

# Förenklandet av studentbostadsökandet i Göteborg

UTVECKLANDET AV MOBILAPPLIKATIONEN  
STUDENTBOSTÄDER GÖTEBORGS

*Jonathan Gildevall, Peter Gärdenäs,  
Erik Magnusson & John Segerstedt*

CHALMERS TEKNISKA HÖGSKOLA  
LSP310 KOMMUNIKATION OCH INGENJÖRSKOMPETENS  
29 maj 2017

## **Sammandrag**

Rapporten redogör för processen i utvecklandet av en mobilapplikation vars syfte är att underlätta bostadssökandet för studenter i Göteborg. Vi-dare beskriver rapporten hur applikation utvecklades med hjälp av kod-design och en iterativ process för framtagning av ny funktionalitet.

Resultatet redovisar vad för syftesuppfyllande funktioner mobilappli-kationen Studentbostäder Göteborg fick samt problem vid implemente- ringen av dessa. Slutprodukten innehåller all den högst prioriterade funk-tionaliteten, men på grund av tidsbrist finns det fortfarande kvarstående poleringsarbete samt en viss avsaknad av funktionalitet vilket medför att applikationen inte fullt uppfyller sitt mål att underlätta bostadssökandet för studenter.

# Innehåll

<b>1 Inledning</b>	<b>1</b>
1.1 Syfte . . . . .	1
<b>2 Teori</b>	<b>2</b>
2.1 Domändriven design . . . . .	2
2.2 Externa bibliotek . . . . .	2
2.2.1 Databasbibliotet Db4o . . . . .	2
2.2.2 Serialiseringsbiblioteket Google Gson . . . . .	2
2.3 Model-View-Controller . . . . .	3
<b>3 Metod &amp; genomförande</b>	<b>4</b>
3.1 Projektets metod & arbetsgång . . . . .	4
3.2 Genomförande av projektimplementationen . . . . .	4
<b>4 Resultat</b>	<b>5</b>
4.1 Implementation av applikationen . . . . .	5
4.1.1 Hämtandet av bostadsdata . . . . .	5
4.1.2 Lagring av objekt med hjälp av Db4o . . . . .	5
4.1.3 Implementering av Model-View-Controller . . . . .	5
4.1.4 Androids påverkan på projektet . . . . .	6
4.2 Användarvänlighet . . . . .	6
4.2.1 Applikationens responsivitet . . . . .	6
4.2.2 Applikationens funktionalitet . . . . .	6
4.2.3 Målgruppsanpassning . . . . .	7
4.2.4 Kvarstående programfel . . . . .	7
<b>5 Diskussion</b>	<b>8</b>
5.1 Arbetsmetodsdiskussion . . . . .	8
5.2 Resultatdiskussion om implementationen . . . . .	8
5.2.1 Val av databas . . . . .	8
5.2.2 Hämtandet av data . . . . .	8
5.2.3 Model-View-Controller implementering . . . . .	9
5.3 Resultatdiskussion angående användarvänlighet . . . . .	9
5.3.1 Diskussion kring funktionalitet . . . . .	9
5.3.2 Målgruppsanpassning . . . . .	9
5.3.3 Kvarstående programfel . . . . .	9
5.3.4 Sammanfattning av användarvänlighet . . . . .	10
5.4 Möjliga förbättringar . . . . .	10
<b>Källförteckning</b>	<b>11</b>

# 1 Inledning

Bostadsmarknaden blir alltmer svår att ta sig in på [1][2], vilket leder till att studenter behöver lägga ner mycket tid och energi på att leta bostad. I Göteborg försvåras processen ytterligare av att man behöver leta genom flera olika hem-sidor. De två ledande bostadsbolagen för studenter i Göteborg är Chalmers Studentbostäder, (*Chalmers*) och Stiftelsen Göteborgs Studentbostäder, (*SGS*) som idag är Göteborgs största aktör inom studentbostadsbranschen [2].

Då Chalmers nyligen skapade sin egna bostadskö [3] har de infört ett lotterisystem när de ger ut hyreskontrakt. Detta har lett till att deras bostäder öppnas upp till en mycket större publik då det nu finns en chans för personer med kortare kötid att ändå få en bostad.

Eftersom studentbostäderna som SGS och Chalmers erbjuder inte finns på samma plattform leder detta till stort besvärs. På grund av detta utvecklades mobilapplikationen Studentbostäder Göteborg.

## 1.1 Syfte

Rapporten redogör för utvecklandet av en Androidapplikation vars mål är att förenkla sökandet av studentbostäder i Göteborgsområdet genom att erbjuda mer funktionalitet än vad bostadsföretagens egna sidor erbjuder, samt presentera de båda största företagens bostäder i samma plattform. Därtill är rapportens syfte att diskutera processen och resultatet av projektet.

## 2 Teori

Under projektets gång användes domändriven design, externa bibliotek implementerades och mjukvarudesignmönstret Model-View-Controller följdes. Nedan kommer förklaringar till dessa begrepp med syftet att underlätta förståelsen för resten av rapporten.

### 2.1 Domändriven design

Vid användning av domändriven design under utvecklingen av ett mjukvaruprojekt blir modelleringssarbetet mindre komplicerat eftersom projektet delas upp i mindre, samt mer lättanterliga delar [4]. För att få kontroll över komplicerade problem är den viktigaste delen i lösningen en domänmodell [5].

Med domän syftar man på det område användaren använder programmet på. Domänen för detta projektets applikation är bostadssökning. En modell är en utvald, förenklad och strukturerad bild av kunskap [5]. En domänmodell är en modell i den valda domänen. För mer avancerade projekt är det vanligt med flera domänmodeller i samma domän.

Fördelarna med domändriven design är att det skapar ett gemensamt språk som utvecklarna, men även andra på företaget eller inom projektet, kan använda för att prata om arbetet. En annan fördel är att man inte behöver gå igenom hela koden för att förstå hur den sitter ihop då det framgår i modellen [4]. Modellen blir en gemensam punkt som hela arbetsgruppen känner till och bygger sina delar utifrån.

### 2.2 Externa bibliotek

Programvaruutveckling kan många gånger underlättas av olika externa databibliotek och ramverk. Ett databibliotek är en stor samling återanvändbar kod vilken kan återanvändas i ett nytt projekt. För att importera bibliotek utöver Javas standardbibliotek användes Androids inbyggda byggverktyg, Gradle.

#### 2.2.1 Databasbibliotet Db4o

Db4o är ett databasbibliotek som tillåter användaren att ladda och spara instanser av bland annat Java-objekt. Db4o är smidigt att komma igång med, databasen behöver inte definiera gränssnitt eller attribut i förväg utan olika Java-objekt kan sparas med en rad kod. En nackdel är att databasen inte längre utvecklas [6].

#### 2.2.2 Serialiserbiblioteket Google Gson

Gson är ett bibliotek som används för att konvertera Java-objekt till JSON-objekt eller tvärtom, JSON är ett textbaserat filformat som används vid överföring av data. Gson är bland annat användbart för att tolka data som hämtas från nätet eftersom datan många gånger använder JSON som format.

## 2.3 Model-View-Controller

Model-View-Controller, (*MVC*), är ett mjukvarudesignmönster vilket kodmässigt tydligt separerar ansvarsområden i tre delar [7]. Den första av de tre delarna är Model (*modell*). Denna del har ansvar över att hålla i all data. I ett spel är spelare, monster, karta, och vapen exempel på modellklasser, vilka är mallar för objektsattribut och kopplingar. Den andra delen, View (*vy*), har ansvar för användargränssnitt och datapresentation. Vyfiler är bland annat de filer som presenterar gränssnittsfönster, menyer, knappar etc. Det är också vyfilernas uppdrag att uppdatera vad som ritas ut på skärmen allteftersom programtillståndet förändras. Den sista delen, som kallas Controller (*kontroller*), ansvarar för att gränssnittet samt programmets tillstånd ändras alltefter användarinmatningar [8].

Implementation av MVC är strukturellt fördelaktig eftersom att det minskar antalet kopplingar mellan de olika delarna i ett program. Bland annat behöver modellen inte ha en referens till någon av vyerna eftersom de kan uppdatera det som ritas ut på skärmen varje gång kontrollernas metoder, där modelltillstånden ändras, kallas. Den enda ordentliga kopplingen som behövs mellan dessa paket är att kontrollen håller i en referens till modellen, då kontrollen behöver kunna ändra på modellens tillstånd [7].

Minimeringen av antalet kopplingar är gynnsamt då det förenklar processen att byta ut samt uppdatera kod men även underlättar återanvändandet av kod. Det vanligaste exemplet är att man återanvänder modellen till andra vyer. Ifall ett program har ett paket, innehållandes filer med en avskärmad implementation av ett visst ansvarsområde, kan återanvändandet av paketet i det framtida projektet ske väldigt enkelt även ifall all annan kod ser annorlunda ut än i ursprungsprojektet [7].

## 3 Metod & genomförande

I detta kapitlet beskrivs arbetsgången samt genomförandet av Studentbostäder Göteborgs utveckling.

### 3.1 Projektets metod & arbetsgång

Det första steget i utvecklandet av Studentbostäder Göteborg var bestämmandet av applikationens krav. I detta skede bestämdes vilken funktionalitet som applikationen borde innefatta samt eventuellt skulle kunna läggas till i mån av tid. Exempel på specificerad funktionalitet var de bostadsattribut man skall kunna använda som sökkriterier och hur den grundläggande navigationen mellan de olika vyerna skulle representeras.

Därefter togs en domänmodell fram i syfte att underlätta applikationens utvecklande. Denna domänmodell låg till grund till designen bakom projektets tidigaste iteration. När domänmodell var sammanställd påbörjades den iterativa processen av designval, implementering av ny funktionalitet, samt programavlusning (*debugging*). Under projektets gång har det varit tydligt vilket typ av funktionalitet som skall byggas vilket har lett till att huvudfokuset har legat på den iterativa processen.

Innan funktionaliteten implementerades bestämdes det var, samt hur, det skulle grafiskt representeras. Efter funktionalitet implementerades medkom flera omgångar avlusande då utbyggandet av funktionalitet kunde påverka stabiliteten hos någon av de redan implementerade delarna. När mängden funktionalitet som fortfarande behövde implementeras minskade skiftades fokuset alltmer till refaktorering av den redan skrivna kodén. I samband med detta ställdes nya krav på kodkvalitet för att minska framtida refaktoreringsbehov. Det som krävde mest arbete i detta skedet var omstruktureringen av kod mellan de olika filerna och paketen i avseende att så väl som möjligt följa MVC-principen.

### 3.2 Genomförande av projektimplementationen

Den största delen i projektets process var den kodmässiga implementeringen. Till hjälp för att administrativt kunna hantera ett projekt där flera medarbetare ingick användes versionshanteringsmjukvaran Git. Detta för att se till att det inte skulle finnas några kodkonflikter då det var möjligt för flera individer att självständigt jobba med samma del kod. Till hjälp för planering och arbetsdelegering användes Git Issues. Med Git Issues var det möjligt att skapa små trådar om funktionalitet som saknades respektive ej fungerade korrekt.

## 4 Resultat

Resultatet av detta projekt är mobilapplikationen Studentbostäder Göteborg som kan användas av studenter för att söka bostäder i Göteborg.

### 4.1 Implementation av applikationen

Nedan följer applikationens fyra viktigaste implementationsdelar.

#### 4.1.1 Hämtandet av bostadsdata

Varken SGS eller Chalmers studentbostäder erbjuder öppna API:er vilket försvårade hämtningen av data, ett API är en mall för hur kommunikation sker med en given kodbas. Android som utvecklingsmiljö försvårade ytterligare hämtningsprocessen från SGS, då Android hade vissa krav angående hur data skall hämtas från internet.

SGS svarade på databegäran med ett JSON-objekt som enkelt kunde omvandas till ett Java-objekt med hjälp av biblioteket GSON. Chalmers studentbostäder svarade däremot med en textformaterad hemsida. För att tolka, formatera och extrahera den relevanta information från detta textformat byggdes det flera icke återanvändbara metoder.

#### 4.1.2 Lagring av objekt med hjälp av Db4o

Processen att implementera databasen Db4o var smidig men innehöll utmaningar. Med hjälp av internetguider kunde egengjorda bostadsobjekt snabbt sparas och laddas mellan applikationens sessioner. Sparande av tidstämplat ställde dock till med problem, tidstämplarna återställdes när dem lästes in. En tidstämpel i detta sammanhang är en siffra på antalet millisekunder som har passerat sedan den första januari 1970. Då Db4o inte längre utvecklas var felsökningsprocessen svår.

Slutligen blev det en sämre lösning där tidstämpeln sparas i en strängvariabel i en egengjord klass. Detta då Db4o inte tillåter att använda vissa andra mer lämpade variabeltyper.

#### 4.1.3 Implementering av Model-View-Controller

Då Androidkodning innefattar bland annat Activities (*aktiviteter*), leder detta till ett försvårande av MVC-implementeringen. Dessa aktivitetsfiler är ett lager som innehåller kod som borde vara i både vypaketet samt i kontrollpaketet. I denna applikation valdes, med vägledning av handledare, att placera alla aktivitetsfiler i ett eget aktivitetspaket. Förutom detta byggdes Studentbostäder Göteborg som ett MVC-projekt med ett tydligt paket till var och en av de respektive delarna. Så mycket som möjligt från alla aktiviteter bröts ut och placerades i kontroller- respektive vypaketet.

#### **4.1.4 Androids påverkan på projektet**

Android har ett starkt stöd för de bakgrundsaktiviteter applikationen använde sig av. Notifikationer såväl som automatiska uppdateringar behövde inga externa bibliotek och det krävde endast ett fåtal rader att implementera.

Androids egna säkerhetsregler komplicerade utvecklingen av applikationen. Detta eftersom reglerna krävde att nätverksaktiviteter trådades, vilket är när mobilen kör olika delar av programmet samtidigt. Kombinationen av detta och att modifiering av vyer inte tillät trådning, försvarade kraftigt implementationen av vyer som dynamiskt uppdaterades med ny data från nätet.

### **4.2 Användarvänlighet**

Det var viktigt att Studentbostäder Göteborg fick god användarvänlighet eftersom de existerande sidorna för bostadssökning har bristande användarvänlighet var en stor motivation till projektet. Däremot ansågs kraven inte vara alldeles för höga eftersom applikationens målgrupp är teknikmedvetna studenter med god mobilvana. Applikationen använder sig på grund utav detta några ganska avancerade funktioner ur ett användarperspektiv.

#### **4.2.1 Applikationens responsivitet**

Genom att låta bakgrundsaktiviteter kontinuerligt uppdatera applikationens data och bilder behövde inte applikation vid uppstart vänta på de långsamma nätverksprocesser som en uppdatering innehåller. Det möjliggör att applikation kunde startas med nyutkommen data inom en sekund.

Att navigera mellan applikationens olika sidor skedde på ett ineffektivt sätt, istället för att återanvända vyerna skapades det en ny varje gång. Den extra tiden som det tog för denna processen förhöll sig dock med god marginal inom den tidsram som är acceptabel för navigation [9], se tabell 1 för exakta navigationstider.

#### **4.2.2 Applikationens funktionalitet**

Huvudfunktionaliteten för Studentbostäder Göteborg var möjligheten till bevakningar av nya bostäder. En användare skulle enkelt kunna spara vissa sökkriterier som applikationen sedan automatiskt använde för att regelbundet jämföra med bostäder från SGS samt Chalmers Studentbostäders egna hemsidor. När applikationen, med hjälp av de förbestämda sökkriterierna, fick en träff skickades en notifikation till användaren. Dock funkade inte bevaknings funktionen felfritt, många gånger skickades ingen notis samt det saknades stöd i applikationen för att visa vilka nya bostäder som dykt upp i bevakningen.

Utöver bevakningar fanns det även funktionalitet för att bläddra genom bostäder på ett smidigt sätt, genom att användaren svepte med fingret över skärmen för att få upp nästa bostad.

En viktig grundfunktionalitet var att applikationen slår samman SGS och Chalmers Studentbostäders bostäder till samma plattform. Tanken var att erbjuda möjligheten att söka bland bostadsdatabasen med fler sökkriterier än hyresvärdars egna implementationer tillät, detta hann vi tyvärr inte med utan enbart några stycken sökkriterier är implementerade. En funktion som ändå blev fullt färdig och som inte fanns hos bostadssökarsidorna var möjligheten att smidigt spara en bostad i en favoritlista.

#### 4.2.3 Målgruppsanpassning

Högskolestuderande personer är Studentbostäder Göteborgs enda målgrupp. Då hälften av alla högskolenyörjare är under 21 år [10] betyder det att målgruppen är unga vuxna. Under utvecklingen av Studentbostäder Göteborg hölls det i åtanke att målgruppen unga vuxna är datorvana och därför både bjuder in till samt kräver vissa mer avancerade funktioner. Personer med god mobilapplikationserfarenhet är vana vid viss funktionalitet, så som exempelvis att man kan svepa ett objekt i en lista åt höger eller vänster för att ta bort det respektive spara det i en favoritlista, se figur 4.

#### 4.2.4 Kvarstående programfel

Då projektets produkt var en mobilapplikation låg fokuset ej på att lösa de diverse programfel som enbart förekommer under emuleringskörning. Det vill säga körning i ett program där man virtuellt använder sig av en androidtelefons operativsystem, på en traditionell persondator. Dessa programfel inkluderar fel i datainsamling, uppdatering, och navigation.

Det finns dessutom några programfel kvar även när applikationen körs på mobiler, de största återfinns vid navigering mellan olika aktiviteter. En utav dem är ifall man använder mobilens inbyggda bakåtknapp då hänger navigationsfältet längst ner inte med utan tror man är kvar på samma sida.

## 5 Diskussion

De tre avsnitten nedan innehåller diskussion om hur arbetet genomfördes, hur resultatet blev och vad som kunde ha förbättrats.

### 5.1 Arbetsmetodsdiskussion

Att tidigt och tydligt definiera vad applikationen skulle innehålla och bestämma den grundläggande arkitekturen för programmet var till stort stöd. Det bidrog dels med att tydliggöra vad som var kvar att göra vilket gjorde att arbetsuppgifterna blev klarare, samt så bidrog det med att skapa en enkel struktur där onödiga och komplexa komponenter inte utvecklades förrän det behövdes. Ett exempel på det är hur domänmodellen utvecklades mellan första, andra, och sista iterationen, se figurer 1, 2, 3.

Vidare visade det sig att det var komplicerat att genomföra refaktorering samtidigt som implementering av ny funktionalitet då det ställde höga krav på kommunikationen. De gånger kommunikationen brast riskerades det att utvecklingen hade gått åt olika håll och ändringarna blev inkompatibla med varandra, då behövdes koden refaktoreras igen.

En slutsats som dras från projektet är att den initiala fasen där applikationens arkitektur och innehåll bestämdes var väl värt det då den förenklade utvecklandet.

En ytterligare slutsats är att det antagligen skulle varit hjälpsamt att utveckla den iterativa implementeringsprocessen genom att lägga till refaktorering på slutet av processen istället för att genomföra refaktoreringen parallellt med implementeringen.

### 5.2 Resultatdiskussion om implementationen

I den här sektionen diskuteras de mer tekniska delarna i projektet samt de beslut som låg bakom dem.

#### 5.2.1 Val av databas

Med facit i hand var valet av Db4o inte det bästa. Det hade lönat sig att välja någon databas som fortfarande utveckling dels av underhållsskäl samt eftersom det antagligen skulle förenklat felsökningar då mer hjälp generellt erbjuds.

#### 5.2.2 Hämtandet av data

Hämtandet av data bjöd också på en del utmaningar, då det inte var så lätt att få ut all information som det verkade från början. Dock var det betydligt enklare att hämta data från SGS då det skickade tillbaka ett JSON-objekt, något som det existerar färdiga verktyg för. Utöver formaten var datan inte densamma hos de två hyresvärdarna, så för att få en konsekvent upplevelse behövdes en del data göras om eller helt enkelt inte användas. Vilket tog mycket tid.

### **5.2.3 Model-View-Controller implementering**

När det kommer till kodstruktureringen var det MVC som tog mest tid, framförallt på grund av att aktiviteterna sågs som kontrollers i början av projektet. Detta ledde till stora krav på refaktorering. Men till slut följer Studentbostäder Göteborg MVC designmönstret så gott det går för en Android applikation.

Med bättre förkunskaper hade mycket tid kunnat sparas, genom att mer tid lagts på att definiera en tydlig kodstruktur innan man börjar implementera funktioner.

## **5.3 Resultatdiskussion angående användarvänlighet**

Användarvänlighet var extra viktigt i projektet eftersom det sågs som en stor del till att uppfylla applikationens syfte.

### **5.3.1 Diskussion kring funktionalitet**

En av de stora anledningarna till att applikationen utvecklades var att det fanns funktionalitet som saknades på den ena eller den andra hyresvärdens sida. En av applikationens största, ur ett marknadsföringsperspektiv, säljpunkter var möjligheten att ha en sökning som skickar en notis till dig när en bostad som uppfyller de krav du ställt genom bevakningar. Denna funktionalitet blev inte riktigt färdig till den grad som önskats för att vara fullt användbar.

Möjligheten att söka och filtrera med fler termer än de existerande sidorna var en annan viktig funktionalitet som inte hann bli implementerad fullt ut.

Men i övrigt är det mesta av funktionaliteten på plats och mestadels fungerande. Med tanke på att en del saker tog betydligt längre tid än förväntat hann inte funktionaliteten utvecklas till den grad som hade önskats från början. För att applikationen ska vara ett klart bättre alternativ till bostadsbolagens hemsidor behövs lite mer utvecklad funktionalitet.

### **5.3.2 Målgruppsanpassning**

Målgruppen underlättade utvecklandet av gränssnittet, då man kan ta en hel del saker för givet. Men den kom också med en del krav, bland annat på responsivitet och enkelhet. Då utförliga användartester krävs för att bestämma en applikations användarvänlighet, vilket det inte fanns tid till, lades fokuset på responsiviteten. Med mätningarna som utgångspunkt anses kravet på responsivitet uppfyllt, medans kravet på enkelhet är betydligt svårare att bekräfta. Användartester med målgruppen är något som med mer tid borde ha genomförts för att faktiskt se hur gränssnittet hade uppfattats.

### **5.3.3 Kvarstående programfel**

Då applikationen ej är utvecklad i avseende att användas genom emuleringskörning efter projektet är avslutat behöver dessa ej avlusas då användare förväntas att

aldrig komma i kontakt med dem. Programfel är något som egentligen skall undvikas, men på grund av att flertalet av dessa inte dyker upp på en riktig telefon i kombination med en viss tidsbrist togs beslutet att bortprioritera dem. Dessutom ansågs flera av de fel som kvarstod inte vara allvarliga nog att förhindra användandet av applikationen.

#### 5.3.4 Sammanfattning av användarvänlighet

Användarvänlighet är den viktigaste aspekten i applikation då det redan finns andra plattformar som erbjuder den viktigaste funktionaliteten för att söka studentbostäder. Med det i åtanke hade funktionalitet i applikationen behövt fungera bättre, bevakningar hade behövt vara helt implementerat samt hade Google Maps funktionaliteten, nämnd i 5.4, gjort applikationen betydligt attraktivare då det hade gett Studentbostäder Göteborg unika samt attraktiva funktioner.

Då applikationen i dagsläget saknar en del av den funktionaliteten som var planerad att riktigt separera den från originalsidorna blir användbarheten låg och många användare skulle troligtvis använda någon av de redan befintliga hemsidorna. Men med lite mer tid lagd på att styrka användarvänligheten genom att utveckla funktionalitet skulle applikationen vara en klar förbättring mot befintliga bostadstjänster.

### 5.4 Möjliga förbättringar

En funktion som hade förbättrat bostadssökandet i Studentbostäder Göteborg betydligt är möjligheten att söka med hjälp av en karta eller reseavstånd till en viss punkt. Denna implementation skulle då ske med hjälp av Google Maps och dess API. Sökresultaten skulle då visas på en karta istället för i en lista.

Något som ej genomfördes på grund av tidsbrist var möjligheten att se de senaste bostadssökningarna som har genomförts och genom att klicka på dem göra om deras sökning direkt utan att behöva fylla i alla sökfilter igen.

En annan sak som från början bestämdes borde finnas men som aldrig implementerades var möjligheten för en bostad att ha flera bilder associerade med sig. I nuläget finns bara en bild till varje bostad. Därtill saknas även planritningen till en bostad.

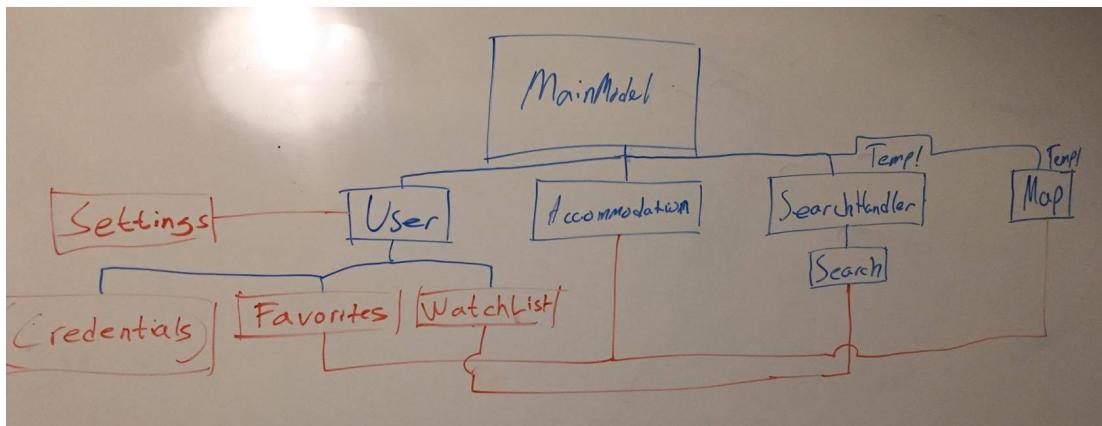
En annan funktionalitet som hade kunnat vara bättre är ansökan till lägenheter, applikationen skickar användaren till hemsidan där annonsen ligger uppe. Då hela poängen med applikationen är att slippa använda dessa hemsidor är det inte optimalt.

En ytterligare del som hade behövts göras bättre för att applikationen ska kunna ses som ett alternativ till de båda hemsidorna är att visa all information. I dagsläget finns det mycket information som inte visas, trots att den i många fall finns i applikationens bakomliggande system.

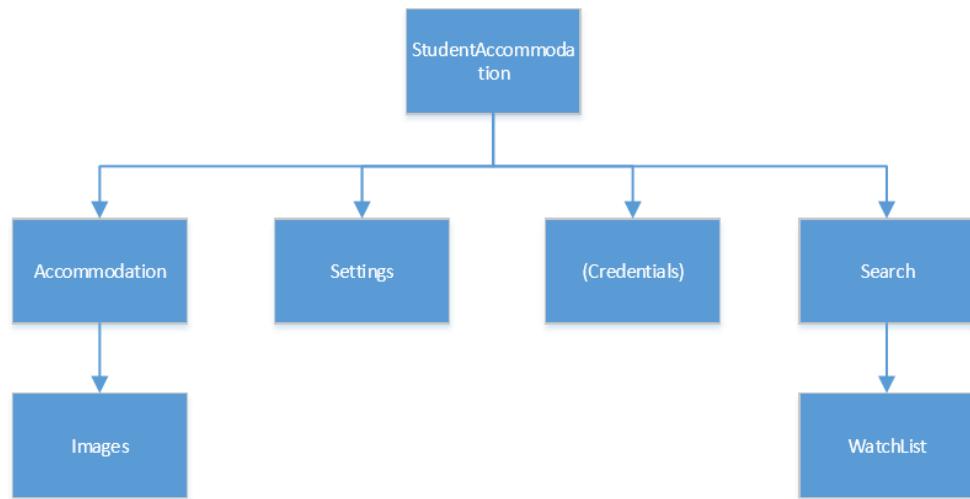
## Källförteckning

- [1] J. Hellekant, “Nya krissiffror: Bostadskön går baklänges,” 2017. [Online]. Tillgänglig: <https://www.svd.se/nya-krissiffror-bostadskon-gar-baklanges>. Hämtad: 29 maj, 2017.
- [2] A. Löwendahl, “Nu är bostadskön som längst för göteborgs studenter,” 2013. [Online]. Tillgänglig: <https://www.hemhyra.se/nyheter/nu-ar-bostadskon-som-langst-for-goteborgs-studenter/>. Hämtad: 29 maj, 2017.
- [3] Chalmers Studentbostäder, “Our rental rules,” 2016. [Online]. Tillgänglig: <https://www.chalmersstudentbostader.se/en/soka-bostad/>. Hämtad: 29 maj, 2017.
- [4] B. Sokhan, “Domain driven design for services architecture,” 2015. [Online]. Tillgänglig: <https://www.thoughtworks.com/insights/blog/domain-driven-design-services-architecture>. Hämtad: 29 maj, 2017.
- [5] E. Evans, *Domain-driven Design: Tackling Complexity in the Heart of Software*, 1 uppl. Westford, MA, USA: Addison-Wesley, 2003. [Online]. Tillgänglig: <http://www.amazon.com>.
- [6] actian, “We are restructuring our versant community website.” [Online]. Tillgänglig: <http://supportservices.actian.com/versant/default.htm>. Hämtad: 29 maj, 2017.
- [7] D. Skrien, *Object-Oriented Design Using Java*, 1 uppl. New York, NY, USA: McGraw-Hill, 2009, hämtad: 29 maj, 2017. [Online]. Tillgänglig: <http://www.allitebooks.com>.
- [8] R. J. E. Gamma, R. Helm, *Design Patterns: Elements of Reusable Object-Oriented*, 1 uppl. Boston, Massachusetts, USA: Addison-Wesley Professional, 1994. [Online]. Tillgänglig: <http://www.amazon.com>.
- [9] M. Kearney, “Measure performance with the rail model,” 2017. [Online]. Tillgänglig: <https://developers.google.com/web/fundamentals/performance/rail>. Hämtad: 29 maj, 2017.
- [10] S. centralbyrån, “Universitet och högskolor: Studenter och examineerade på grundnivå och avancerad nivå 2014/15,” 2015. [Online]. Tillgänglig: [http://www.scb.se/Statistik/UF/UF0205/2014L15E/UF0205\\_2014L15E\\_SM\\_UF20SM1601.pdf](http://www.scb.se/Statistik/UF/UF0205/2014L15E/UF0205_2014L15E_SM_UF20SM1601.pdf). Hämtad: 29 maj, 2017.

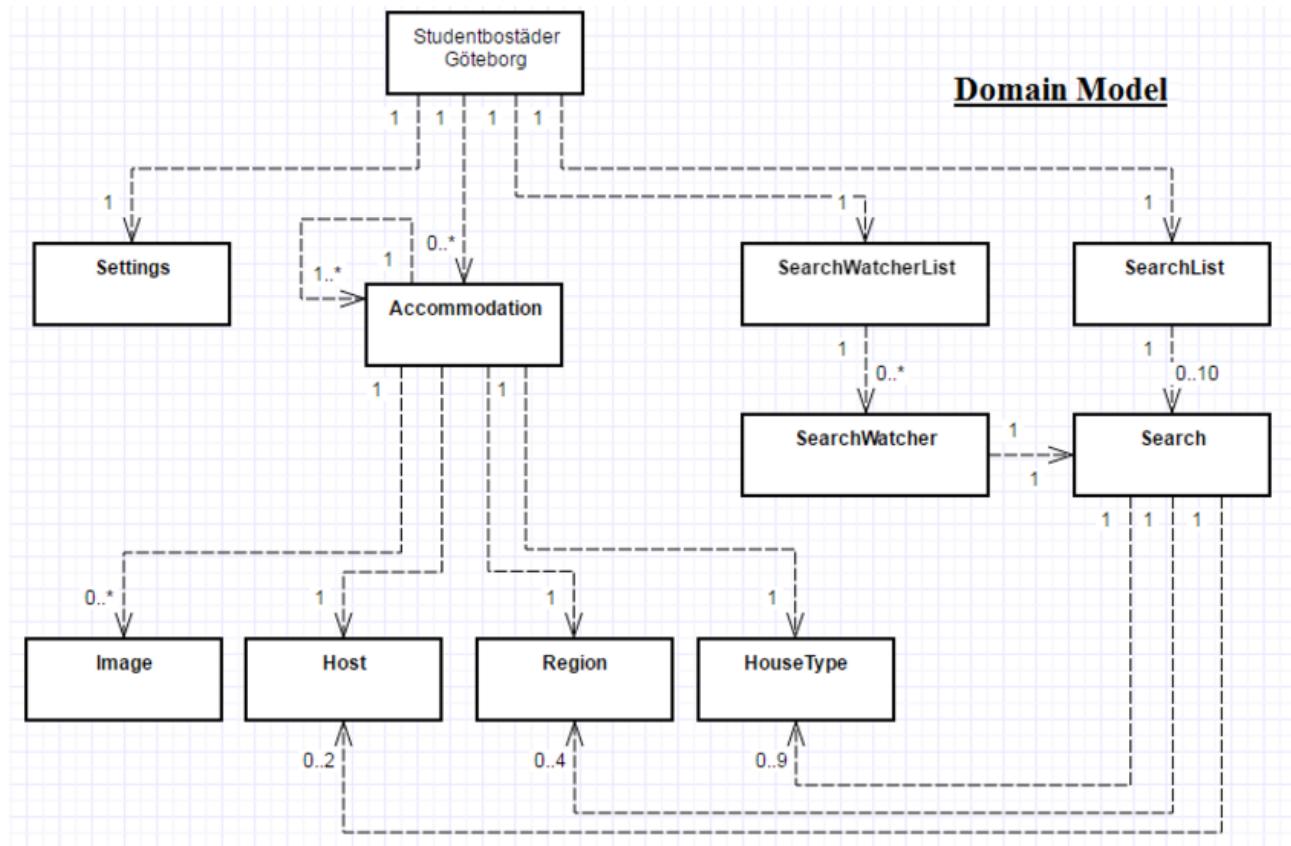
## Bilagor



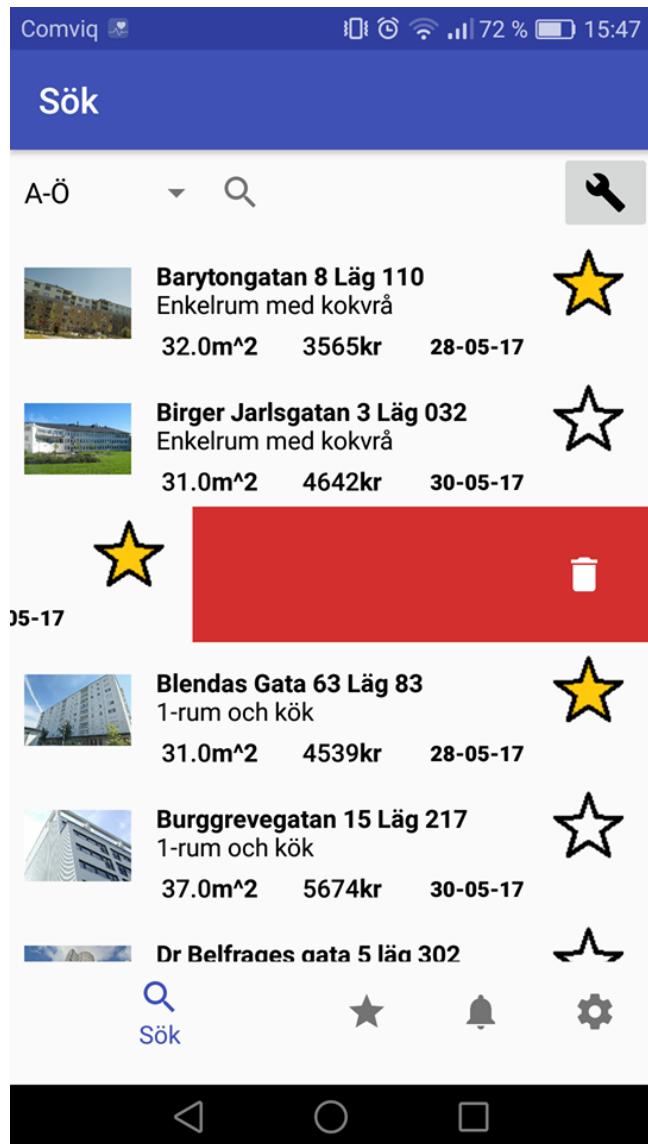
Figur 1: Första iterationen



Figur 2: Andra iterationen



Figur 3: Implementerade iterationen



Figur 4: En vänstersvept bostad

	Försök 1	Försök 2	Försök 3	Genomsnitt
Sök	141	126	122	130
Favoriter	153	112	101	122
Bevakningar	128	107	103	113
Inställningar	163	163	120	148

Tabell 1: Navigationstid mellan vyer i milisekunder