en notis visas ibland kan variera ganska mycket. Ett bättre sätt vore att använda Apples system för så kallade push-notiser för det här ändamålet. Det här kräver dock en hel del arbete från uppdragsgivarens sida då webbtjänsten behöver kopplas samman med Apples servrar och det här var inte någonting som de ansåg sig ha tid till att genomföra under det här projektet.

Den färdiga mobilapplikationen visar på att alla genomförbara mål relaterade till funktionalitet har genomförts i projektet. När det kommer till frågeställningen gällande delande av information mellan webbplatsen och mobilapplikationen i syfte av att underlätta framtida underhåll, har projektet visat på att det är möjligt att dela mycket av informationen mellan de två delsystemen genom att använda webbsidor från webbplatsen för mycket av innehållet i mobilapplikationen.

Eftersom mycket av innehållet i mobilapplikationen hämtas från den mobila webbplatsen är användarupplevelsen väldigt beroende av hur bra internetanslutning som finns tillgänglig. Det här är något som hade gått att förbättra genom att implementera mer av innehållet i mobilapplikationen med hjälp av native-komponenter. I och med att ett av de ursprungliga önskemålen från uppdragsgivaren var att man skulle använda så mycket som möjligt av innehållet från den mobila webbplatsen samt den begränsade tiden som fanns tillgänglig för projektet var det här inte något som var möjligt att göra i projektet.

Arbetsgången med projektet fungerade generellt bra. Även om en del av den tekniska kompetensen som krävdes för projektet fanns sedan tidigare hos den som utförde projektet var det svårt att på förväg uppskatta hur mycket tid som skulle behövas för att implementera olika typer av funktionalitet. Det här är nog något som är ganska vanligt i mjukvaruprojekt, men även en brist på erfarenhet av arbete inom området kan ha varit en bidragande faktor. Projektet innebar också en fördjupning inom iOS-utveckling på det sättet att information i form av dokumentation för bibliotek och funktionalitet i operativsystemet behövde inhämtas och läsas in för att kunna genomföra utvecklingsarbetet. Även efterforskningar kring hur man enligt praxis utvecklar en applikation av den här typen genomfördes i början av utvecklingsarbetet.

Arbetet med projektet utfördes av en person och det mesta av det genomfördes på distans. Distansarbete ger en rad fördelar som exempelvis att det går att välja sin egen arbetsplats och arbetstider. Det ställer dock höga krav på eget ansvar och planeringen av arbetet. Planeringen av olika delmål i projektet var någonting som hade kunnat fungera bättre. Det här sammantaget med att det var ganska långt mellan kontakterna med uppdragsgivaren gjorde arbetet lite ostrukturerat. Ett bättre sätt hade varit att satt upp delmål för varje vecka och sedan följa upp dessa med uppdragsgivaren kontinuerligt.

Det som har varit mest lärorikt med att genomföra projektet var att lära sig att vara flexibel när man arbetar fram en lösning tillsammans med andra. Man kan ha haft en idé om hur man vill lösa olika saker från början men för att få det att fungera har det behövts göra ändringar med kort varsel. Ett exempel på det här inom projektet var att webbapplikationens API implementerades sent i projektet. Det stämde inte heller helt överens med det API som ingick i ursprungliga specifikationen. Eftersom utvecklingen av mobilapplikationen baserades på den ursprungliga specifikationen behövde man göra en hel del ändringar för att få applikationen att fungera tillfredställande.

När det kommer till hållbara aspekter är applikationen konstruerad på ett sådant sätt som möjliggör delande av information mellan applikationen och webbplatsen. Det här underlättar underhållet av information och minskar på så sätt arbetet som krävs för det här.

7. FÖRSLAG TILL FORTSATT ARBETE

I det här kapitlet ges ett par förslag på funktionalitet som skulle innebära en förbättring av mobilapplikationen vid ett fortsatt arbete.

7.1 Platsrelaterad funktionalitet med hjälp av GPS-mottagare

Den här funktionaliteten var med i den ursprungliga kravspecifikationen för projektet och den gick ut på att en användare skulle ha möjligheten att kunna se andra användare som befinner sig i närheten på en karta i applikationen. Eftersom man från uppdragsgivarens sida inte hann med att implementera de ändpunkter i API:et som krävdes inom tidsramen för projektet, var det här något som inte var möjligt att genomföra.

7.2 Notiser med hjälp av Push

Som nämnt tidigare i tidigare kapitel 4.4.2 har notiser med hjälp av Push en rad fördelar jämfört med den lösningen som används i den utvecklade mobilapplikationen i dagsläget. Då det mesta av utvecklingsarbetet för att kunna använda det istället ligger på servern och då det här inte var något som uppdragsgivaren ansåg sig ha tid med inom ramen för projektet gick det inte att implementera.

7.3 Erbjuda köp av guldmedlemskap i mobilapplikationen

På webbplatsen erbjuds användare att betala för ett guldmedlemskap. Vilken ger olika fördelar gentemot ett traditionellt medlemskap och det är genom det här som uppdragsgivaren gör en del av sina intäkter. Apple tillåter inte att man erbjuder användare att köpa tillgång till funktionalitet utanför App Store om man inte samtidigt erbjuder köp med hjälp av deras system i iOS. Den här funktionalitet innefattar mycket arbete både på serversidan och i mobilapplikation. Vilket gjorde det till något som inte gick att genomföra inom tidsramen för projektet.

Denna funktionaliteten hade möjliggjort att man kunde göra reklam för fördelarna med guldmedlemskapet i applikationen. Man hade också kunnat begränsa funktionalitet för icke-betalande medlemmar som man i dagsläget gör på webbplatsen. På sikt hade det eventuellt kunnat leda till ett ökat antal medlemmar som betalar för guldmedlemskap och på så sätt ökade intäkter för uppdragsgivaren.

REFERENSER

[1] Kernel Architecture Overview [Internet] Apple Developer, Apple Inc, 2013, https://developer.apple.com/library/content/documentation/Darwin/Conceptual/KernelProgramming/Architecture/Architecture.html (Acc 171126)

[2] Cocoa (Touch) [Internet] Apple Developer, Apple Inc, 2015, https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Cocoa.html (Acc 171126)

[3] View Controller Programming Guide for iOS: The Role of View Controllers [Internet]. Apple Developer, Apple Inc, 2015, https://developer.apple.com/library/content/featuredarticles/ViewControllerPGforiPhoneOS/ (Acc 171004)

[4] Dynamic binding [Internet]. Apple Developer, Apple Inc, 2015, https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/DynamicBinding.html (Acc 171203)

[5] Model-View-Controller [Internet]. Apple Developer, Apple Inc, 2015, https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html (Acc 170827)

[6] Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST) [Internet] Roy Thomas Fielding, 2000, https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (Acc 171211)

[7] Supporting App Transport Security [Internet]. Apple Developer, Apple Inc, 2016, https://developer.apple.com/news/?id=12212016b&1482372961 (Acc 171003)

[8] Local and Remote Notification Programming Guide: APNs Overview [Internet]. Apple Developer, Apple Inc, 2017, https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html (Acc 171003)

[9] Background Execution [Internet]. Apple Developer, Apple Inc, 2017, https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html (Acc 171003)

[10] Keychain Services Concepts [Internet]. Apple Developer, Apple Inc, 2016, https://developer.apple.com/library/content/documentation/Security/Conceptual/keychainServConcepts/02concepts/concepts.html (Acc 171003)

BILAGOR

API-specifikation

Autentisering

Kontrollerar ifall ett användarnamn och lösenord är korrekt.

- **URL**

  /api/member/get

- **Method**

  POST

- **Data**

  { “epost:”: “namn@epost.com”, “losen”: “12345” }

- **Respons**

```
{
    "epost": "namn@epost.com",
    "alias": "namn",
    "loginURL": "http://www.sportdate.se/mobile",
    "success": true
}
```

Om “success” är lika med “false” gick något fel och medlemmen är inte autentiserad. “loginURL” innehåller den URL som användaren skall skickas till en vid lyckad autentisering.

Notiser

Hämtar meddelanden för en specifik användare.

- **URL**

  /api/message/check

- **Method**

  GET

- **Data**

  { “epost:”: “namn@epost.com”, “losen”: “12345” }

- **Respons**

```
{
	"messages":
	[{
		"type": "notify",
		"messagetext": "Du har ett nytt meddelande från Lisa",
		"actionURL": "http://wwww.sportdate.se/mobile/mail.aspx"
	},
	{
		"type": "messagebox",
		"messagetext": "Se vårt nya erbjudande",
		"actionURL": "http://wwww.sportdate.se/mobile/nyhet.aspx"
	}]
	,"success": true
}
```

Om “success” är lika med “false” gick någonting fel.