Besöksapplikation med administratörskontroll

Lars Tallinger, Nino Selimovic

LUNDS UNIVERSITET
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för datavetenskap

Examensarbete

Nino Selimovic dat11nse@student.lu.se
Lars Tallinger dat11lta@student.lu.se

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds Universitet
Lund 2018

Sammanfattning

I Båstad på Bjärehalvön bedrivs campingverksamheten Båstad Camping. Med många gäster från hela Sverige så vill verksamheten på ett modernt sätt förmedla information om Bjärehalvöns besöksmål, sportaktiviteter, evenemang och restauranger. Campingen vill därtill ha möjligheten att förmedla ny eller uppdaterad information kontinuerligt till sina gäster.

Syftet med examensarbetet är att utveckla ett informationssystem bestående av en besöksapplikation, ett administratörsgränssnitt och en molntjänst som hanterar all data.

Systemet delades upp i arbetsmoment enligt tre komponenter: besöksapp, molnserver och kartfunktionalitet samt administratörsgränssnitt. Varje komponent utvecklades genom att först utföra intervjuer för att veta vilken information, utseende eller funktion som den skulle ha. Komponenten utvecklades sedan och som sista moment så utfördes test av komponenten.

Resultatet blev ett informationssystem bestående av en besöksapplikation, ett administratörsgränssnitt och en molntjänst som hanterar all data. Besöksappen innehåller information om Bjärehalvön och Båstad Camping i form av kartor, länkar och beskrivningar enligt tolv olika kategorier. Besöksapplikationen har även har möjlighet att ladda ner ny information från molntjänsten Google Drive och uppdatera innehållet enligt ändringar som utförs i administratörsgränssnittet. Administratörsgränssnittet som utvecklats är ett Java-baserat program som körs på en dator. När ändringarna som utförts i gränssnittet sparas så genereras en XML-fil. XML-filen placeras i Google Drive och laddas ner av applikationen.

Nyckelord: administratörsgränssnitt, Android Studio, besöksapplikation, Java, molnserver

Abstract

Located in Båstad, at the Bjäre peninsula, is the camping business Båstad Camping. With many guests from all over of Sweden, the owners of the business want to convey information about tourist attractions, sporting activites, events and restaurants at the Bjäre peninsula in a modern way.

The purpose of the thesis is to develop a visitors mobile application with information about the Bjäre peninsula and Båstad Camping, an administrator functionality where information can be updated and added in the application continually.

The process of creating the system was divided into three parts, each consisting of one component of the system: the visitors mobile application, functionality for maps and cloud server, and the administrator interface. Each component was developed by conducting interviews to know which information, design or function that the component should have. The component was then developed and lastly tested.

The result is an information system consisting of a visitors mobile application, an administration interface and a free cloud service functioning as a cloud server holding the data. The visitors mobile application contains information about Bjärehalvön and Båstad Camping in the form of maps, links and descriptions all sorted in twelve categories. The application also has the ability to download new information from the cloud service Google Drive and update its contents according to changes made in the administration interface. The administration interface that has been developed is written in Java that runs on a computer.

Keywords: administration interface, Android Studio, visitors mobile application, Java, cloud server

Innehåll

Terminologi

Administratör - De personer som har möjlighet att administrera information i informationssystemet, i detta examensarbete de två ägarna av Båstad Camping.

Administratörsgränssnitt - Ett gränssnitt med funktionalitet för administration av information i informationssystemet.

Android - Ett mobilt operativsystem för främst smarta telefoner och pekplattor.

Android Studio - En utvecklingsmiljö utvecklad av Google för att utveckla applikationer till operativsystemet Android.

Besöksapplikation - En applikation med samlad information om en specifik geografisk plats riktad mot besökare av platsen.

Båstad Camping - En campingverksamhet i Båstad som är beställaren av informationssystemet.

Element - Ett redigerbart gränssnittselement i applikationen. Elementen innefattar rubrik, underrubrik, text samt länk.

PNG - *Portable Network Graphics*, ett grafikformat.

Slutanvändare - De tänkta användarna av besöksapplikationen, främst Båstad Campings gäster.

SVG - *Scalable Vector Graphics*, ett grafikformat.

XML - *Extensible Markup Language*, ett märkspråk som i detta examensarbete används i Android Studio för att definiera element i applikationens gränssnitt.

Kapitel 1

Inledning

1.1 Bakgrund

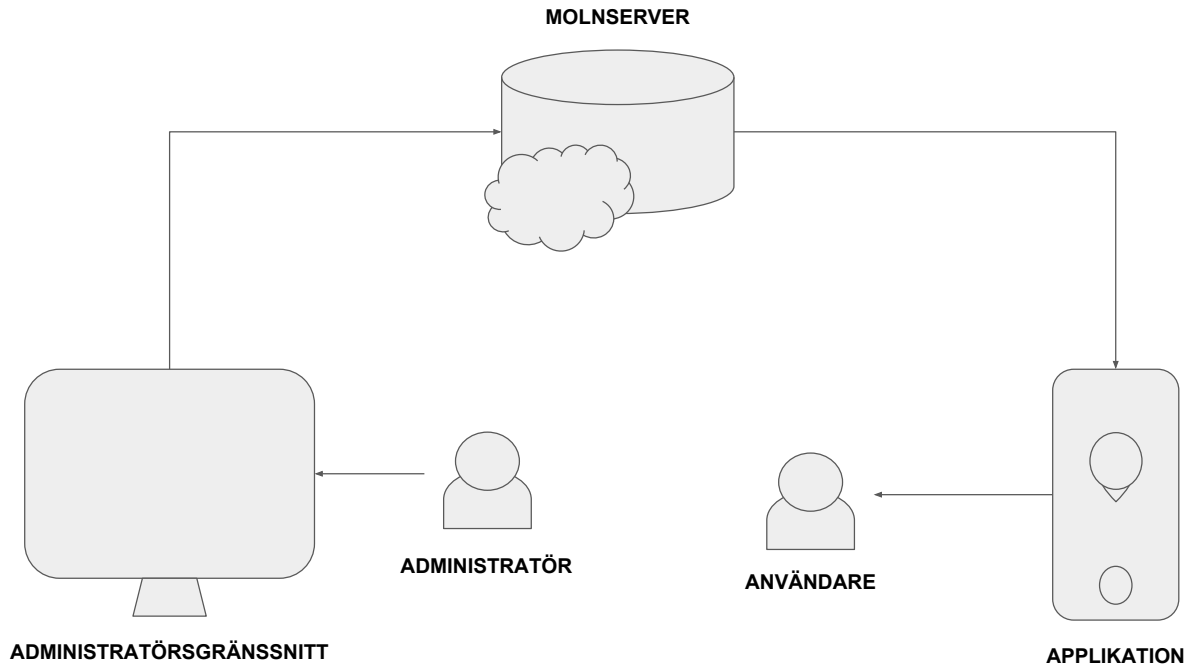
Detta examensarbete har utförts i samarbete med Båstad Camping och förmedlas av Miljöbron, som är en ideell organisation med syfte att skapa kontakter mellan högskola och näringsliv genom mindre och större projekt.

Båstad Camping är en privat näringslivsidkare som bedriver en camping-verksamhet en bit utanför centrala Båstad, Skåne län. Campingen har enligt ägarna ett bra rykte bland sina nationella och internationella besökare som kommer på besök mellan mars och september månad, där högsäsongen är under sommaren. Bjärehalvön, som till största del tillhör Båstads kommun, är en halvö med många sevärdheter och besöksmål som exempelvis klippvandring, vandrings- och cykelleder och större turistattraktioner som den årliga Tennisveckan. Campingens gäster tar idag del av information om besöksmål dels genom broschyrer men även genom information från campingpersonal och Internet. Informationen är på så vis utspridd och det finns ett behov av att samla all information på ett ställe så att gästerna på ett mer övergripligt sätt kan se vilka möjligheter till utflykter som finns. Båstad Camping vill hjälpa sina gäster att utforska och upptäcka Bjärehalvön och dess attraktioner. För att kunna uppnå detta mål så har Båstad Camping efterfrågat en besöksapplikation med samlad information om Bjärehalvön.

Målet med examensarbetet är att utveckla ett informationssystem som består av ett administratörsgränssnitt, en mobilapplikation samt en server. Det som är nyskapande med detta examensarbete är det kommer undersöka möjligheterna att använda gratis molntjänster som servrar åt mindre applikationer samt hur ett gränssnitt kan anslutas till denna server för att tillåta personer utan teknisk bakgrund att göra ändringar i en mobilapplikation.

1.2 Syfte

Syftet med examensarbetet är att ta fram ett informationssystem enligt figur 1.1. Detta informationssystem kommer att presentera information om besöksmål. Informationssystemet ska ge information om Bjärehalvöns olika besöksmål och sevärdheter som är lätt att förstå för användarna, samt vara lätt för administratörerna att administrera.

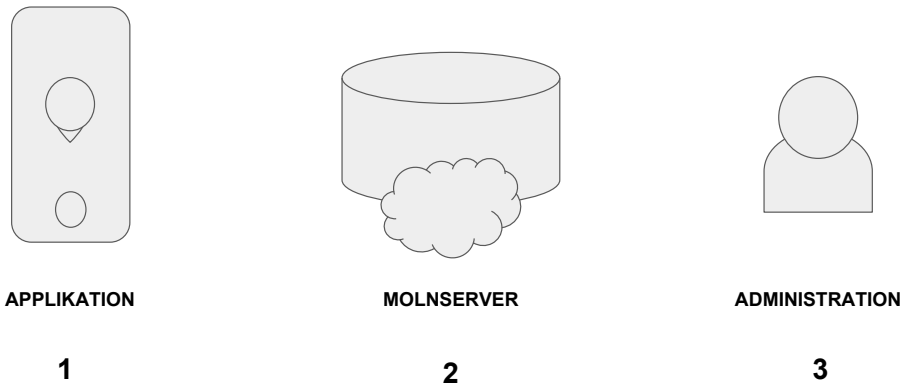


Figur 1.1: Informationssystem

1.3 Mål

Målet med examensarbetet är att enligt figur 1.2 :

1. Utveckla en applikation med kartfunktionalitet samt med information om Bjärehalvön och Båstad Camping.
2. Utveckla möjligheten till att använda en gratis molntjänst som molnserver.
3. Utveckla möjlighet till administration av molnservern genom ett administratörsgränssnitt.



Figur 1.2: Mål

1.4 Problemformulering

De problemställningar som kommer att besvaras av examensarbetet är:

1. Vilken typ av funktionalitet är viktigast för informationssystemet?
2. Finns det ytterligare funktionaliteter som är relevant specifikt för Båstads camping?
3. Hur kan ett administratörsgränssnitt utformas för att vara lättanvänt för personer utan teknisk utbildning?
4. Hur kan informationssystemet testas och vilka tester ska utföras för att kunna säkerställa att funktionerna är fungerande och relevanta för användarna?
5. Vilken typ av server lämpar sig bäst för denna typ av system?
    5.1. Kommunikation till mobilapplikationen: Vilka möjligheter och restriktioner finns det med att implementera och använda sig av gratis molntjänster som molnserver.
    5.2. Administratörsgränssnitt: Hur kan man använda en gratis molntjänst som molnserver och genom denna tillåta administratörer utan teknisk bakgrund att göra ändringar i en applikation?
6. Vilka APIer kan användas för att anpassa kartor vid utvecklandet av en mobilapplikation?
7. Hur kan ett gränssnitt i en mobilapplikation utformas för att underlätta navigation?

1.5 Avgränsningar

Mobilapplikationen kommer inte att utvecklas för Apples mobila operativsystem iOS utan endast för Googles mobila operativsystem Android.

1.5.1 Utveckling för iOS och Windows Phone

Utöver mobiloperativsystemet Android så finns det två andra: Apples iOS och Microsofts Windows Phone (WP).

Att utveckla för iOS medför två begränsningar för detta examensarbete: dels så krävs en Apple-dator då utvecklingsmiljön Xcode endast är kompatibel med macOS och det existerar inte ett officiellt stöd för Windows. Det krävs även ett avgiftsbelagt års-medlemskap för att ha möjligheten att publicera en applikation på App Store. På grund av detta så kommer mobilapplikationen endast att utvecklas för Android.

Avgränsningen att inte utveckla för Windows Phone görs då majoriteten av mobiltelefoner använder antingen Android eller iOS som operativsystem.

1.5.2 Google Maps API

Vid integrering av Google Maps i en mobilapplikation så används s.k. requests för att hämta och skicka data när kartfunktioner används. Det finns flertalet valmöjligheter när det kommer till antalet förfrågningar som kan göras mellan mobilapplikationen och Googles kartserver, det val som är gratis tillåter 2500 requests per tjugofyratimmarsperiod samt maximalt 50 requests per minut. I examensarbetet kommer detta gratis-alternativ att användas då användarbasen förväntas vara liten och således antalet requests.

Kapitel 2
Teknisk bakgrund

Detta kapitel går igenom de tekniker som använts för att utveckla applikation och administratörsgränssnitt.

2.1 Android Studio

För att utveckla applikationen i projektet så användes utvecklingsmiljön Android Studio. Android Studio är en integrerad utvecklingsmiljö vilket innebär att den innehåller en kompilator, text-editor samt avlusningsfunktionalitet. Android Studio erbjuder även drag-and-drop funktionalitet där applikationen kan skapas visuellt genom att välja design-element från en meny. Med samma funktionalitet kan beteende vid olika slags rörelser implementeras för knappar. Android Studio har även en inbyggd emulator som kan emulera applikationen direkt i utvecklingsmiljön utan att den behöver paketeras och exporteras till en mobiltelefon för att köras. [1][2]

2.1.1 XML

XML, *Extensible Markup Language*, är ett märkspråk designat för att lagra och skicka information. Informationen lagras mellan taggar som definieras av programmeraren. Informationen kan även lagras i hierarki genom att placera taggar inuti taggar.

Element definierar XML-filens innehåll, elementen är det innehåll som finns mellan start- och sluttaggar, till exempel <HUVUDRUBRIK >information </HUVUDRUBRIK>i detta examensarbete.

XML kan användas i Android Studio för att definiera layouts. En layout är en hierarki av vyer och vy-grupper i en applikation. En vy-grupp är en samling av vyer och andra vy-grupper där vyer innehåller visuella element och beteenden som användare kan interagera med. [11]

2.2 Eclipse

För att utveckla administratörsgränssnittet så har utvecklingsmiljön Eclipse använts. Eclipse är en integrerad utvecklingsmiljö med kompilator och texteditor.

För att designa gränssnittet så har plug-in-programmet *WindowBuilder* använts. För att tolka och hantera XML så har *DocumentBuilderFactory* använts som är ett standard-API i Java.

2.2.1 WindowBuilder

WindowBuilder är ett plug-in-program till Eclipse som installeras separat. Programmet möjliggör drag-and-drop funktionalitet för att skapa gränssnitt och beteenden till knapptryck. Programmet genererar sedan kod baserat på det som skapats. [4]

2.2.2 DocumentBuilderFactory

För att administratörsgränssnittet ska kunna läsa in och tolka elementen i en XML-fil så används ett Java standard-API, DocumentBuilderFactory.

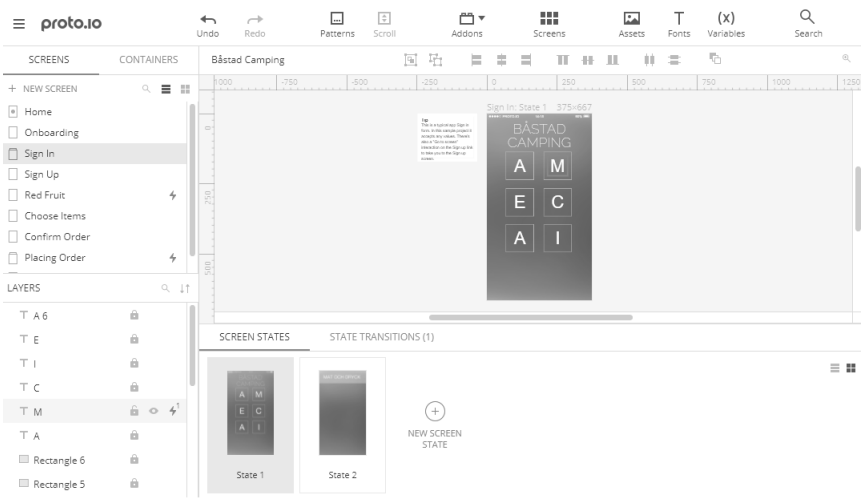
DocumentBuilderFactory används för att instansiera DocumentBuilder-objekt som sedan tolkar XML-filen efter valda inställningar. DocumentBuilder ger sedan sin output som ett DOM-objekt (Document Object Model). DOM-objekt behåller all information i trädform så att man enkelt kan hitta och manipulera information i trädet. [6]

Sättet som DOM lagrar kräver stora mängder minne, men i detta examensarbete ställs inget krav på administratörsgränssnittets snabbhet eftersom att applikationens XML-fil är liten.

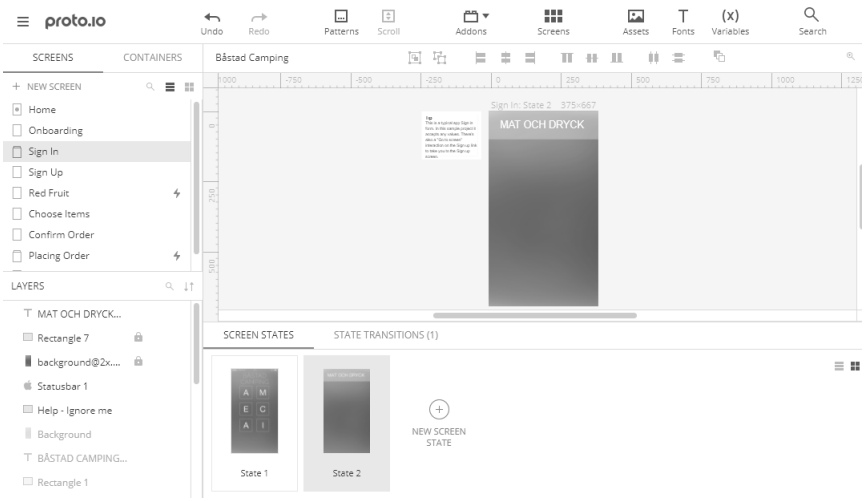
2.3 Proto.io

I andra fasen så användes Proto.io, ett webbaserat prototypverktyg, för att visualisera hur en prototyp-applikation som ska användas för elicitering av krav kan se ut. Med verktyget kan övergångar mellan menyer skapas. [9]

I figur 2.1 visas huvudmenyn som utvecklades i prototypverktyget. Huvudmenyn har olika knappar som vid användning tar användaren till olika undermenyer. Till exempel, när knapp M"väljs så sker en övergång till figur 2.2 som visar en undermeny, mat och dryck, till huvudmenyn.



Figur 2.1: Applikationens huvudmeny i prototypverktyget



Figur 2.2: Övergång till en undermeny efter val av knappen M"

Detta verktyg hjälpte examensarbetarna att utveckla en prototyp i Android Studio i form av en applikation. Applikationen användes sedan vid elicitering av krav.

2.4 Molntjänst

En molntjänst är en tjänst som främst tillhandahåller datalagring. I detta examensarbete så har möjligheten att använda en molntjänst som molnserver undersökts och implementerats.

Den molntjänst som har använts som molnserver är Google Drive. Google Drive används antingen genom webbläsaren eller genom ett nedladdningsbart program som skapar en mapp i datorn där allt innehåll i mappen sedan lagras i molnet.

Molntjänsten Google Drive har i detta examensarbete endast lagrat en XML-fil.

2.5 Inkscape

Inkscape är en gratis vektorgrafik-editor med öppen källkod. Programmet har i detta examensarbete använts för att skapa vektorgrafik för applikationen i form av ikoner, texter och knappar i formatet SVG. [5]

Kapitel 3
Metod

Detta kapitel beskriver vilka metoder och processer som använts för att utveckla prototypen. Kapitlet beskriver även vilka faser som projektet har varit uppdelat i och vad som gjorts i varje fas.

3.1 Tillvägagångssätt

Examensarbetet delades in i fyra faser:

1. Undersökning och utvärdering av olika utvecklingsmiljöer.
2. Informationsinsamling om vilken information som applikationen ska innehålla, utveckling av prototyp och sedan applikation.
3. Utforska möjligheterna att använda en gratis molntjänst som server samt samla information och integrera kartfunktionalitet.
4. Informationsinsamling och utveckling av:
    4.1. automatisk hämtning av ny information från molnserver till den utvecklade besöksapplikationen.
    4.2. administratörsgränssnitt.

Fas två till fyra började med elicitering av krav (intervjuer, informationsinsamling) för den specifika komponenten av systemet, sedan utvecklades och designades komponenten, slutligen så utfördes test av komponenten.

3.1.1 Fas ett

Arbetet påbörjades med att utforska vilka alternativ till utvecklingsmiljöer som fanns tillgängliga genom att ta reda på vilka som kan användas för utveckling av Android-applikationer. Diskussion fördes även mellan examensarbetarna vilken utvecklingsmiljö som bäst lämpar sig för applikationen.

3.1.2 Fas två

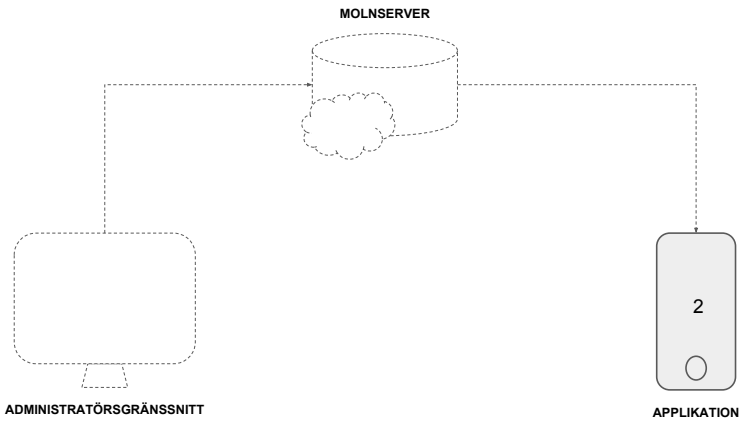
Andra fasen, figur 3.1, bestod av informationsinsamling om vilken information som skulle finnas tillgänglig för användarna av applikationen. Insamlingen av information utfördes med hjälp av öppna och strukturerade intervjuer (se avsnitt 3.2.2) med Båstad Campings personal och de tänkta slutanvändarna. Ytterligare information samlades genom besök på Båstad turistbyrå där broschyrer och faktablad om Bjärehalvön erhölls, diskussion fördes även kring de centrala turistattraktionerna på Bjäre-halvön med turistinformatörer.

För att få inspiration till applikationens design så undersöktes andra applikationer utgivna på Google Play med liknande syfte och fokus på turistinformation. Det som undersöktes var upplägget i gränssnittet, meny-hierarki, knappar och design i allmänhet och vilken funktionalitet som fanns tillgänglig. För att förstå vad som i allmänhet kännetecknar användarvänlig design så användes även information från böcker i ämnet, till exempel boken *Don't Make Me Think* av Steve Krug [15].

Undersökningen gav inspiration till upplägg av design och arbetet gick vidare med målet att utveckla en prototyp. Prototypen (se avsnitt 3.2.3), i form av en applikation, användes för att elicitera krav från ägarna av Båstad Camping genom att föra en öppen diskussion kring valen av design i prototypen samt om de olika kategorierna av information.

Med de definierade kraven som erhölls från eliciteringen med prototyp-applikationen så utvecklades sedan en slutgiltig applikation, när det kommer till design och meny-hierarki. Applikationen testades sedan mot Båstad Camping och besökarna på Båstad Camping.

Fasen resulterade i en applikation designad enligt eliciterade krav på innehåll av information. Fasen resulterade även i förfinad information om vad applikationen ska innehålla.



Figur 3.1: Systemkomponent som utvecklats under fas två

3.1.3 Fas tre

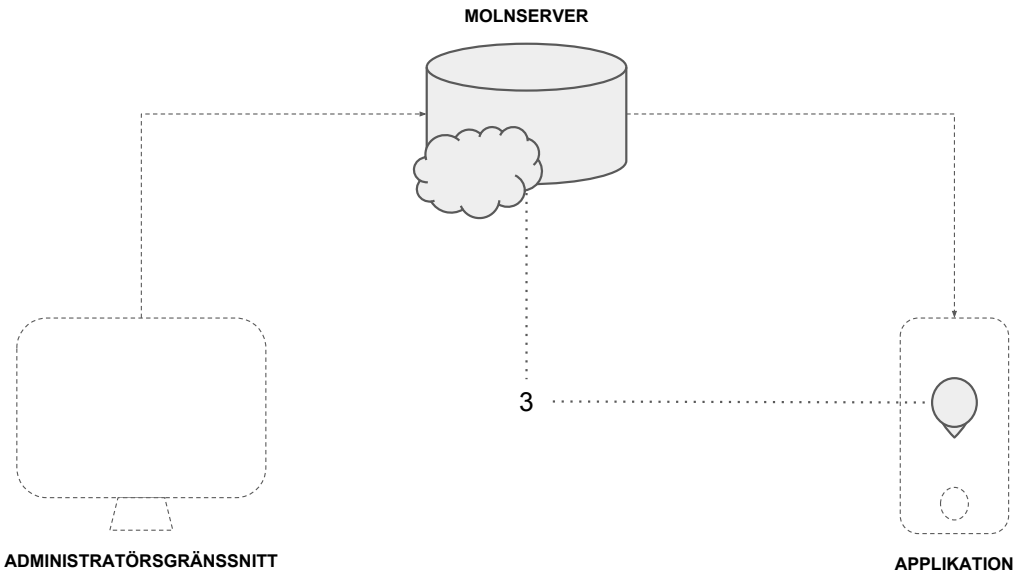
Resultaten från andra fasen tydliggjorde vilken funktionalitet som var önskvärd för besöksapplikationen samt vilka data som behövde finnas i servern.

I denna fas, figur 3.2, så samlades information om kartfunktionalitet i androidapplikationer. Det som undersöktes var vilka möjligheter som Google Maps har och vilka slags funktioner som är relevanta för applikationen utifrån den information som samlats in tidigare. Med den insamlade informationen så utfördes sedan en semi-strukturerad intervju (se avsnitt 3.2.2) med två personer, som är ägare till Båstad Camping och tänkta administratörer, för att elicitera krav på kartfunktionalitet.

Under denna fas så undersöktes även möjligheter och vilka restriktioner som finns med att använda en gratis molntjänst som molnserver. Detta gjordes genom att jämföra olika molntjänster (se Kapitel 4.3.2).

Med de eliciterade kraven (se appendix A) så implementerades kartfunktioner i applikationen, även information som samlades in från andra fasen lades in. I slutet av fas tre så utfördes ett test mot två personer från Båstad Campings personal för att försäkra att önskvärd kartfunktionalitet fungerade på korrekt sätt. Testet utfördes genom att kartfunktionaliteten visades upp och sedan hölls en öppen diskussion för att höra testpersonernas åsikter. Testning av kartfunktionalitet utfördes även av examensarbetarna genom att se till att kartlänkarna har önskat beteende i olika fall.

Fasen resulterade i att kartfunktionalitet integrerades i applikationen. Fasen resulterade även i ett beslut att använda molntjänsten Google Drive som molnserver och Google Maps för kartfunktionalitet.



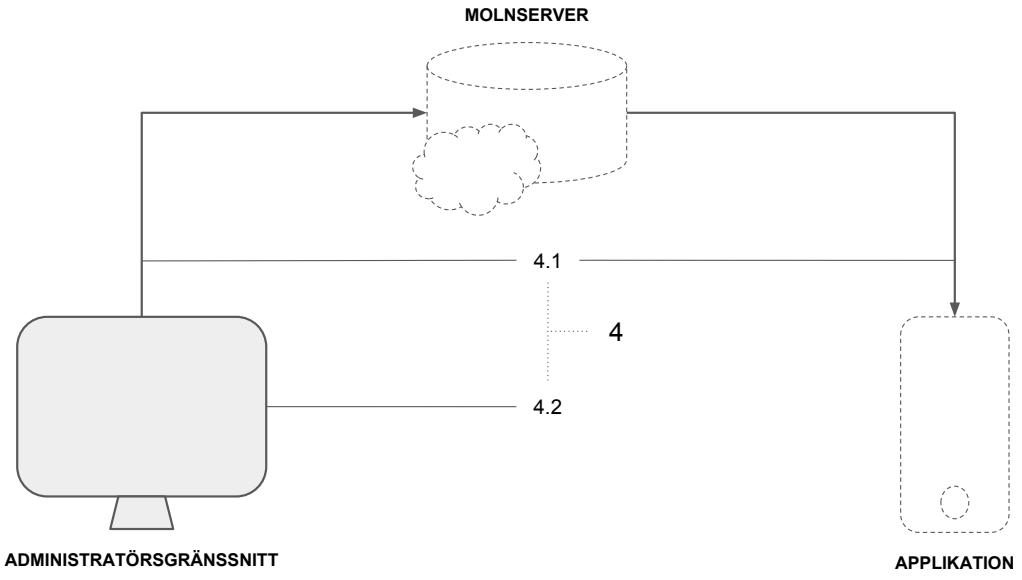
Figur 3.2: Systemkomponent som utvecklats under fas tre

3.1.4 Fas fyra

Den fjärde och sista fasen, figur 3.3, delades upp i två mindre delar. Den första delen bestod av att undersöka vilka möjligheter som fanns till att kunna ändra, ta bort samt lägga till information i en mobilapplikation utan att behöva ändra direkt i källkoden. I den andra delen av fjärde fasen gjordes intervjuer mot Båstad Campings personal som är de tänkte administratörerna, för att elicitera krav om vad som ska vara ändringsbart i applikationen, sedan undersöktes vilka möjligheter som fanns till att låta administratörerna ändra i applikationen på ett sätt som inte kräver kunskap om programmering eller avancerade datorkunskaper i allmänhet.

När tillräckligt med information samlats in om hur administratörsgränssnittet skulle fungera och se ut, så utvecklades ett Java-baserat program. Uppdaterings-funktionalitet utvecklades och integrerades i applikationen, 4.1 i figuren. För att säkerställa att administratörsgränssnittet var lätt att använda och att det producerar önskvärd funktionalitet för Båstad Campings personal så utfördes ett sista test mot Båstad Camping med hjälp av olika scenarion som testar de olika funktionerna och enkelheten att hitta rätt funktion.

Fasen resulterade i ett användargränssnitt i form av ett Java-baserat program som tillåter ändring i applikationen. Programmet används av administratörerna för att lägga till, ändra och ta bort information i applikationen. 4.2 i figuren.



Figur 3.3: Systemkomponent som utvecklats under fas fyra

3.2 Elicitering

I detta avsnitt behandlas de metoder som använts för att elicitera krav under projektet. Eliciteringen har resulterat i designkrav, datakrav och funktionella krav.

3.2.1 Intressentanalys

Innan elicitering av krav påbörjades så utfördes en första intervju i form av en mindre intressentanalys. Intervjun var helt öppen genom att inga förberedelser gjordes och inga stödfrågor användes.

En intressentanalys är en process som, bland annat, tar reda på vilka intressenterna är i ett projekt och vad de har för mål med produkten [16].

Analysen gav en övergripande förståelse för verksamheten Båstad Camping, och vad de hade för mål med systemet. Den insamlade informationen gav svar på vilka vardagliga arbetsuppgifter som Båstad Camping har, demografin hos besökarna, tillika slutanvändare, samt information om populära turistattraktioner.

3.2.2 Intervju

För att elicitera krav under de olika faserna så användes intervjuer som metod. Det finns olika typer av intervjuer.

Ostrukturerade intervjuer använder sig av öppna frågor, frågor som inte har något krav på ett specifikt svar, som ger en bättre bild av ett visst ämne. Ett exempel på en fråga som ställs kring ett specifikt ämne är “vad är attraktivt med Båstad?”, frågan är öppen i det avseendet att den är utforskande och letar inte efter ett exakt svar men är specifik i ämnet, attraktioner i Båstad. Denna typen av frågor genererar mycket information som sedan får förfinas för att få fram det som är relevant, den förfinade informationen ger fakta om exempelvis besökarnas ålder och vilka aktiviteter som de är intresserade av. Strukturerade intervjuer ställer konkreta frågor som har ett begränsat antal svarsalternativ, fördelen med denna typ av intervju är att tydliga svar erhålls om frågorna är koncisa. Frågorna är ofta förberedda i enkätform. Semi-strukturerade intervjuer kombinerar både konkreta och öppna frågor. [12] [18]

Under andra fasen så utfördes en strukturerad intervju med 16 stycken av de tänkta slutanvändarna enligt enkät-frågor i appendix C.1.1. En lista med öppna frågor förbereddes, utrymme för informell diskussion lämnades för att erhålla ett samtalsklimat som känns naturligt. Frågorna handlade om vilka fritidsaktiviteter de ägnar sig åt när de är här, deras allmänna kännedom om vad Båstad och Bjärehalvön har att erbjuda samt om de är stamgäster. Syftet med dessa intervjuer var att ge stöd till den information som togs fram från intressentanalysen med Båstad Camping.

Vid tredje fasen så intervjuades två ur personalen på Båstad Camping om kartfunktionaliteter. Här delgavs Båstad Camping först vilka möjligheter och begränsningar som finns för kartfunktionalitet i applikationen. Sedan utfördes en semi-strukturerad intervju om vilken funktionalitet som de hade kunnat tänka sig vara av nytta i en besöksapplikation enligt stödfrågorna i appendix C.1.2.

Vid den fjärde fasen gjordes en sista intervju med två ur Båstad Campings personal. Intervjun var semi-strukturerad och delgav först information om hur ett eventuellt administratörsgränssnitt kan fungera enligt tre olika alternativ. Sedan ställdes frågor enligt en förberedd lista med både konkreta och öppna frågor (se appendix C.1.3).

3.2.3 Prototyp

En annan metod som användes för att elicitera krav var användandet av en prototyp. En prototyp är en förenklad version av den slutgiltiga produkten och används för att elicitera krav i ett tidigt skede utan att behöva mycket tid till att utveckla en fullt fungerande applikation[16].

För att ta fram en prototyp att elicitera krav mot Båstad Camping så användes först Proto.io. Proto.io är ett verktyg som möjliggör utvecklandet av mobilapplikations-prototyper direkt i webbläsaren [9]. Med verktyget kan händelser vid knapptryck skapas, såsom visuella övergångar mellan menyer och design av gränssnitt. Den webbaserade prototypen gav en grund för hur den riktiga prototyp-applikationen, utvecklad i Android Studio, skulle se ut visuellt och hur flödesschemat skulle se ut i meny-hierarkin.

Denna typ av verktyg valdes för att det är ett bra sätt att diskutera applikationens design och funktionalitet examensarbetarna emellan, utan att ägna tid åt kodning. Med denna webbaserade prototyp är det möjligt att tidigt upptäcka otydligheter i till exempel design och flödesschema i meny-hierarkin.

Detta prototyp-verktyg användes tillsammans med brainstorming mellan examensarbetarna för att sedan utveckla en slutgiltig prototyp i Android Studio i form av en android-applikation. Prototyp-applikationen hade ett enkelt gränssnitt med menyer och knappar utan innehåll som användes mot Båstad Camping för elicitering.

3.2.4 Informationsinsamling

Information i applikationen

För att veta vilken information som skulle finnas i applikationen så besökte examensarbetarna Båstad Turistbyrå. Syftet med besöket var att främst ta reda på vilka aktiviteter som utmärker Båstad och Bjärehalvön och vad som attraherar dit turister under sommarsäsongen.

Annat material som användes var broschyrer som erhölls från besöket samt Båstad kommuns hemsida med turistinformation. [3]

Design

Vid informationsinsamling om design så användes främst litteratur, (till exempel *Don't Make Me Think* av Steve Krug) för att ta reda på vad som kännetecknar användarvänlig design. [15]

Informationen som samlades in handlade om att när människor använder hemsidor, eller i detta fall applikationer, så görs en snabb överblick och sedan väljs det som matchar bäst med det som eftersöks. En annan viktig sak att tänka på vid design är konventioner. Konventioner i design är att till exempel använda placering av design-element på ett sätt som de flesta användare är vana vid, t.ex. att kryss-rutan för att stänga ner något ska befinna sig längst upp till höger. Ett tredje viktigt designbeslut är att användaren ska ha lätt för att

orientera sig, till exempel att när användaren befinner sig i kategorin *Campinginformation* så ska det tydligt stå att användaren är där. [15]

Molnserver
För att avgöra hur en gratis molntjänst kan användas som molnserver så undersöktes molntjänsterna Google Drive, OneDrive och Dropbox.

Det som studerades var gratis minnesutrymme, API, språkstöd samt begränsningar på anrop. Möjligheten att skicka information mellan ett datorprogram till en applikation undersöktes också.

Google Maps API
Information om anpassning av Google Maps kartor hämtades från officiell dokumentation. Det som undersöktes var vilka möjligheter som fanns till att skräddarsy egna kartor. Till exempel, kunna ändra kartornas färgscheman, skapa egna platsnålar med ikoner.

3.3 Testning

I en applikation eller i ett program så kan många olika aktiviteter utföras av användaren, innan testning utförs så bör en lista upprättas på de viktigaste aktiviteterna som ska testas. Då antalet aktiviteter på en sådan lista kan bli många så bör dessa koncentreras till en rimlig mängd enligt olika kriterier.

Ett kriterium kan vara: att ha med aktiviteter som kan avslöja otydligheter, till exempel att vissa funktioner innehåller otydligheter som till exempel att hitta information om campingens egna restaurang. Informationen kan vid första anblick passa in i både kategorin *Campinginformation* men även kategorin *Mat & dryck*.

Ett annat kriterium kan vara: aktiviteter som av utvecklarna anses vara potentiella problem för användarna. Utvecklarna kan ha en aning om var det kan uppstå problem för användaren i form av exempelvis en knapp där designen inte heller för utvecklaren framstår som en knapp.

Lista kan också ordnas enligt kriterium som t.ex. aktiviteter som är svåra att återhämta sig ifrån om de utförs på fel sätt. [14]

Testning (se avsnitt 4.2) utfördes mot Båstad Campings besökare, Båstad Campings ägare, anhöriga till examensarbetarna. Testning utfördes även av examensarbetarna.

3.3.1 Scenario

För att testa design så gjordes först en lista med aktiviteter som skulle testas. Aktiviteterna sattes sedan i scenarion som skulle utföras av testpersonerna. Vid möte med testpersonerna så förklarades först hur testet skulle utföras och sedan lästes scenariot upp som testpersonen skulle utföra.

För att kvantifiera utfallet av varje scenario så observerades antalet rörelser som gjordes för att navigera till rätt funktion samt tiden det tog. För att få ytterligare underlag från testerna så ombads testpersonen att berätta varför varje beslut togs, detta gjordes för ytterligare tydliggöra tvetydigheter. Ett tidsintervall bestämdes där en utförd uppgift anses godtagbar om den befinner sig inom intervallet.

Scenario (se avsnitt 4.2.1) användes i test under fas två och fyra.

3.4 Arbetsmetodik

Detta avsnitt behandlar den metod som använts för att utvecklingen inom examensarbetet, varför den har valts, hur den använts och om det skett några avsteg från den. Avsnittet behandlar även hur kommunikation och versionshantering har skötts.

3.4.1 Kanban

Examensarbetarna valde att arbeta enligt utvecklingsmetoden Kanban. Kanban valdes på grund av att det inte finns många föreskrifter att förhålla sig till, det vill säga verktygen som definierar utvecklingsmetoden är få till antalet, en föreskrift kan exempelvis vara dagliga möten. Kanban valdes också för att den är modulär i det avseendet att det går att lägga till egna föreskrifter. Kanban använder visualisering av arbetsflödet och begränsning av hur mycket arbete som kan utföras samtidigt. En annan del av Kanban som passade projektet är möjligheten att inte behöva ha tidsbestämda sprintar som andra agila utvecklingsmetoder har. En sprint kan istället anses klar när arbetet i den är utfört. [13]

Det tänkta systemet bröts ner i komponenterna applikation, molnserver och kartor samt administratörsgränssnitt. Komponenterna fick utgöra faserna två till fyra i projektet.

Examensarbetarna valde att lägga till föreskriften att objekten, som i detta fall är komponenterna, i Kanban-fasen *to do* ska prioriteras. Systemet byggs upp från applikation till molnserver och kartor och till sist administratörsgränsnitt. Prioriteringen görs på detta vis för att när applikationens gränssnitt har bestämts så underlättar det att veta vilken information servern ska hantera samt vilken information som ska vara ändringsbar i administratörsgränssnittet och hela systemet blir på så vis klarlagt.

För att visualisera projektets gång så användes en Kanban-tavla, figur 3.4, där arbetsgången uppdaterades kontinuerligt. Antalet komponenter som kan utvecklas samtidigt begränsades även till en per Kanban-fas. Begränsningen gjordes för att examensarbetarna ville utföra arbetet sekventiellt och inte parallellt och endast flytta en komponent till en ny Kanban-fas när den tidigare var färdig.

Kanban valdes för att den föreskriver visualisering av arbetsprocessen. Kanban valdes även då fokus och arbetstid inte ville tas från projektets huvuduppgifter för att upprätthålla korrekthet när det kommer till föreskrifter.

To do 3	Utveckling 1		Testning/Avstämning 1	Done
	Elicitering/Informationsinsamling	Utveckling		
F3 - Molnserver & Kartor F4 - Administratörsgr.			F2 - Applikation	

Figur 3.4: Kanban-tavla

3.4.2 Utvärdering av Kanban

Visualiseringen av projektet underlättade arbetet, Kanban-brädet användes för det mesta i början av projektet. Ett avsteg från visualiserings-föreskriften gjordes efter en tid eftersom det redan stod klart hur hur processen skulle se ut för examensarbetarna och den blev således inte längre användbar.

Under projektets gång så gjordes vissa avsteg från utvecklingsmetoden. Ett avsteg som gjordes var föreskriften att mäta ledtiden, tiden det tar för en objekt att gå från *to do* till *done*, detta ska göras för att optimera ledtiden för nästkommande objekt. Då projektet endast bestod av tre objekt så ansågs detta inte vara nödvändigt och tillämpades således inte.

Utvecklingsmetoden Kanbans största hjälp till detta projekt var att visualisera arbetet och dela upp det i mindre delproblem och prioritera dessa. Kanban var mindre användbart mot projektets slut.

3.4.3 Kommunikation

Kommunikation mellan examensarbetarna bestod av arbetsmöten. När arbete utfördes hemifrån så användes Skype, både vid muntlig kommunikation men också vid chatt. Kommunikation mot Miljöbron sköttes främst med e-mail. Två gånger per månad skickades ett avstämningsmail med en kort uppdatering om examensarbetets status och vad som ska ske framöver. Vid vissa tillfällen delgavs uppdateringar även per telefonsamtal. Kommunikation mot Båstad Camping sköttes främst med möten och till viss del per e-mail. Mötena skedde i samband med besök då intervjuer och tester skulle utföras.

3.4.4 Versionshantering

För att versionshantera kod så användes varsitt lokalt repository hos examensarbetarna, kod synkroniserades sedan successivt.

För versionshantering av dokument så användes gemensam Google Drive som har inbyggd versionskontroll.

3.5 Källkritik

Detta avsnitt beskriver de källor som använts, om de är trovärdiga och vart de har tagits ifrån.

3.5.1 Litteratur

Litteraturen som har använts är följande:

- Soren Lauesen. *Software Requirements: Styles and Techniques*. Addison-Wesley, 2002.
- Janice C. Redish, Joseph S. Dumas. *A Practical Guide to Usability Testing*. Intellect Books, 1999.
- Yvonne Rogers, Jenny Preece, Helen Sharp. *Interaction Design - Beyond human computer interaction*. John-Wiley-and-Sons-Ltd, 2015.
- Mattias Skarin, Henrik Kniberg - *Kanban and Scrum, making the most of both*, 2010.
- James H. Frey, Andrea Fontana - *The Art of Science*, 1994.
- Bjarne Stroustrup. *The C++ Programming Language, Special Edition.*. Addison-Wesley, 2003.
- Steve Krug. *Don't Make Me Think, Revisited*. New Riders Publishing, 2014.

Litteraturen har använts i kurser på LTH eller blivit rekommenderade av handledare och examinator för examensarbetet, därför anses dessa källor tillförlitliga.

3.5.2 Hemsidor

- *Proto.io*
- Oracle Help Center - *Java garbage collection basics*
- Oracle Help Center - *Managing memory and garbage collection*
- stroustrup.com - *Stroustrup: FAQ*
- bastad.com - *Båstad & Bjärehalvön - The official Travel and Tour information*

Hemsidorna är tillförlitliga som källor då de är officiell dokumentation från företag, personer och kommuner.

Kapitel 4
Analys

Detta kapitel beskriver resultat från de metoder som använts samt vilka lösningar som dessa har motiverat. Kapitlet beskriver även vilka avvägningar som gjorts mellan olika tekniker och varför de valda teknikerna valts.

4.1 Elicitering

Detta avsnitt beskriver resultatet av de metoder som använts för att elicitera krav. Alla krav är samlade i appendix A.1.

4.1.1 Intressentanalys

Intressentanalysen gav följande information:

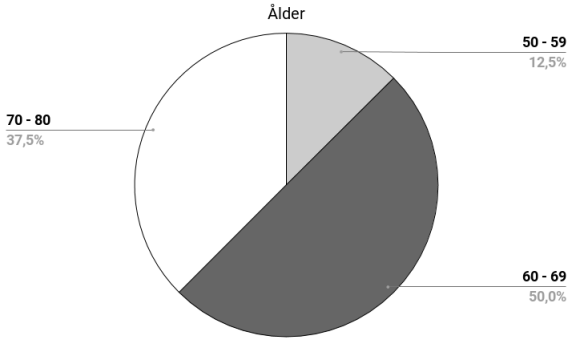
1. Båstad Campings mål med produkten är att ha samlad information om campingen och Bjärehalvön som kan administreras
2. De tänkta administratörerna är två till antalet och har inga avancerade tekniska kunskaper
3. Majoriteten av Båstad Campings besökare är pensionärer
4. Cykling och vandring är en av de vanligaste aktiviteterna hos besökarna
5. Besökarna kan få information om Bjärehalvön genom broschyrer i receptionen

4.1.2 Intervju

Båstad Camping är öppet mellan mars och september varje år. Högsäsongen inträffar mellan slutet av juni och mitten av augusti. Intervjuerna har utförts under försäsongen och eftersäsongen. Besökskategorierna är, enligt ägarna av Båstad Camping, främst pensionärer men ibland även barnfamiljer.

Fas två
Resultatet av intervjun (appendix C.1.1) som hölls i fas två besökarna på Båstad Camping tillika slutanvändare bekräftade uppgifterna från intressentanalysen att de flesta besökarna är i pensionärer samt att deras favoritaktiviteter är cykling och vandring.

Undersökningen visade även att evenemang på Bjärehalvön är något som de hade velat veta mer om men även information om vandringsleder.



Figur 4.1: Åldersfördelning hos besökarna

Figur 4.1 visar åldern hos de tillfrågade besökarna. 16 stycken besökare intervjuades.

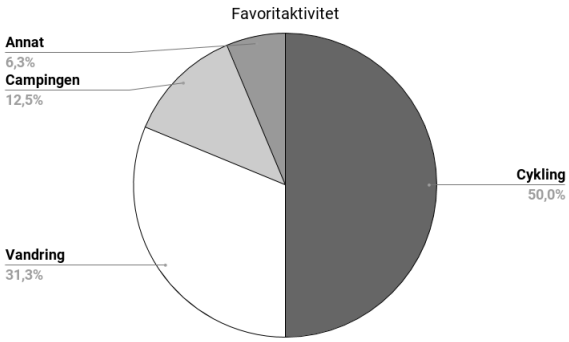
Fördelningen visar att de flesta är i pensionsåldern, vilket bekräftar den information som togs fram från intressentanalysen. Med denna åldersfördelning i åtanke så gjordes ett beslut att fokusera på starka kontraster, stora teckenstorlekar och ikoner i applikationen.

Figur 4.2: Andel besökare som är stamgäster

Figur 4.2 visar hur stor andel besökare som är stamgäster. De som inte är stamgäster är också förstagångsbesökare i Båstad.

Fördelningen visar att samlad information om till exempel förhållningsregler på campingen och parkeringsinstruktioner kan komma att vara nödvändigt för förstagångsbesökare, därför gjordes beslutet att skapa kategorin *Campinginfo* i applikationen.

Beslutet att skapa kategorin *Turistinfo* med allmän information riktad mot turister som aldrig varit i Båstad gjordes baserat på att många besökare aldrig varit i Båstad innan.



Figur 4.3: Favoritaktivitet hos besökarna

Figur 4.3 visar favoritaktiviteterna bland besökarna på Båstad Camping. Många besökare har med sig egna cyklar men det finns även cyklar till uthyrning. Med denna information så togs beslutet att skapa en kategori relaterat till *Cykling*. Även vandring är populärt vilket motiverar kategorin *Vandring*.

Figur 4.4: Vad besökarna vill veta mer om

Figur 4.4 visar vad besökarna vill veta mer om när de är på besök.

Evenemang är det som de flesta vill veta mer om och det innefattar evenemang på campingen till exempel grillkvällar, kubb- och bouletävlingar. Evenemangen innefattar även olika händelser på Bjärehalvön som exempelvis den årliga tennisveckan.

Denna information motiverade kategorin *Evenemang* och *Vandringsled*.

Figur 4.5: Hur besökarna får tag på information

Figur 4.5 visar hur stor andel besökare som använder en smarttelefon för att ta reda på information. Informationen ger insikt i potentiella användare av en besöksapplikation.

Resultat, fas två
Med resultatet av intressentanalysen och intervjuerna så konkluderades att applikationen ska innehålla information om evenemang, cykling, vandringsleder, turistinformation samt campinginformation.

Potentiella användare av applikationen hade även dragit nytta av en applikationsdesign som har stora design-element och tydlig kontrast mellan färgerna.

Fas tre
Under fas tre så gjordes en intervju med Båstad Camping för att ta fram vilken kartfunktionalitet som ska användas i applikationen.

Båstad Camping delgavs först information om vilka kartmöjligheter som finns (ruttinformation och platsinformation) sedan fördes en semi-strukturerad intervju med enkäten i appendix C.1.2 som stöd.

Det kom under intervjun fram att platsinformation kan passa bra för de flesta kategorierna. De platser som föreslogs ha platsinformation var tennisbanor för kategorin *Tennis*, golfklubbar för kategorin *Golf* samt ruttinformation för kategorin *Vandringsleder*. Intervjun gav även idén att ha med en platsinformation lokalt näringsliv som till exempel gårdsbutiker.

Resultat, fas tre Resultatet av intervjun i fas tre var att ha med två olika typer av kartfunktionalitet: platsinformation som är en plats utpekad på kartan med en platsnål samt ruttinformation som är en vägbeskrivning från Båstad Camping till en destination.

Fas fyra
Under fas fyra så utfördes ytterligare en intervju mot Båstad Camping för att ta fram vilken administratörsfunktionalitet som administratörsgränssnittet ska ha. Enkäten i appendix C.1.3 användes som stöd i en semi-strukturerad intervju.

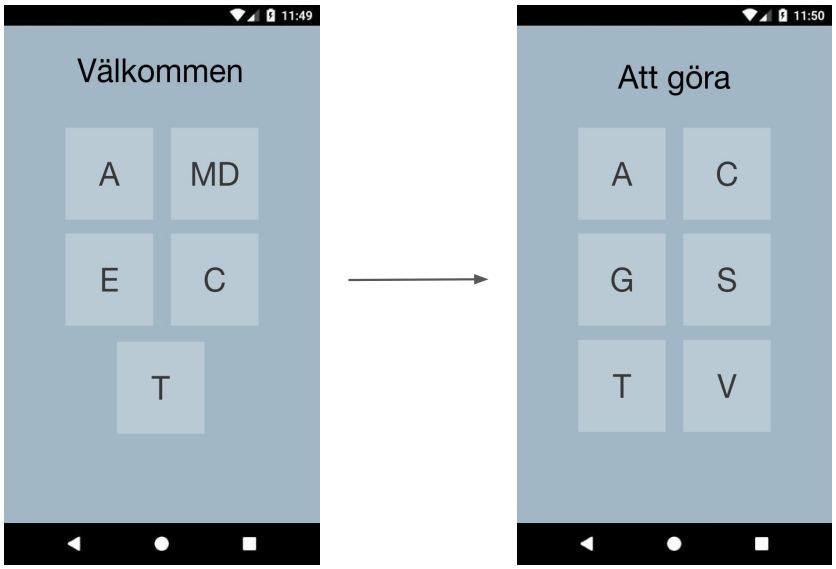
Båstad Camping vill ha möjlighet att främst kunna lägga till information om den egna verksamheten. De kategorier där det främst finns information om campingen är *Campinginfo* samt *Evenemang*. Den funktionalitet som togs beslut om att implementera var funktionaliteterna att lägga till, ändra och ta bort information i applikationen.

Resultat, fas fyra
Resultatet av fas fyra gav information om vad administratörerna ska kunna administrera i applikationen och på vilket sätt.

4.1.3 Prototyp

Resultatet av prototyp-applikationen blev figur 4.6. Eliciteringen med prototypen utfördes under ett möte med Båstad Camping i fas två. Mötet var helt öppet i det avseendet att inga stödpunkter användes, det vill säga ordet var fritt. Båstad Camping fick tala fritt om vad de tyckte om upplägget på menyn och designen och om de tänkta kategorierna.

Resultatet av eliciteringen var beslutet att ha ytterligare en kategori *Lokalt* med lokalt näringsliv på Bjärehalvön. Lokalt näringsliv är exempelvis gårdsbutiker.



Figur 4.6: Prototyp-applikation utvecklad i Android Studio

4.2 Test

4.2.1 Test, fas två

Under fas två så testades applikationen. Det som testades var kategorierna, menyhierarkin samt design. Testet utfördes först med scenarion mot Båstad Camping (appendix B.2). Testpersonerna var två stycken administratörer på Båstad Camping och två anhöriga till examensarbetarna. De ansågs vara lämpliga testpersoner eftersom de inte har bra tekniska kunskaper.

Testtillfället påbörjades med en kort genomgång om vad som ska testas och hur det kommer gå till och testpersonerna uppmuntrades att tänka högt vid funderingar. Under testet så antecknades tiden det tog att utföra scenariot med hjälp av ett tidtagarur, även antalet rörelser antecknades och jämfördes sedan med det minimala antalet rörelser.

Scenario	Tid	Rörelser	Anteckningar
1	>7	>optimalt	Första användningen av applikationen, går direkt in på evenemang istället.
2	≤7	optimalt	Hittar direkt.
3	≤7	optimalt	Hittar direkt.
4	>7	>optimalt	Går in på campinginfo istället för mat & dryck.
5	≤7	optimalt	Hittar direkt och klickar på tom kartlänk.
1			

Scenario	Tid	Rörelser	Anteckningar
1	≤7	optimalt	Hittar direkt.
2	≤7	optimalt	Hittar direkt.
3	≤7	optimalt	Hittar direkt.
4	>7	optimalt	Hittar direkt men tog längre tid än förväntat.
5	≤7	optimalt	Hittar direkt och klickar på tom kartlänk.
2			

Scenario	Tid	Rörelser	Anteckningar
1	>7	optimalt	Hittar direkt men tog längre tid än förväntat.
2	≤7	optimalt	Hittar direkt.
3	≤7	optimalt	Hittar direkt.
4	≤7	optimalt	Hittar direkt.
5	>7	>optimalt	Hittar direkt och klickar på tom kartlänk.
3			

Scenario	Tid	Rörelser	Anteckningar
1	>7	optimalt	Hittar direkt men tog längre tid än förväntat.
2	≤7	optimalt	Hittar direkt.
3	≤7	optimalt	Hittar direkt.
4	≤7	optimalt	Hittar direkt.
5	≤7	optimalt	Hittar direkt.
4			

Figur 4.7: Testets utfall

Testets utfall visas i figur 4.7. Scenario fyra misslyckades av två testpersoner men inget beslut togs att flytta campingens mat-meny till kategorin *Campinginfo* från *Mat & dryck*

då testpersonerna, som var administratörerna, ansåg det viktigare att campingens meny syns högst upp över alla restauranger i *Mat & dryck*.

En annan observation som gjordes är att scenario ett misslyckas för tre testpersoner. En anledning till detta är att applikationen precis blivit introducerad till testpersonerna och de därför gör en överblick av applikationen innan de tar ett beslut. Denna initiala överblick tar tid eftersom applikationen precis blivit introducerad, varför två av testpersonerna inte lyckades utföra scenario ett på utsatt tid men på optimalt antal rörelser.

Test mot slutanvändare

Ett mindre test mot Båstad Campings besökare och tänkta slutanvändare gjordes också för att säkerställa att designen i applikationen var tydlig. Fem besökare fick frågor om designen enligt enkäten i appendix B.1, de fick samtidigt applikationen uppvisad. Endast designen i huvudmenyn testades då det var samma storlek på design-elementen i kategorin *Aktivitet*.

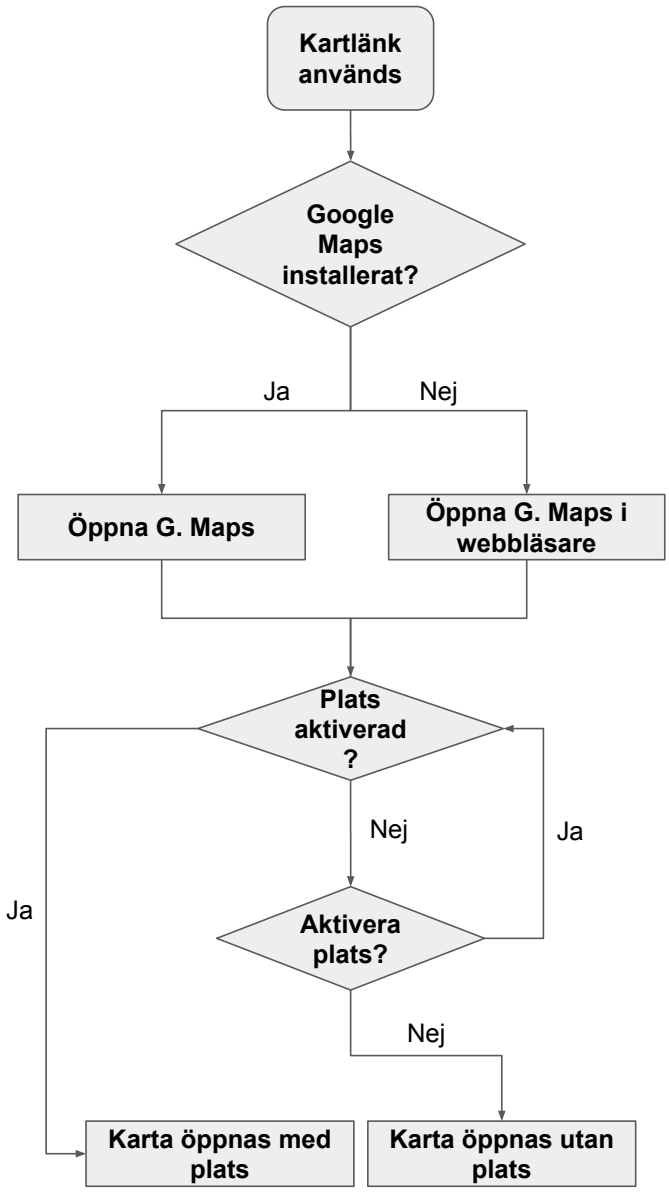
Resultatet av testet motiverade ändring av knapparnas storlek i applikationen. Detta gjordes eftersom att testpersonerna upplevde att knapparna var för små och därför svårt att se knapparnas texter och ikoner.

4.2.2 Test, fas tre

Under fas tre implementerades kartfunktionalitet i applikationen. För att testa kartfunktionalitet så utfördes först ett mindre test mot de två administratörerna på Båstad Camping för att säkerställa att önskad kartfunktionalitet fanns. Testet utfördes genom att kartfunktionerna demonstrerades i applikationen och sedan fick testpersonerna lämna åsikter om dem.

Ytterligare tester utfördes av examensarbetarna i form av funktionstest. Testen gick igenom alla kartlänkar och såg till att de uppvisade korrekt beteende enligt flödesschemat i figur 4.8.

Resultatet av testet visade att kartfunktionerna fungerade på önskat sätt både funktionellt och på det sätt som Båstad Camping önskade.



Figur 4.8: Flödesschema för test av kartfunktionalitet

4.2.3 Test, fas fyra

Ett sista test för att säkerställa att administratörsgränssnittet fungerar på korrekt vis utfördes i fas fyra. Testpersonerna var två tänkta administratörer tillika ägare av verksamheten Båstad Camping.

Innan testet utfördes så hölls en kort presentation över hur systemet fungerade så att testpersonerna har en förståelse för vilken roll administratörsgränssnittet har. Sedan visades administratörsgränssnittet upp och vilka funktioner som fanns. Uppvisningen gjordes inte grundligt för att behålla autenticitet när testet sedan skulle utföras. Administratörerna fick sedan sitta vid en dator och utföra scenarion som lästes upp enligt appendix B.3.

Resultatet av testerna gav insikt om att sidomenyerna i administratörsgränssnittet behöver ha tydliga visuella avgränsningar för att särskilja sidomenyn för rubriker och sidomenyn för element som tillhör en rubrik.

Ett annat problem som belystes var att i vissa kategorier så var rubrikerna inte ordnade efter bokstavsordning, något som bidrog med att det tog lång tid att hitta en viss rubrik.

Testet visade att administratörsgränssnittets design och funktionalitet till största del var utvecklat på ett sätt som underlättar personer utan avancerade tekniska kunskaper att redigera information i en applikation.

4.3 Val av tekniker

4.3.1 Programmeringsspråk

För att undersöka vilket programmeringsspråk som bäst lämpar sig för projektet så undersöktes två alternativ, C++ och Java. Dessa två programmeringsspråk jämfördes eftersom examensarbetarna har tagit kurser i dem.

Systemet består av applikation, molnserver och administratörsgränssnitt. För att utveckla en android-applikation så kan båda språken användas.

Fördelen med C++ är att språket har funktionalitet som tillåter programmeraren att göra anpassningar i koden så att applikationer har mindre och mer effektiv minnesanvändning, eftersom språket tillåter manuell minneshantering. Språket har därför ingen definierad skräpsamlingsfunktion i sin grundkod, en sådan kan däremot importeras från externa källor [17]. Språket är även designat med skalbarhet i åtanke, en fördel om det utvecklade programmet förväntas bli stort. [10]

Nackdelen med C++ är att det kan vara komplicerat att sköta minneshantering manuellt beroende på hur stor eller liten applikationen är så måste varje objekt som hanterar minne allokeras och avallokeras korrekt.

Fördelen med att använda Java är att utvecklingsmiljön Android Studio är specialanpassad till Java och erbjuder således en komplett utvecklingsmiljö framtagen specifikt för apputveckling för Android. En annan fördel med Java är att avallokering av objekt som upptar minne sköts automatiskt, detta sker med hjälp av en skräpsamlare som definierats i språket. Skräpsamlaren går igenom alla objekt, identifierar de som ej används och raderar dessa.[7] Nackdelen med Java är att stora applikationer kan bli långsamma eftersom att många objekt tar längre tid att gå igenom av skräpsamlaren när den ska identifiera vilka av dessa som ej används. [8]

För att utveckla ett gränssnitt så hade både C++ och Java passat in. Det blev snabbt klarlagt att Java även här hade passat bättre. Anledningen till detta är för att Java används för applikationen och viss kod som tolkar XML kunde återanvändas i desktop-applikationen.

Java valdes som programmeringsspråk för applikationsdelen av systemet för att den inte förväntades bli stor när det kommer till minne, hänsyn till minneshantering behövdes därför inte tas. Java valdes även eftersom det fanns en anpassad utvecklingsmiljö för applikationsutveckling samt att en del kod förväntades vara överförbar till administrationsgränssnittet om den skulle skrivas i samma språk.

4.3.2 Molnserver

För att bestämma vilken molntjänst som skulle användas så jämfördes tre stycken molntjänster. Molntjänsterna OneDrive, Google Drive och Dropbox valdes för jämförelse eftersom att examensarbetarna är välbekanta med dem. Det som undersöktes var om de olika tjänsterna erbjöd ett API och om detta API erbjuder verktyg för kommunikation mellan molntjänst och en android-applikation. Undersökningen genomfördes genom att läsa API-dokumentation från de olika tjänsternas hemsidor. Sedan undersöktes även hur stor lagringskapacitet som de olika tjänsterna erbjöd utan extra kostnad samt om denna är tillräckligt stor för att hålla tänkt data. Även begränsningar på antalet anrop till servern undersöktes för att säkerställa att applikationens funktionalitet inte begränsas vid frekvent användning.

Jämförelse	Google Drive	OneDrive	Dropbox
Gratis minnesutrymme	15GB	5GB	2GB* * upp till 32GB genom anslutning av sociala medier och värvning
API	Ja	Ja	Ja
Stöd	Java, Python, .NET	ASP.NET, JavaScript, Java, Objective-C, C#	HTTP, .NET, Java, JavaScript, Python, Swift, Objective-C
Begränsningar på anrop	1 000 000 000 per dag	Ej officiellt angivet Ska enligt dokumentation vara tillräckligt för normalfallen	Ej officiellt angivet Ska enligt dokumentation vara tillräckligt för normalfallen

Figur 4.9: Jämförelse av molntjänster

Resultatet av jämförelsen var beslutet att använda Google Drives molntjänst som molnserver. De faktorer som motiverade beslutet var det faktum att Google Drive har störst gratis lagringskapacitet (utan att behöva värva användare för mer kapacitet). Även om det som ska lagras på molntjänsten inte förväntas uppta mer än en megabyte så säkerställer 15 gigabyte en god marginal. En annan faktor är att begränsningen på anrop är angiven, om applikationen får många användare så kan det vara bra att veta att det finns en tydlig övre gräns innan en betald uppgradering måste övervägas. Den sista faktorn är att utvecklingsmiljön, kartorna och operativsystemet alla kommer från Google och att det därför är naturligt att också använda Googles molntjänst.

Nästa steg var att undersöka hur molntjänsten kan användas som molnserver. Vid efterforskningar hur en Google Drive API kan användas för att ladda ner en fil från Google Drive till en applikation så gjordes upptäckten att det går att göra utan Google Drive API. Detta ledde till att ett beslut togs att inte använda Google Drive API utan att endast ha molntjänsten Google Drive.

Något som ytterligare bekräftade att Google Drive API inte behövdes i projektet var upptäckten att administratörsprogrammet kan köras direkt i en Google Drive-mapp (om Google Drive är installerat på datorn), när programmet sedan genererar en XML-fil så ersätts den äldre XML-filen i Google Drive automatiskt med den ny-genererade. Om Google Drive inte är installerat kan XML-filen föras över manuellt med drag-and-drop till Google Drive i webbläsaren.

4.3.3 Utvecklingsmiljö

Android Studio var en av utvecklingsmiljöerna som bäst skulle lämpa sig för utveckling av applikationen. Verktygen som finns tillgängliga i Android Studio var ett av de främsta fördelarna då dessa skulle underlätta utvecklandet av applikationen. Exempel på verktyg är visuell representation av den kod man skriver i realtid, drag-and-drop av visuella element, automatisk kod-optimering för att möjliggöra användning på äldre enheter samt en emulator för att kunna testa applikationen utan att behöva föra över den till en Android-telefon. Nackdelen med att använda Android Studio är att det är en utvecklingsmiljö som är krävande på minnesresurser eftersom att den har många verktyg och bibliotek förinstallerade. Andra utvecklingsmiljöer, som exempelvis Eclipse, hade krävt att olika bibliotek läggs till manuellt för att underlätta programmeringen.

För att utveckla administratörsgränssnittet så användes Eclipse. Denna utvecklingsmiljön valdes för att den är välkänd hos examensarbetarna.

4.4 Administratörsgränssnitt

När administratörsgränssnittet skulle utvecklas så skulle hänsyn tas till hur man kan tillåta ändring av information i applikationen på ett sätt som varken kräver avancerad teknisk kunskap i form av t.ex. programmeringserfarenhet eller tillåter obehöriga att utföra ändringar. Följande alternativ togs upp:

1. Manuell ändring i XML-fil
2. Administratörsgränssnitt i applikationen
3. Administratörsgränssnitt i Java-program

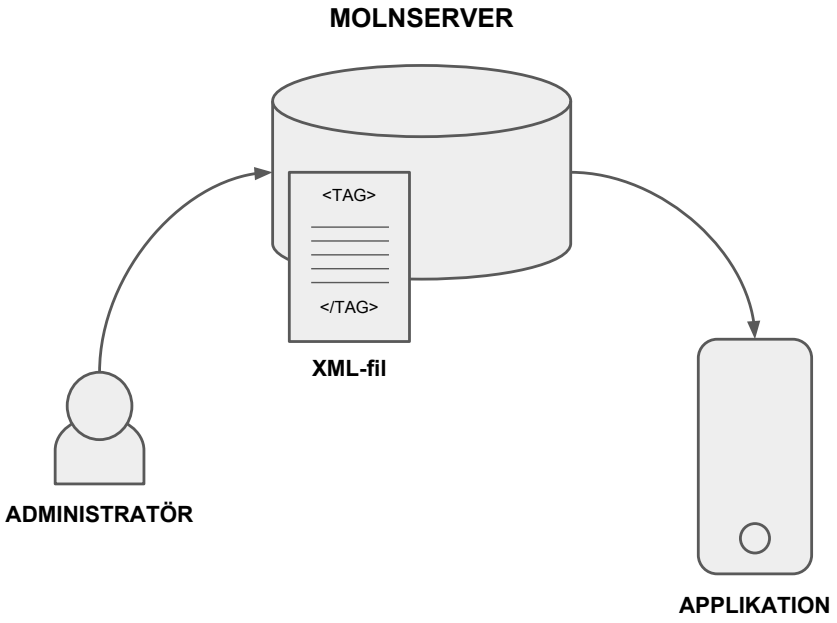
Fördelarna med det första alternativet är att systemet inte skulle bli komplext. Det krävs endast ändring i en fil på molnservern. Nackdelarna är att alternativet kräver programmeringskunskaper samt att kostnaden är hög ifall det sker ett misstag.

Fördelarna med det andra alternativet är att systemet inte skulle bli komplext för administratörerna, ändringar utförs direkt i applikationen. Nackdelarna är att systemet är komplext att utveckla ur ett säkerhetsperspektiv. Fördelarna med det tredje alternativet är att systemet är användarvänligt. Nackdelarna är systemet blir större samt att det krävs en dator att utföra ändringar. Det tredje alternativet valdes eftersom att det bäst tillgodoser kriteriet på användarvänlighet. Anledningen till varför detta alternativ är användarvänligt är för att gränssnittet används på en dator och har knappar och vyer som många kan känna igen sig i.

4.4.1 Manuell ändring i XML-fil

Den information som finns i applikationen finns i en XML-fil mellan olika öppnings- och stängningstaggar, exempelvis <HUVUDRUBRIK >information </HUVUDRUBRIK>.

Det första alternativet var att låta administratörerna ändra “information” som finns mellan olika taggarna direkt i XML-filen. Detta skulle kunna utföras genom att administratörerna loggar in på Google Drive på en dator och ändrar i filen, sedan laddar applikationen in den uppdaterade XML-filen.



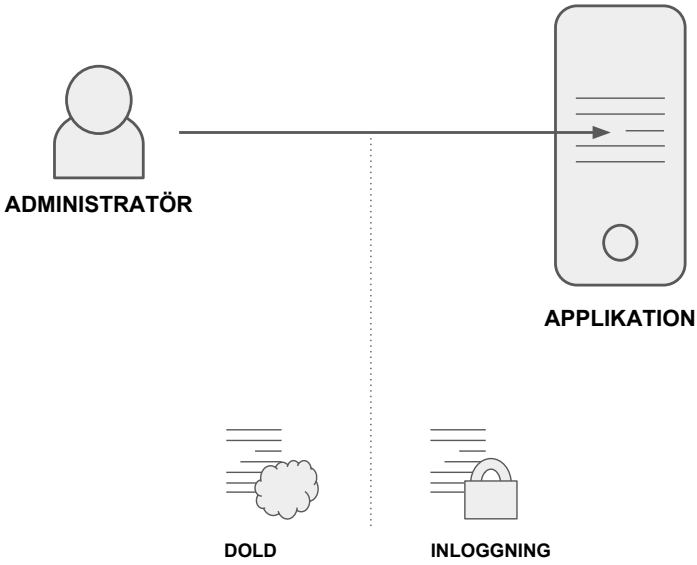
Figur 4.10: System där ändring utförs direkt i XML-fil

Lösningen är enkel ur en implementerings-synvinkel och kräver endast att applikationen har funktionalitet för att ladda in filen. Lösningen är däremot inte användarvänlig eftersom att det finns en hög risk att administratörerna, som inte har några programmeringskunskaper, omedvetet skulle kunna råka ändra XML-filens syntax. Exempelvis att en tag inte stängs på ett korrekt vis. Detta hade lett till att applikationen kraschar när den laddar in filen, något som hade varit kostsamt att återhämta sig ifrån då det hade krävts extern hjälp för att återställa problemet.

4.4.2 Administratörsgränssnitt i applikationen

Den andra möjligheten som utforskades var att utveckla ett administratörsgränssnitt direkt i applikationen. Detta hade kunnat utföras på två sätt:

1. En dold redigeringsfunktion utan verifiering av den som försöker få åtkomst.
2. En öppen redigeringsfunktion med inloggning för verifiering av den som försöker få åtkomst.



Figur 4.11: System där ändring utförs direkt i applikationen med antingen dold funktionalitet eller genom inloggning

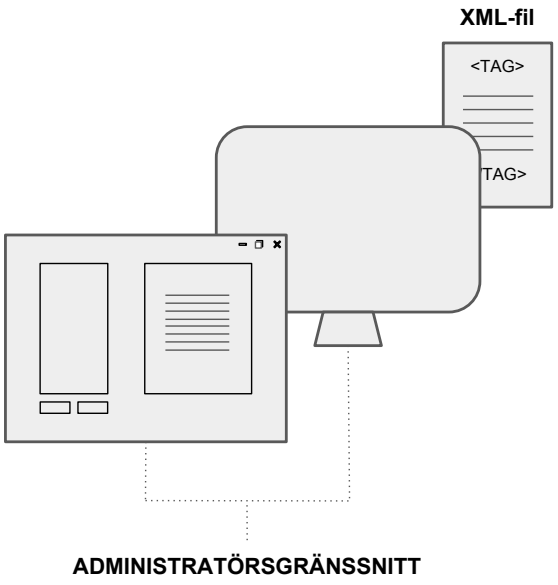
Det första alternativet hade krävt att funktionaliteten döljs så att inte användare kan komma åt den. Administratörerna hade kunnat nå redigeringssidan genom att exempelvis hålla inne en knapp i huvudmenyn i tio sekunder. Det andra alternativet hade varit att implementera en inloggningsfunktion för administratörerna. Administratörerna loggar in och får tillgång till en redigeringssida.

Fördelen med dessa två alternativ är att de gör systemet mindre komplext, administratörerna behöver inte ha tillgång till dator för att utföra redigeringar och kan göra ändringar på stående fot så länge de har nätverksuppkoppling.

Nackdelarna med första alternativet är att den dolda redigeringssidan alltid är utsatt för risken att bli upptäckt, en risk som inte helt kan åtgärdas då applikationen har publik tillgänglighet. Detta hade kunnat lösas med det andra alternativet men det hade i sin tur krävt ytterligare en server som hanterar inloggningsuppgifter samt säkerhet av dessa, något som hade gjort systemet komplext.

4.4.3 Administratörsgränssnitt i Java-program

Det tredje alternativet som undersöktes var att utveckla ett program som körs på en dator. Programmet är ett gränssnitt för innehållet som finns i XML-filen.



Figur 4.12: System där ändring utförs genom ett gränssnitt på en dator

Detta alternativ var intressant då det är lätt att göra användarvänligt, gränssnittet kräver inte många knappar och vyer. Programmets gränssnitt kan utvecklas så att det är lättförståeligt genom att använda sig av design-konventioner som de flesta som använt datorprogram kan känna igen sig i.

De nackdelar som finns är att administratörerna behöver använda en dator för att utföra ändringar samt att de manuellt behöver lägga upp XML-filen som genereras på molnservern om Google Drive inte är installerat.

4.5 Liknande system

För att få inspiration till applikationen så undersöktes andra besöksapplikationer på Google Play. Det som undersöktes var hur huvudmenyn såg ut, vilka kategorier av information som fanns samt om de har kartfunktionalitet. Tre applikationer undersöktes: *Båstad & Bjärehalvön*, *Copenhagen*, *Stockholm*.

Stockholm
Stockholm-applikationen hade en startsida som direkt vid uppstart visar en karta över centrala Stockholm. I kartan finns platsnålar som pekar ut olika kategorier av information, som teatrar, hotell och butiker. Från startsidan kan man även öppna ett filter som filtrerar kategorierna med checkboxar.

Huvudmenyn nås genom att klicka på en ikon längst upp i vänstra hörnet. Huvudmenyn glider in från sidan och visar tillgång till funktionaliteter som forum, spel, favoriter och tips.

Båstad & Bjärehalvön
Applikationen har sin huvudmeny som startsida. Huvudmenyn består av olika knappar för olika kategorier, för att delge information så används hemsidelänkar.

Applikationen har även viss kartfunktionalitet i form av en karta över Bjärehalvön med platsnålar för olika shopping-platser.

Copenhagen
Applikationen har allt samlat på sin startsida. Det finns sex olika rubriker för varje kategori, under varje rubrik går det att svepa för att se alla objekt som finns för den rubriken.

Applikationens kartfunktionalitet fungerar i offline-läge då kartorna laddas ner tillsammans med applikationen.

Resultat av undersökningen
Undersökningen av applikationerna resulterade i inspiration till att ha huvudmeny direkt på startsidan. Detta ger en bra översikt direkt vid uppstart av applikationen och ett enklare sätt att orientera sig.

När det kommer till kartfunktionalitet så beslöts det att det inte skulle finnas en karta med flera platsnålar samtidigt eftersom när kartan laddar in alla knappnålar så är det svårt att särskilja vad som är vad på en mobiltelefonskärm. Vill man klicka på knappnålarna blir det även då svårt att pricka rätt. Därför valdes separata kartlänkar specifika för en typ av destination i den utvecklade applikationen. Inspiration till kategorier togs av applikationen *Båstad & Bjärehalvön*. Kategorin *Gårdsbutiker* i den undersökta applikationen inspirerade, tillsammans med eliciteringen med prototypen i fas två, skapandet av kategorin *Lokalt* i Båstad Campings besöksapplikation.

Kapitel 5
Resultat

Detta kapitel beskriver resultatet av examensarbetet. Kapitlet går in på vad som har utvecklats när det kommer till administratörsgränssnitt, applikation, kartfunktionalitet och molnserver.

5.1 Besöksapplikation

Slutresultatet för besöksapplikationen blev en applikation med en huvudmeny, figur 5.1, som har sex stycken kategorier av information. Dessa är:

1. Aktivitet, figur 5.3
2. Mat & Dryck, figur 5.2
3. Evenemang
4. Campinginfo
5. Lokalt
6. Turistinfo

Kategorierna innehåller information i form av platsinformation, vägbeskrivningar, länkar till hemsidor med ytterligare information samt kortare beskrivningar som till exempel restaurangers tema.

Figur 5.1: Applikationens huvudmeny

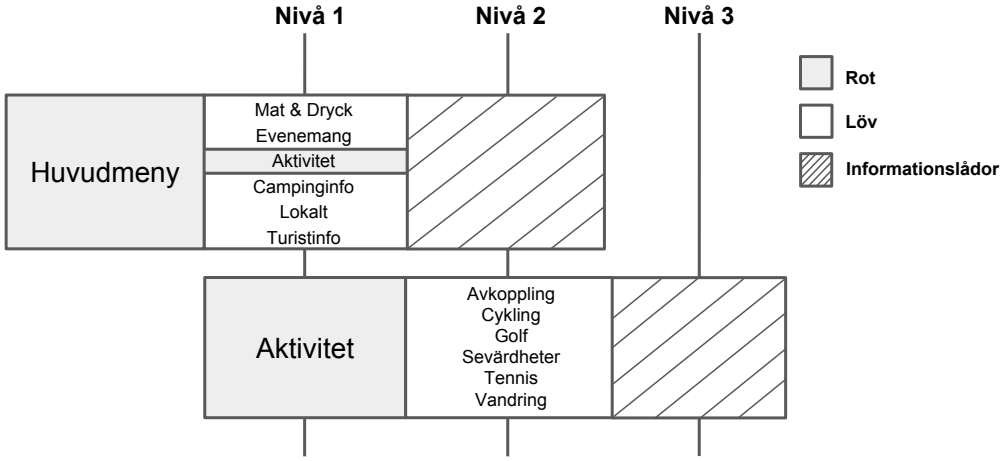
Figur 5.2: Kategorin *Mat & Dryck*

Figur 5.3: Kategorin *Aktivitet*

5.1.1 Design

Menyhierarki
Applikationen har en menyhierarki enligt figur 5.4, hierarkin är tre nivåer djup. Grå färg betecknar rötter, vit färg betecknar löv, streckat är informationslådor, varje löv har en eller flera informationslådor.

På nivå ett finns huvudmenyn, hit kommer användaren när applikationen startas. Huvudmenyn visar sex stycken knappar som representerar sex olika kategorier. Klickar användaren på dessa så tas de till nivå två som är informationslådor för respektive kategori, förutom *Aktivitet* som på nivå två har en egen meny med ytterligare sex kategorier.

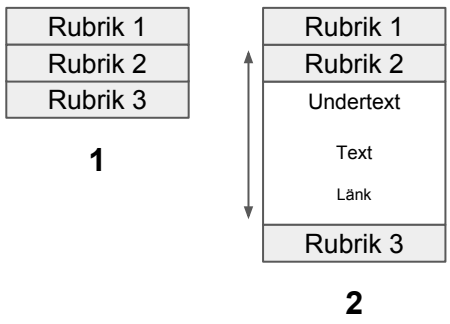


Figur 5.4: Applikationens menyhierarki

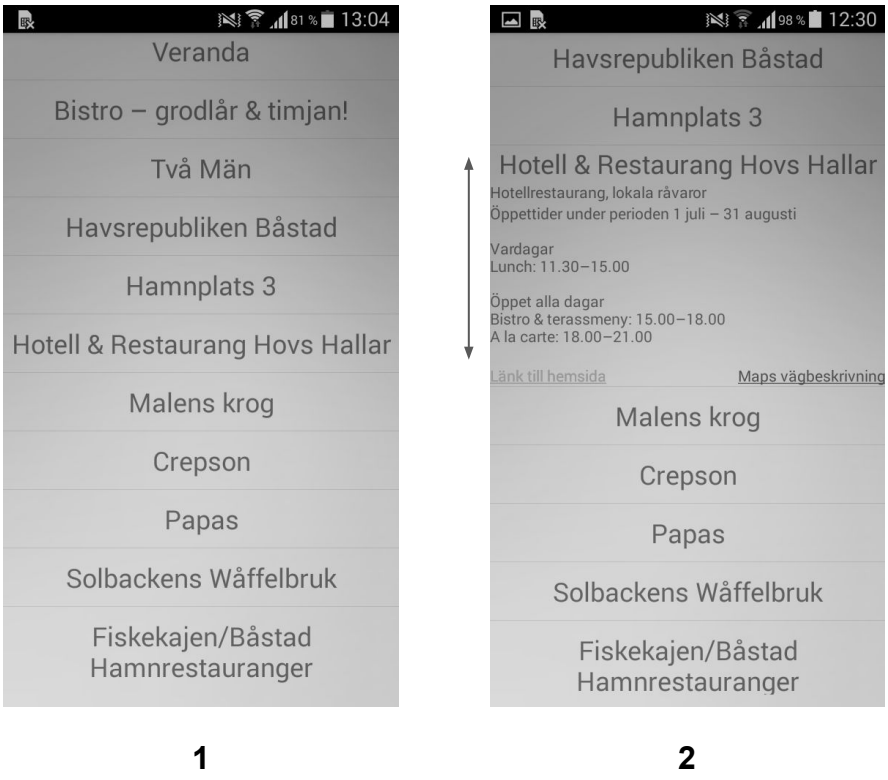
Informationslådor
Informationslådorna håller all information som finns i applikationen. En informationslåda kan vara i två lägen: minimerad (1) eller expanderad (2) enligt figur 5.5. Deras standardläge är minimerat och de expanderar när användaren klickar på informationslådans rubrik och minimeras när användaren klickar på den igen eller på en annan lådas rubrik.

En informationslåda innehåller någon av, eller en kombination av, elementen undertext, text eller länk, dessa element tillhör en rubrik. Det kan finnas flera element av samma typ men de kan bara tillhöra en rubrik.

Figur 5.6 visar konceptet i applikationen.



Figur 5.5: Minimerad och expanderad informationslåda



Figur 5.6: Minimerad och expanderad informationslåda i applikationen

Knappar

Knapparna har filformatet SVG *Scalable Vector Graphics* som är ett XML-baserat vektorgrafikformat. Formatet har bland annat egenskapen att grafikens skärpa inte påverkas vid skalning. Skalning sker när applikationen anpassas till mobiltelefonens skärmstorlek.

Äldre versioner av Android omvandlar en applikations vektorelement från SVG till PNG *Portable Network Graphics* på grund av bristande stöd.

Knapparna i applikationen består av tre vektorelement, en ikon som representerar kategorin, text samt låda.

Figur 5.7: Knappar i applikationen

5.1.2 Information

Aktivitet
Denna kategori innehåller sex ytterligare kategorier av information. Kategorin *Avkoppling* innehåller information om spa, massage, badställen, sjukgymnastik, gym, naprapati och andra aktiviteter som har med avkoppling och hälsa att göra. *Cykling* innehåller information om cykelled enligt två kategorier: mountainbike och landsvägscykling. Kategorin har allmän information relaterat till cykling som exempelvis cykeluthyrning. Kategorin *Golf* innehåller information om golfklubbar, golfbanor med kartor till dessa som öppnas i Google Maps. *Sevärdheter* innehåller information om museum, ateljéer och Bjärehalvöns historia. *Tennis* visar platsinformation om tennisbanor runt hela Bjärehalvön, samt länkar för mer information om uthyrning. *Vandring* har information om vandringsleder i natursköna områden, informationen visar bland annat vandringsledens rutt i Google Maps.

Mat & Dryck
Denna kategori innehåller information om Båstad Campings egna restaurang samt 14 andra matställen på Bjärehalvön som enligt *TripAdvisor* har bäst omdömen. Informationen består av platsinformation, länkar till hemsidor med öppettider samt en kort beskrivning av vad för slags mat som serveras.

Evenemang
Innehåller information om kommande evenemang på hela Bjärehalvön men främst i Båstad och på Båstad Camping. Evenemangen är ordnade efter månad och datum.

Campinginfo
Kategorin innehåller information om Båstad Camping, förhållningsregler, öppettider, kontaktinformation till campingen. Denna kategori är avsedd för besökarna av Båstad Camping.

Lokalt
Denna kategori innehåller information om lokalt näringsliv som till exempel säljer grönsaker, bröd, mejeri och fisk. Informationen består av platsinformation samt korta beskrivningar om det lokala näringslivet. Kategorin innehåller även information om livsmedelsbutiker och platsinformation.

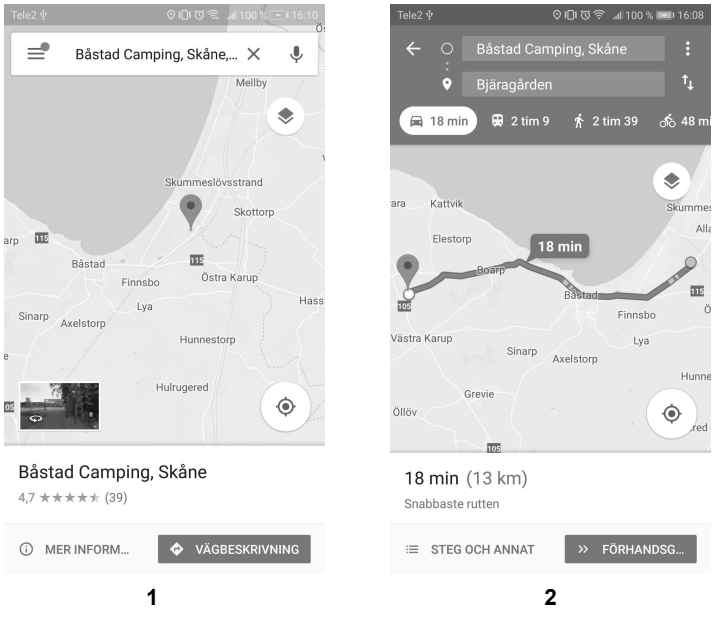
Turistinfo
Denna kategori innehåller information riktad mot turister till exempel lokaltrafik, buss och tåg samt nödnummer till veterinär, till exempel.

5.1.3 Kartfunktionalitet

Applikationen har kartfunktionalitet inom följande kategorier:

1. Aktivitet
    1.1. Avkoppling
    1.2. Vandringsleder
    1.3. Golf
    1.4. Tennis
2. Mat & Dryck
3. Lokalt
4. Turistinfo

Det finns två typer av kartor i applikationen, en med en förutbestämd rutt, (2) i figur 5.8 från Båstad Camping till destinationen och en med platsinformation, (1) i figur 5.8, det vill säga, en punkt markerad på kartan. Kartfunktionaliteten nås genom att användaren klickar på en länk i den valda kategorins informationslåda. Systemet öppnar då Google Maps och visar antingen en rutt eller platsinformation.



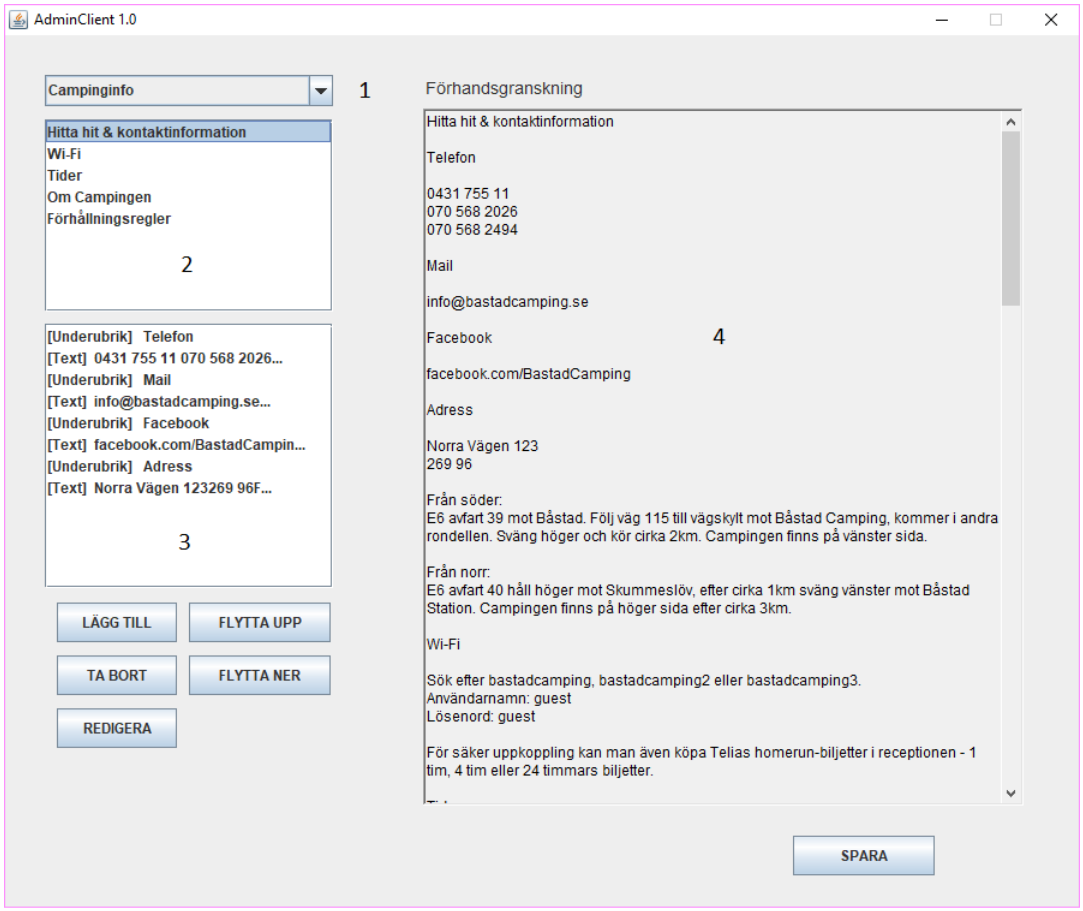
Figur 5.8: Kartor i applikationen

5.2 Administratörsgränssnitt

Resultatet av administratörsgränssnittet är ett Java-baserat program som körs på en dator. Administratörsgränssnittet har funktionalitet för redigering av informationen som visas i applikationen. Varje kategoris information finns i informationslådor, de element som utgör en informationslåda är rubrik, underrubrik, text samt länk.

Administratörsgränssnittet kan byta plats på element (rubriker, underrubriker, texter och länkar), ta bort dem, lägga till nya samt redigera dem. När en ändring utförs och sparas av administratören så skapas en ny XML-fil som sedan laddas upp till molnservern. För att göra ändringar i informationen så väljer administratören först en kategori i rullgardinen (1) enligt figur 5.9. När en kategori valts så fylls de två sidomenyerna (2)(3) med den information som finns i den kategorin. Den övre sidomenyn (2) fylls med rubrikerna och den nedre sidomenyn (3) fylls med de undertexter, texter samt länkar som tillhör den, i övre menyn, valda rubriken.

Gränssnittet har även en ruta för förhandsgranskning (4) som uppdateras vid varje utförd ändring.



Figur 5.9: Administratörsgränssnittet

5.2.1 Hantering av XML-fil

Inläsning av XML-fil

För att administratörsgränssnittet ska visa den befintliga informationen som finns i applikationen så måste XML-filen finnas i samma mapp på datorn som programmet när programmet startas upp. Gränssnittet läser vid uppstart in innehållet från filen enligt figur 5.10. Innehållet som finns mellan taggarna i filen fylls in i gränssnittets redigeringselement (rullgardin, sidomenyer, förhandsgranskning et cetera).

```
public void run() {
      try {
       File fXmlFile = new File("bastad.xml");
       DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
       DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
       Document docu = dBuilder.parse(fXmlFile);
       AdminClient window = new AdminClient(docu);
       window.AdminClient.setVisible(true);
       } catch (Exception e) {
           e.printStackTrace();
       }
}
```

Figur 5.10: Java-kod som laddar in XML-fil

Skrivning till XML-fil

När administratören sparar sitt arbete i administratörsgränssnittet så placeras den nya informationen mellan de taggar där redigeringen utfördes. Till exempel, om redigering utfördes på rubriken *CAMPINGINFO* så sparas den mellan taggarna <CAMPINGINFO >information </CAMPINGINFO>. Sedan skapas en ny XML-fil med den uppdaterade informationen. Filen enligt koden i figur 5.11.

```
public static void saveXML(Document docu) {
      try {
            TransformerFactory trsFactory =  TransformerFactory.newInstance();
            Transformer transformer = trsFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            DOMSource source = new DOMSource(docu);
            StreamResult result = new StreamResult(new File("file2.xml"));
            transformer.transform(source, result);
      } catch(TransformerException tfe) {
            tfe.printStackTrace();
      }
}
```

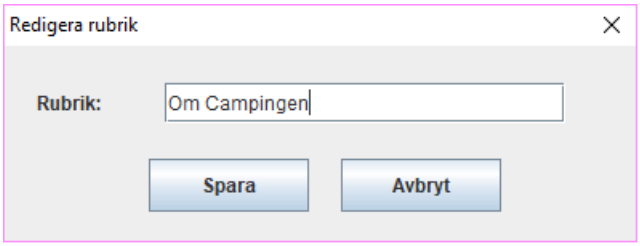
Figur 5.11: Java-kod som skapar ny XML-fil

5.2.2 Funktionalitet

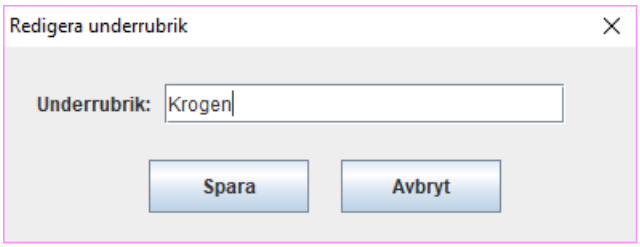
Flytta upp, flytta ner
Med knapparna *FLYTTA UPP* och *FLYTTA NER* kan administratören skifta plats mellan elementen. För att skifta plats mellan rubriker så väljs en rubrik i den övre sidomenyn, när rubriken skiftat plats så följer alla element som tillhör den rubriken med. Element som finns inom en rubrik kan endast skifta plats inom den rubrik de tillhör.

Redigera
Med knappen *REDIGERA* så kan administratören ändra information. För att redigera en rubrik så väljer administratören en rubrik och klickar på redigera. Då öppnas en redigeringsruta enligt figur 5.12.

För att redigera element inom en rubrik så måste först en rubrik väljas och sedan ett element i den nedre sidomenyn. I figur 5.13 så har en underrubrik till rubriken *Om campingen* valts.



Figur 5.12: Redigering av rubrik

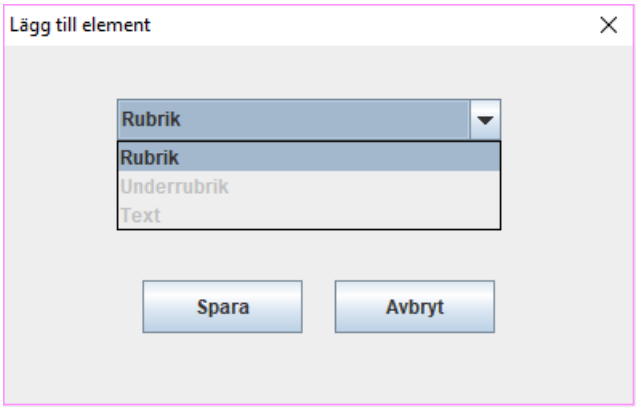


Figur 5.13: Redigering av underrubrik

Lägg till, ta bort
Med knapparna *LÄGG TILL* och *TA BORT* så kan administratören ta bort och lägga till nya element.

För att lägga till ett element så klickar användaren på knappen *LÄGG TILL*. I en ny ruta kan användaren välja vad för slags element som ska skapas från en rullgardin, figur 5.14. För att lägga till elementen underrubrik eller text så måste en rubrik existera annars visas de elementen i grått i rullgardinen. Om en rubrik finns och är vald i menyn så skapas elementen underrubrik och text under samma rubrik.

Vid borttagning så försvinner det valda elementet. När en rubrik väljs för borttagning så försvinner även alla element inom den rubriken.



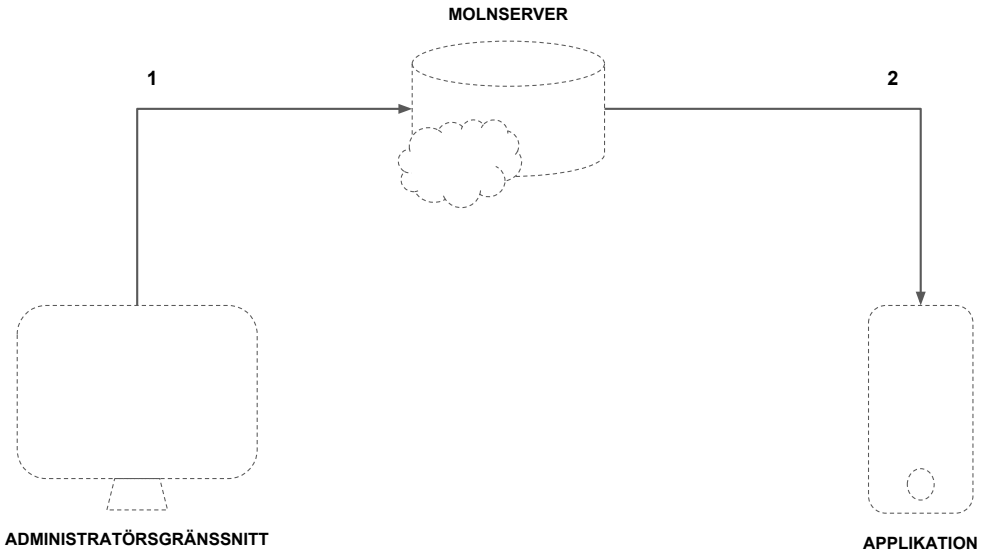
Figur 5.14: Ruta som visas vid tilläggning av element

Spara

När administratören är färdig med ändringarna så sparas arbetet med knappen *SPARA*. Programmet genererar då en XML-fil med den uppdaterade informationen.

5.3 Uppdatering av information

Uppdatering från administratörssidan
Uppdatering av information i applikationen efter redigering sker både från administratörsgränssnitt och applikation. När administratören har redigerat klart och klickar på knappen *SPARA* i gränssnittet så genereras en ny XML-fil med den uppdaterade informationen. Filen förs sedan över manuellt på Google Drive i webbläsaren eller genom att placera den i Google Drives mapp om tjänsten är installerad på datorn. Detta representeras av (1) i figur 5.15.



Figur 5.15: Uppdateringsfunktionalitet i systemet

Uppdatering från applikationssidan
När XML-filen laddats upp så laddar applikationen in XML-filen vid uppstart. Den nya informationen finns då i applikationen. Detta representeras av (2) i figur 5.15. Sker ändringar i XML-fil då applikationen är uppstartad så krävs en omstart för att applikationen ska ha den senaste XML-filen.

En del av koden som laddar ner filen från molnservern och uppdaterar informationen i applikationen visas i figur 5.16. Hela klassen finns i appendix D.1.

```
    .
    .
    .
protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     this.verifyStoragePermissions(this);
     setContentView(R.layout.activity_testdownload);
     downloadTask.execute(link to google drive file omitted);
 }
    .
    .
```

Figur 5.16: Java-kod som laddar ner XML-fil

Kapitel 6
Slutsats

Detta kapitel sammanfattar resultatet av examensarbetet och besvarar problemställningarna. Kapitlet går även igenom framtida utvecklingsmöjligheter.

6.1 Sammanfattning av resultat

Resultatet av examensarbetet är ett system bestående av en besöksapplikation, ett administratörsgränssnitt och molntjänsten Google Drive. Applikationen innehåller information enligt flera olika kategorier som är riktade mot besökare av Båstad Camping och Bjärehalvön i allmänhet. Vissa kategorier i applikationen har även kartfunktionalitet. Kartor finns i två olika format: platsinformation som har en knappnål utplacerad på en karta samt ruttinformation som har en vägbeskrivning från Båstad Camping till en specifik destination. Information kan ändras, läggas till och tas bort från applikationen genom ett administratörsgränssnitt. Ändringen utförs genom att administratörsgränssnittet sparar ner en XML-fil enligt den nya informationen som matats in. Filen ska sedan föras över till Google Drive för att besöksapplikationen ska kunna ladda ner den och uppdatera informationen i applikationen. Detta kan ske på två sätt: manuellt genom en drag-and-drop rörelse av filen till Google Drive i webbläsaren eller genom att Google Drive är nedladdat och att administratörsgränssnittet körs i Google Drive-mappen som skapas vid installation.

6.2 Problemformuleringar

Siffrorna hänvisar till problemformuleringarna definierade i kapitel 1.4.

1: *Vilken typ av funktionalitet är viktigast för prototypen?*
Den viktigaste typen av funktionalitet för prototypen är möjligheten att utföra ändringar i applikationen, administratörsfunktionalitet. Denna funktionalitet är viktigast eftersom den gör applikationen relevant över tid, utdaterad information kan tas bort och ny information läggas till.

2: *Finns det ytterligare funktionaliteter som är relevant specifikt för Båstads camping?*
Andra viktiga funktionaliteter är att applikationen ska innehålla samlad information för campingens besökare, till exempel campinginformation och turistinformation. Detta är viktigt eftersom att applikationen ska kunna ersätta broschyrer, hemsidor och andra informationskällor.

3: *Hur kan ett administratörsgränssnitt utformas för att vara lättanvänt för personer utan teknisk utbildning?*
Det kan utformas som ett Java-program som körs på en dator med vyer och knappar som följer designkonventioner. Det kan utformas på detta vis eftersom att det är en utformning som många idag är vana vid och har stött på i sin vardag, ett program på en dator.

4: *Hur kan prototypen testas och vilka tester ska utföras för att kunna säkerställa att funktionerna är fungerande och relevanta för användarna?*
Testning med scenario kan användas för att säkerställa att prototypen fungerar. Scenarion kan användas därför att de på ett verklighetstroget sätt testar funktioner.

5: *Vilken typ av server lämpar sig bäst för denna typ av system?*
Den typ av server som bäst lämpar sig är en server i form av en molntjänst, till exempel Google Drive. Molntjänsterna OneDrive och Dropbox undersöktes också som möjliga kandidater men valdes i huvudsak bort på grund av att Google Drive erbjöd större lagringskapacitet som är gratis.

5.1: *Kommunikation till mobilapplikationen: Vilka möjligheter och restriktioner finns det med att implementera och använda sig av gratis molntjänster som molnserver.*
Restriktionerna med att använda en gratis molntjänst som molnserver är begränsningar i lagringskapacitet, begränsningar i molntjänstens API-stöd för vissa programmeringsspråk samt att större applikationer hade krävt en databashanterare, en gratis molntjänst passar därför bara mindre applikationer. Möjligheterna är att mindre applikationer som inte innehåller mycket information kan utnyttja gratis molntjänster som molnserver.

5.2: *Administratörsgränssnitt: Hur kan man använda en gratis molntjänst som molnserver och genom denna tillåta administratörer utan teknisk bakgrund att göra ändringar i en applikation?*
Administratörer kan utföra ändringar i en applikation med hjälp av ett administratörsgränssnitt som laddar upp en XML-fil till en gratis molntjänst som agerar molnserver. En nedladdningsfunktion kan implementeras i applikationen för att vid uppstart ladda ner XML-filen och uppdatera innehållet i applikationen. Genom att tillåta ändringar från ett administratörsgränssnitt så kräver det inget avancerat tekniskt kunnande från administratörerna.

6: *Vilka APIer kan användas för att anpassa kartor vid utvecklandet av en mobilapplikation?*
Enligt Googles dokumentation av Google Maps API så kan Google Maps API användas för att anpassa kartor. För att implementera kartfunktionalitet krävs inget API. I examensarbetet så integrerades Google Maps utan att använda Google Maps API eftersom att de önskvärda anpassningarna ej krävde Google Maps API.

7: *Hur kan ett gränssnitt i en mobilapplikation utformas för att underlätta navigation?*
För att underlätta navigation kan gränssnittet använda sig av designkonventioner som: tydlig menyhierarki, menynamn för att veta var i menyhierarkin man befinner sig, stora knappar med ikoner som beskriver kategorier. Detta kan användas eftersom att det enligt litteratur och tester underlättar navigering.

6.3 Reflektion över etiska aspekter

Samhällsnytta
Samhällsnyttan av detta examensarbete är att applikationen innehåller turistinformation som inte bara riktar sig mot Båstad Campings besökare men även mot besökare av Bjärehalvön. Applikationen som har skapats kan användas av turister för att hitta mer information och uppfyller en slags turistinformations-funktionalitet där informationen finns samlad på ett ställe.

Med administratörsfunktionaliteten kan information om till exempel lokalt näringsliv och andra ställen läggas till. Detta kan potentiellt hjälpa näringslivet att få flera kunder och olika turistattraktioner fler besökare då applikationen är tillgänglig för alla som har Android.

Med administratörsfunktionalitet kan även felaktig information läggas till i applikationen. Detta kan leda till att användarna av applikationen inte hittar rätt om informationen i en karta är felaktig. Effekten av sådana händelser kan bli att användarna vilseleds och att applikationens information kan uppfattas som icke trovärdig, något som hade kunnat leda till att den används i mindre utsträckning.

Dataintrång
Om kontot som är kopplat till Google Drive som styr innehållet i applikationen utsätts för dataintrång så kan webblänkar i applikationen ersättas med länkar som går till webbsidor som exponerar användaren mot nätfiske och andra sätt som kan stjäla personuppgifter.

Olämpligt material hade också kunnat spridas, till exempel information om privatpersoner och politisk propaganda.

6.4 Framtida utvecklingsmöjligheter

Under examensarbetets utveckling så kom det fram många idéer kring funktionalitet i systemet. Detta avsnitt går igenom några av dessa idéer som kan passa bra för framtida vidareutveckling.

Språk
Båstad Camping har även besökare från andra länder, en engelskspråkig version kan implementeras där språk väljs antingen vid uppstart eller genom menyalternativ.

iOS
Applikationen har endast utvecklas för operativsystemet Android, en framtida utvecklingsmöjlighet kan vara att utveckla applikationen till iOS.

Design
Applikationens design kan även förbättras genom att studera UX-design och skapa skarpare design-element.

Notifikationer
Ett utvecklingsförslag som togs upp under projektet men som inte utvecklades var möjligheten att skapa funktionalitet i administratörsgränssnittet där administratörerna kan skicka ut notiser till alla som har applikationen installerad. Notiserna kan t.ex. vara påminnelser om evenemang på campingen och andra informationsutskick.

Uppdateringsfunktionalitet
Systemet uppdaterar information genom att administratören klickar på knappen *SPARA* i administratörsgränssnittet och sedan för över den XML-fil som genereras till Google Drive. Uppdatering kan även ske om Google Drive är nedladdat och administratörsgränssnittet startas upp i en Google Drive mapp, då sker uppdateringen utan att administratören behöver hantera XML-filen. XML-filen laddas sedan ner av applikationen vid uppstart.

En utvecklingsmöjlighet hade varit att använda Google Drive API i administratörsgränssnittet för att ladda upp filen utan att behöva handskas med Google Drive på varken dator eller webbläsare, uppdatering sker direkt när administratören sparar arbetet.

En utvecklingsmöjlighet för applikationen hade varit att skapa en funktion som kontinuerligt undersöker om XML-filen uppdaterats och sedan laddar ner den när en uppdatering sker.

Skal för besöksapplikationer
Det system som tagits fram i detta examensarbete är anpassat för Båstad Camping i synnerhet och Bjärehalvön i allmänhet.

En utvecklingsmöjlighet hade varit att undersöka möjligheten att skapa hela systemet som ett tomt skal där sedan kommun och näringsliv kan ladda ner hela systemet och fylla i sin egna information. Det som hade behövt undersökas är hur de statiska delarna i systemet kan göras ändringsbara, som t.ex. ikoner och menyrubrik.

Litteraturförteckning

[1] Android developers. https://developer.android.com/(visited on 14/05/2018).

[2] Android developers - emulator. https://developer.android.com/studio/run/emulator(visited on 14/05/2018).

[3] Båstad & bjärehalvön - the official travel and tour information. https://www.bastad.com/sv(visited on 15/05/2017).

[4] Eclipse - windowbuilder. https://www.eclipse.org/windowbuilder/(visited on 22/05/2018).

[5] Inkscape. https://inkscape.org/en/about/(visited on 31/05/2018).

[6] Java doc - documentbuilderfactory. https://docs.oracle.com/javase/7/docs/api/javax/xml/parsers/DocumentBuilderFactory.html(visited on 22/05/2018).

[7] Java garbage collection basics. http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html(visited on 12/04/2018).

[8] Managing memory and garbage collection. https://docs.oracle.com/cd/E19159-01/819-3681/6n5srlhqf/index.html(visited on 12/04/2018).

[9] proto.io. https://proto.io(visited on 20/03/2018).

[10] Stroustrup: Faq. http://www.stroustrup.com/bs_faq.html#garbage-collection(visited on 12/04/2018).

[11] w3schools - xml introduction. https://www.w3schools.com/xml/xml_whatis.asp(visited on 16/05/2018).

[12] James H. Frey Andrea Fontana. The art of science. http://jan.ucc.nau.edu/~pms/cj355/readings/fontana%26frey.pdf(visited on 20/03/2018).

[13] Mattias Skarin Henrik Kniberg. Kanban and scrum, making the most of both. http://www.agileinnovation.eu/wordpress/wp-content/uploads/2010/09/KanbanAndScrum_MakingTheMostOfBoth.pdf(visited on 15/03/2018).

[14] Janice C. Redish Joseph S. Dumas. *A Practical Guide to Usability Testing*. Intellect Books, 1999.

[15] Steve Krug. *Don't Make Me Think, Revisited*. New Riders Publishing, 2014.

[16] Soren Lauesen. *Software Requirements: Styles and Techniques*. Addison-Wesley, 2002.

[17] Bjarne Stroustrup. *The C++ Programming Language, Special Edition*. Addison-Wesley, 2003.

[18] Helen Sharp Yvonne Rogers, Jennifer Preece. *Interaction Design - Beyond human-computer interaction*. John Wiley and Sons Ltd, 2015.

Appendix

Appendix A
Krav

A.1 Funktionella krav

A.1.1 Funktionella krav, applikation

Kravnr.	Beskrivning
1	Följande ska ske när applikationen startas: 1. Applikationen laddar in XML-fil från Google Drive. 2. Huvudmenyn visas.
2	När användaren klickar på knappen "AKTIVITET" i huvudmenyn så ska en ny meny visas med sex knappar för sex olika aktiviter.
3	När användaren klickar på knappen "MAT & DRYCK" i huvudmenyn så ska en undermeny om mat och dryck visas.
4	När användaren klickar på knappen "EVENEMANG" i huvudmenyn så ska en undermeny om evenemang visas.
5	När användaren klickar på knappen "CAMPINGINFO" i huvudmenyn så ska en undermeny om campinginformation visas.
6	När användaren klickar på knappen "LOKALT" i huvudmenyn så ska en undermeny om lokalt visas.
7	När användaren klickar på knappen "TURISTINFO" i huvudmenyn så ska en undermeny om turistinformation visas.
8	När användaren klickar på "Länk" så ska en hemsida öppnas i systemets webbläsare.
9	Scenario 9 ska stödjas. Scenario 9 - Öppna kartlänk. Förutsättning: Användaren har klickat på en kartlänk. Systemet har Google Maps installerat. GPS-positionering är aktiverat. 1. Google Maps öppnas.
10	Scenario 10 ska stödjas. Scenario 10 - Förfrågan om aktivering av GPS-positionering. Förutsättning: Användaren har klickat på en kartlänk. Systemet har Google Maps installerat. GPS-positionering är inaktiverat. 1. En popup-ruta öppnas med en uppmaning att aktivera GPS-positionering (enligt figur Continue anyway/GPS-settings). 2. a) Användaren klickar på knappen “Continue anyway”. b) Användaren klickar på knappen “GPS-settings”. 3. a) Ursprungsmenyn visas. b) Systemets meny för att slå på GPS-positionering öppnas.

A.1.2 Funktionella krav, administratörsgränssnitt

Kravnr.	Beskrivning
11	Scenario 11 ska stödjas. Scenario 11 - Redigera element. Förutsättning: Användaren har valt ett element i menyn. 1. Användaren klickar på knappen “REDIGERA”. 2. En text-ruta öppnas där elementet kan redigeras. 3. Användaren redigerar elementet. 4. a) Användaren klickar på knappen “SPARA”. b) Användaren klickar på knappen “AVBRYT”. 5. a) Det redigerade elementet sparas. b) Användaren skickas till ursprungsmenyn.
12	Scenario 12 ska stödjas. Scenario 12 - Ta bort element. Förutsättning: Användaren har valt ett element i menyn. 1. Användaren klickar på knappen “TA BORT”. 2. Elementet tas bort.
13	Scenario 13 ska stödjas. Scenario 13 - Flytta upp element. Förutsättning: Användaren har valt ett element i menyn. Det valda elementet är inte längst upp. 1. Användaren klickar på knappen “FLYTTA UPP”. 2. Elementet byter plats med ovanstående element.
14	Scenario 14 ska stödjas. Scenario 14 - Flytta ner element. Förutsättning: Användaren har valt ett element i menyn. Det valda elementet är inte längst ner. 1. Användaren klickar på knappen “FLYTTA NER”. 2. Elementet byter plats med nedanstående element.
15	Scenario 15 ska stödjas. Scenario 15 - Lägg till element. Förutsättning: Användaren har valt ett element i menyn. 1. Användaren klickar på knappen “LÄGG TILL”. 2. Ett tomt element läggs till i vald meny.
16	Scenario 16 ska stödjas. Scenario 16 - Spara information till XML-fil. Förutsättning: Användaren befinner sig i huvudmenyn. 1. Användaren klickar på knappen “SPARA”. 2. En XML-fil skapas med uppdaterad information.

A.2 Designkrav

A.2.1 Designkrav, applikation

Kravnr.	Beskrivning
17	Alla knappar ska ha dimension om 14 x 14 pixlar.
18	Alla knappar ska ha ikoner.
19	Alla knappars text ska vara skrivna i versaler.
20	Alla menyer ska ha en meny-rubrik.
21	Alla länkar ska vara följande: 1. Understrukna. 2. Blå när de är oanvända. 3. Lila när de är använda.
22	Alla informationslådor ska ha en rubrik.
23	Informationslådor ska ha följande beteenden: 1. När användaren klickar på en informationslåda som inte är expanderad så ska den expandera. 2. När användaren klickar på en informationslåda som är expanderad så ska den kollapsa. 3. En expanderad informationslåda ska kollapsa om en annan informationslåda expanderas.

A.2.2 Designkrav, administratörsgränssnitt

Kravnr.	Beskrivning
24	När markören vilar på ett objekt i rullgardinen ska objektet markeras med blå färg.
25	När ett objekt i rullgardinen väljs så ska objektet markeras med blå färg.
26	När ett objekt väljs i den översta menyn så ska det markeras med blå färg.
27	När ett objekt väljs i den nedre menyn så ska följande hända: 1. Objektet i den övre menyn ska markeras med grå färg. 2. Objektet i den nedre menyn ska markeras med blå färg."
28	Redigeringsfält ska ha en textmarkör.
29	När markören vilar på en knapp så ska knappens kanter markeras med blå färg.
30	När en knapp blir klickad på så ska knappen markeras med blå färg.

A.3 Datakrav

A.3.1 Datakrav, applikation

Kravnr.	Beskrivning
31	Applikationen ska innehålla information om följande: 1. Lokalt näringsliv. 2. De, enligt Tripadvisor, 14 bäst rankade restaurangerna på Bjärehalvön. 3. Turistinformation. 4. Bjärehalvön. 5. Avkoppling. 6. Tennis. 7. Golf. 8. Cykling. 9. Vandringsleder. 10. Sevärdheter.

A.3.2 Datakrav, administratörsgränssnitt

Kravnr.	Beskrivning
32	Följande information ska kunna förändras i Administratörsgränssnittet: 1. Lokalt näringsliv. 2. Mat & dryck 3. Turistinformation. 4. Bjärehalvön. 5. Avkoppling. 6. Tennis. 7. Golf. 8. Cykling. 9. Vandringsleder. 10. Sevärdheter.

A.3.3 Datakrav, molnserver

Kravnr.	Beskrivning
33	En XML-fil ska kunna lagras på molnservern.
34	En XML-fil ska kunna laddas upp till molnservern.
35	En XML-fil ska kunna laddas ner från molnservern.

Appendix B
Test

B.1 Enkät-test, applikation

Åsikter om			
Knappar	Knapptext	Typsnitt, menyrubrik	Kontrast

B.2 Scenario, applikation

Scenario	1.
Mål	Navigera till specificerad kategori
Kategori	Aktivitet < Tennis
Antal nödvändiga klick	två
Beskrivning	Du vill hitta information om var det finns tennisplaner i Båstad. Navigera till den kategori där du kan hitta information om detta.

Scenario	2.
Mål	Navigera till specificerad kategori
Kategori	Turistinformation < Lokaltrafik
Antal nödvändiga klick	två
Beskrivning	Du vill åka buss in till centrum. Navigera dig fram till information om lokaltrafik.

Scenario	3.
Mål	Navigera till specificerad kategori
Kategori	Aktivitet < Avkoppling
Antal nödvändiga klick	två
Beskrivning	Du vill gå på spa. Hitta information om spa i Båstad.

Scenario	4.
Mål	Navigera till specificerad kategori
Kategori	Mat & Dryck < Restaurang Båstad Camping
Antal nödvändiga klick	två
Beskrivning	Du vill se vad campingen har för mat att erbjuda. Hitta campingens meny.

Scenario	5.
Mål	Navigera till specificerad kategori och öppna kartlänk
Kategori	Aktivitet < Vandring < Hovs Hallar < Kartlänk
Antal nödvändiga klick	fyra
Beskrivning	Du vill se en karta med en vägbeskrivning från Båstad Camping till Hovs Hallar. Hitta denna information.

B.3 Scenario, administratörsgränssnitt

Scenario	1.
Funktion	Flytta upp/ner
Kategori	Campinginformation
Antal nödvändiga rörelser	Musklick: 4, Tangentbord: 0
Beskrivning	Du vill göra informationen om campingens "Wi-Fi" mer synlig och vill därför ha den ovanför informationen som finns om "Kontaktinformation". Byta plats på "Wi-Fi" och "Kontaktinformation".

Scenario	2.
Funktion	Ta bort
Kategori	Evenemang
Antal nödvändiga rörelser	Musklick: 5, Tangentbord: 0
Beskrivning	Ett evenemang som redan har varit finns fortfarande kvar i applikationen. Ta bort ett evenemang som redan varit.

Scenario	3.
Funktion	Lägg till rubrik och text
Kategori	Evenemang
Antal nödvändiga rörelser	Musklick: 11, Tangentbord: 2
Beskrivning	Campingen ska ha grillfest och du vill ha med detta i appen. Lägg till en rubrik och text för grillfesten i kategorin "Evenemang".

Scenario	4.
Funktion	Redigera länk
Kategori	Turistinformation
Antal nödvändiga rörelser	Musklick: 12, Tangentbord: 1
Beskrivning	Du vill ha med nyheter i turistinformationen. Lägg till en rubrik "Nyheter" och till denna rubrik en länk till en nyhetssajt.

Scenario	5.
Funktion	Redigera kartlänk
Kategori	Mat & Dryck
Antal nödvändiga rörelser	Musklick: 6, Tangentbord: 1
Beskrivning	Restaurangen "Veranda" har bytt adress till Strandpromenaden 110. Uppdatera adressen.

Scenario	6.
Funktion	Spara arbete
Antal nödvändiga rörelser	Musklick: 1, Tangentbord: 0
Beskrivning	Du är klar med ändringarna. Spara arbetet.

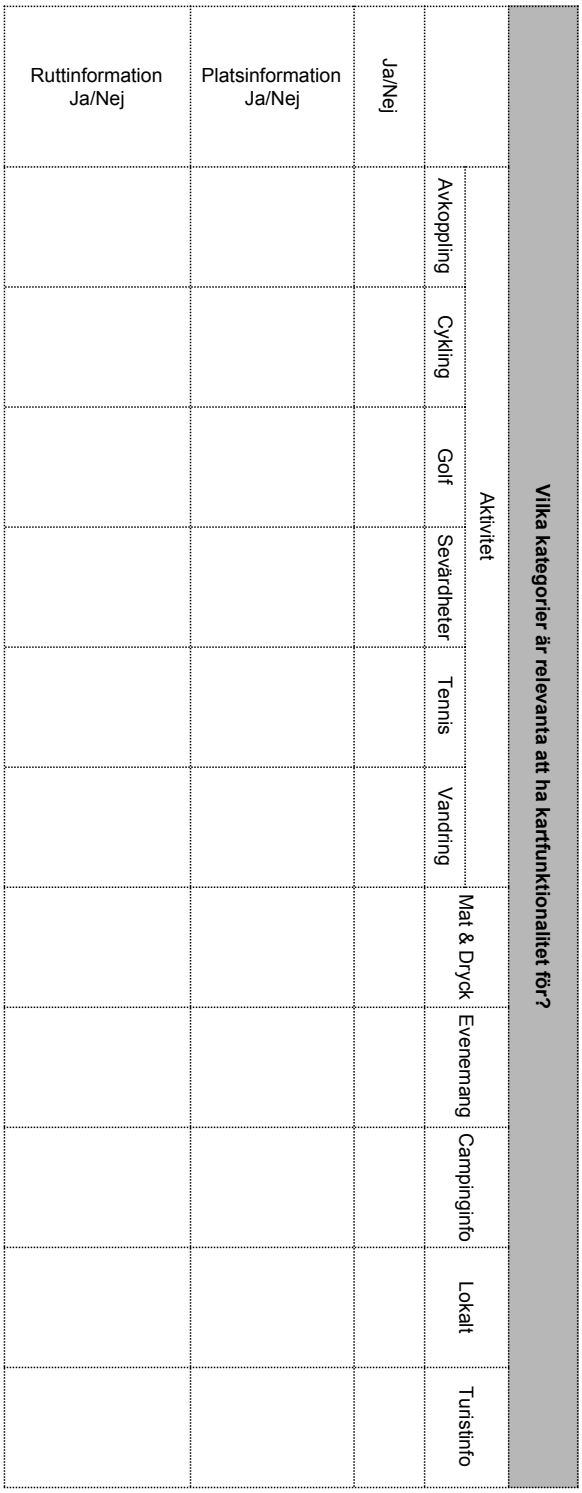
Appendix C
Elicitering

C.1 Intervju

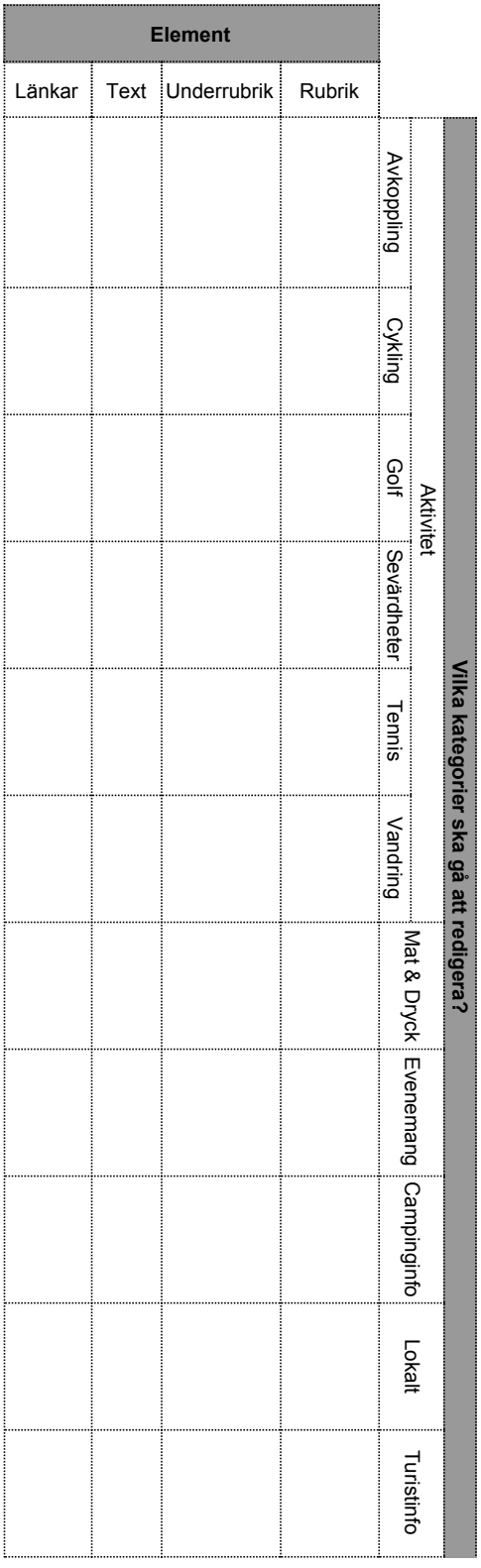
C.1.1 Strukturerad intervju, slutanvändare, fas två

Ålder	Stamgäst	Favoritaktivitet	Vill ni veta mer om Båstad/Bjäre.?	Hur får ni tag i information?

C.1.2 Semi-strukturerad intervju, Båstad Camping, fas tre



C.1.3 Semi-strukturerad intervju, Båstad Camping, fas fyra



Appendix D
Kod

D.1 Nedladdning av XML-fil

```
package com.example.dat11nse.myapplication;

import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.AsyncTask;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class LaunchScreen extends AppCompatActivity {

    final DownloadTask downloadTask = new DownloadTask(LaunchScreen.this);
    private static final int REQUEST_EXTERNAL_STORAGE = 1;
    private static String[] PERMISSIONS_STORAGE = {
            Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.WRITE_EXTERNAL_STORAGE
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.verifyStoragePermissions(this);
        setContentView(R.layout.activity_launchscreen);
        downloadTask.execute(omitted);

    }

    public static void verifyStoragePermissions(LaunchScreen activity) {
        int permission = ActivityCompat.checkSelfPermission(activity,
Manifest.permission.WRITE_EXTERNAL_STORAGE);
        if (permission != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(
                    activity,
                    PERMISSIONS_STORAGE,
                    REQUEST_EXTERNAL_STORAGE
            );
        }
    }

    private class DownloadTask extends AsyncTask<String, Integer, String> {

        private Context context;

        public DownloadTask(Context context) {
            this.context = context;
        }

        @Override
```

```
        protected void onPostExecute(String result) {
            if (result != null)
                Toast.makeText(context,"Download error: "+result, Toast.LENGTH_LONG).show();
                    // @ TODO
                //if filen redan finns nerladdad så gå ändå vidare och kör på gammal data
            else {
                //  Toast.makeText(context,"File downloaded Radera detta meddeöamde mär appen är klar",
Toast.LENGTH_SHORT).show();
                Intent intent = new Intent (LaunchScreen.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        }

        @Override
        protected String doInBackground(String... sUrl) {
            InputStream input = null;
            OutputStream output = null;
            HttpURLConnection connection = null;
            try {
                URL url = new URL(sUrl[0]);
                connection = (HttpURLConnection) url.openConnection();
                connection.connect();
                if (connection.getResponseCode() != HttpURLConnection.HTTP_OK) {
                    return "Server returned HTTP " + connection.getResponseCode()
                            + " " + connection.getResponseMessage();
                }
                // download när den vet att den har internet. destination och namn bör bytas.
                input = connection.getInputStream();
                output = new FileOutputStream("/sdcard/test.xml");
                byte data[] = new byte[4096];
                int count;
                while ((count = input.read(data)) != -1) {
                    if (isCancelled()) {
                        input.close();
                        return null;
                    }
                    output.write(data, 0, count);
                }
            } catch (Exception e) {
                return e.toString();
            } finally {
                try {
                    if (output != null)
                        output.close();
                    if (input != null)
                        input.close();
                } catch (IOException ignored) {
                }
                if (connection != null)
                    connection.disconnect();
            }
            return null;
        }
    }
}
// Frodo: It’s gone. It’s done.
// Sam: Yes, Mister Frodo. It’s over now.
```