

## **SmartElement**

Automatiserad anpassning av webbinnehåll

Staffan Enberg

# INNEHÅLL

<b>1</b>	<b>Inledning</b>	<b>5</b>
1.1	Målsättning och syfte	5
1.2	Metoder	5
1.3	Avgränsningar	5
<b>2</b>	<b>Bakgrund</b>	<b>6</b>
2.1	Anpassat innehåll	6
2.2	Behov	7
2.3	Befintliga lösningar	7
<b>3</b>	<b>SmartElement</b>	<b>9</b>
3.1	Beståndsdelar	9
3.1.1	<i>Webbsidan</i>	9
3.1.2	<i>Element</i>	10
3.1.3	<i>Filter</i>	10
3.1.4	<i>Innehåll</i>	10
3.2	Funktion	10
3.2.1	<i>Konfiguration</i>	11
3.2.2	<i>Sidvisning</i>	11
<b>4</b>	<b>Teknisk arkitektur</b>	<b>13</b>
4.1	PHP	13
4.2	Databas	14
4.2.1	<i>MySQL</i>	14
4.2.2	<i>MongoDB</i>	14
4.3	Memcache	15
4.4	JavaScript	16
<b>5</b>	<b>Processer</b>	<b>17</b>
5.1	Information om användaren	17
5.1.1	<i>Hänvisningsinformation</i>	17
5.1.2	<i>IP Geolokalisering</i>	18
5.1.3	<i>Besökarstatistik</i>	18
5.1.4	<i>Tid</i>	18
5.1.5	<i>Användardefinierad information</i>	18
5.2	Filtrering	18
<b>6</b>	<b>Arkitektur</b>	<b>19</b>
6.1	Back-End system	19
6.1.1	<i>Matcher</i>	19
6.1.2	<i>Site objektet</i>	19
6.1.3	<i>Element objekten</i>	19

6.1.4	<i>Filter objekten</i>	19
6.1.5	<i>Content objekten</i>	20
6.2	Javascript tagen	20
<b>7</b>	<b>Sammanfattning och Diskussion</b>	<b>21</b>
<b>8</b>	<b>För uppgiften Utvidgad disposition</b>	<b>21</b>
<b>Källor</b>		<b>23</b>



# 1 INLEDNING

Detta arbete dokumenterar systemet SmartElement som utvecklades för företaget Developer's Helsinki Ab. Systemets syfte är att möjliggöra enkel konfiguration av anpassat innehåll för webbsidor, vilket tillåter webbsidors ägare att själva kunna konfigurera regler som bestämmer vad för innehåll som skall lyftas fram för olika användare.

## 1.1 Målsättning och syfte

Målsättningen med arbetet är att kartlägga den arkitektur som bäst stöder målet av ett effektivt system för innehålls filtrering och leverering, som även är lätt att vidare utveckla och utvidga i form av nya filter och nya algoritmer för filtrering.

## 1.2 Metoder

Systemet är i sig bara en motor för identifikation av användare och filtrering av innehåll. Det ända gränssnitt som SmartElement har är ett JSON-gränssnitt genom vilket ett skilt grafiskt gränssnitt tillåter användare att konfigurera sitt innehåll.

## 1.3 Avgränsningar

Detta är en redogörelse för den tekniska implementeringen inte användningen av anpassat innehåll på webbplatser, arbetet tar därmed inte ställning till hur anpassat innehåll skall användas, endast hur det kan implementeras.

Även om en av målsättningarna är att skapa ett effektivt system så finns det ingen klar standard för vad som kan anses vara bra prestanda för ett liknande system, därav kommer arbetet inte att innehålla grundliga prestanda analyser.

## 2 BAKGRUND

Projektet började våren 2013 som ett internt verktyg på Developer's Helsinki efter ett behov uppstått för ett system som skulle tillåta snabb implementering av personligt innehåll på kunders webbsidor.

Eftersom det i den tiden inte fanns någon färdig implementation att använda, bestämdes det att det skulle utvecklas en mjukvara först för internt bruk men med målsättningen att lanseras som en SaaS produkt.

### 2.1 Anpassat innehåll

Your Amazon.co.uk

Computers & Accessories



New Release  
Kensington Orbit ...  
★★★★☆ (202)  
£18.99 **£17.50**  
Why recommended?



PCSL / Adafuit Clear ...  
★★★★☆ (19)  
£3.49  
Why recommended?



Clear Transparent top ...  
★★★★☆ (55)  
£12.99 **£4.80**  
Why recommended?



Raspberry Pi Camera ...  
★★★★☆ (12)  
£24.99 **£19.00**  
Why recommended?

› See all recommendations in Computers & Accessories

Books



New Release  
Don't Make Me Think: ...  
Steve Krug  
★★★★☆ (4)  
£22.99 **£20.99**  
Why recommended?



Clean Code: A ...  
Robert C. Martin  
★★★★☆ (36)  
£31.99 **£23.99**  
Why recommended?



Java Concurrency in ...  
Brian Goetz  
★★★★☆ (20)  
£38.99 **£29.24**  
Why recommended?



Growing ...  
Steve Freeman  
★★★★☆ (22)  
£36.99 **£27.74**  
Why recommended?

› See all recommendations in Books

Figur 1. Amazons användarsida med anpassat innehåll

Anpassat innehåll, eller personligt innehåll (eng. *Personalized content*), är innehåll som väljs ut på basis av egenskaper hos användaren. I fallet av webbsidor kan det hand-

la om information som användarens geografiska läge, användarens språk, användarens webbläsare, antalet besök som användaren gjort till sidan, med mera. Tanken är att förse användaren med det den behöver eller vill ha utan att denne behöver be om det. (Mulvenna et al. 2000)

Anpassat innehåll har bland annat användts inom nätbutiker för att visa reklam anpassad för kunden på basis av dennes beställningshistorik, bilden 1 visar användarsidan som visas då man loggar in på amazons webb-butik. Inom social media har används anpassat innehåll för att lyfta fram innehåll som antas vara intressant för användaren (Uppdateringar från vänner som användaren ofta är i kontakt med, reklam från företag användaren har gillat m.m.). (Guy et al. 2010)

## **2.2 Behov**

I takt med att stora aktörer införde allt mer anpassat innehåll så började även mindre kunder fråga om möjligheten att använda tekniken på sina webbsidor. Det visade sig att det inte fanns något företag på finska marknaden som erbjöd ett tillräckligt flexibelt system för anpassning av innehåll och därmed bestämdes det att Developer's Helsinki skulle utveckla ett sådant.

Systemet skulle kunna leverera innehåll dels i form av text (HTML eller rå text) för direkt injicering i webbsidans Document Object Model (DOM), dels i form av data i JavaScript Object Notation (JSON) format och dels som Json-with-padding (JSONP) svar vilket tillåter exekvering av javascript vid svaret. För filtrering skulle det samlas in data om beökares webbläsare och dator men det skulle även vara möjligt att tillägga egen data som sedan kunde användas för filtrering.

## **2.3 Befintliga lösningar**

Vid tillfället för beslutet att utveckla produkten var den enda liknande lösningen på finska marknaden nosto.com som levererar personaliserat innehåll med inriktning på webbuti-

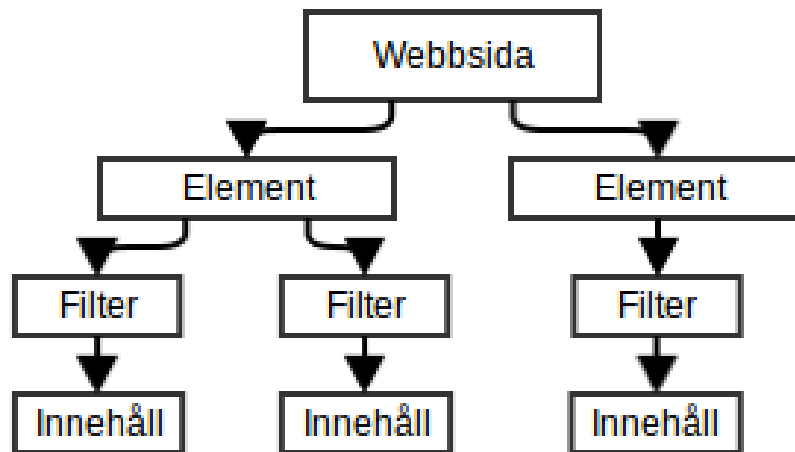
ker.(Nosto Website 2014) På grund av produktens klara vinkling passade den inte in i alla de användningsområden var SmartElement skulle användas.



### 3 SMARTELEMENT

SmartElement är ett back-end system som sköter filtrering av innehåll på basis av information som samlas in om besökare på en webbsida. Upprätthållare av en webbsida kan genom att integrera SmartElements JavaScript tag i sin sida, låta back-end systemet fylla i specificerade element med personligt innehåll.

#### 3.1 Beståndsdelar



Figur 2. Exempel på en hierarki i SmartElement

SmartElement bygger på några grundläggande koncept; webbsidor, element, filter och innehåll. Med dessa fyra byggstenar konfigureras det objekt, visualiserat i bild 2, som back-end systemet använder för att filtrera och leverera innehåll.

##### 3.1.1 Webbsidan

Webbsidan är det högsta elementet i hierarkin som SmartElement handskas med. Det representerar en hel webbplats, under vilken man kan definiera element med innehåll. Varje sida har en unik identifikationsnummer som används för att ladda SmartElement

tagen.

### **3.1.2 Element**

Elementen är hållare för den data som returneras från back-end systemet och det som knyter innehållet till webbsidan. Elementet har i sig flere innehåll och filter, filtren används för att välja ut det innehåll som skall visas i elementet. Element har en inom sidan unik kod som används för att specificera vilka element som behöver processeras när man kallar på back-end systemet.

### **3.1.3 Filter**

Filter består i SmartElement av flere olika regler som ställer en fråga om besökaren samt en länk till ett innehåll, vilket visas om filtret matchar. För att ett filter skall matcha krävs det att besökaren passar alla regler i filtret. För att hantera fall var två eller flere filter passar för en besökare har filter en prioritetsordning inom elementet, valbar av användaren.

### **3.1.4 Innehåll**

Innehåll är den konkreta data som levereras för ett element efter att ett filter valts som vinnare. Innehållet kan bestå av text, HTML kod eller JSON data, vad som helst för data som kan sparas som en textsträng, det är upp till användaren att definiera vad för data som sparas.

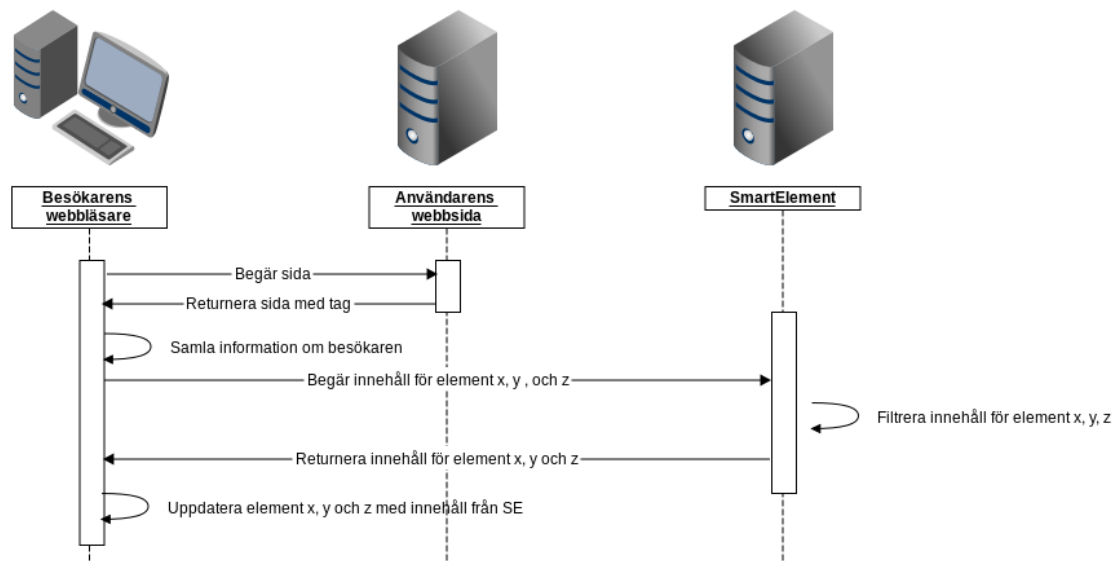
## **3.2 Funktion**

SmartElement har två huvudsakliga funktionsområden, konfiguration av sidor och hantering av sidvisningar.

### 3.2.1 Konfiguration

För att SmartElement skall kunna leverera innehåll, måste användaren konfigurera sidan och dens element på förhand. Processen består av att registrera element under sidobjektet, skapa innehåll samt att konfigurera filter genom att kombinera regler om vilka användare innehållet passar för.

### 3.2.2 Sidvisning



Figur 3. Processen vid en sidvisning

Bild 3 representerar kommunikationen vid en sidvisning. När en användares webbsida laddas, inkluderar den en JavaScript fil som innehåller SmartElements tag. Efter att sidan laddats klar körs tagen, som samlar ihop information om besökaren samt eventuell extra data som användaren specificerat i tagen, och skickar informationen till back-enden.

Backenden tar emot data från tagen och söker upp sidan som motsvarar det id som skickats. Efter att en sida har hittats går back-enden igenom de element som tagen begär innehåll för och filtrerar deras innehåll på basis av den data som tagen skickat. Allt innehåll kompileras till en lista med element-id länkat till innehåll och skickas tillbaka till besökarens webbläsare.

Om användaren inte valt att ändra på beteendet av tagen så kallar backenden på tagen för

att fylla elementen med den data som returnerats.

## 4 TEKNISK ARKITEKTUR

SmartElement bygger i stort sett på fem centrala tekniker, programmeringsspråket PHP som användts för att skriva koden bakom systemet, MySQL som databaslager för avändar och systemdata, Memcache som caching lager vilket tillåter att information som är tung att beräkna kan sparas mellan förfrågningar, dokument databasen MongoDB för lagring av information om besökare och statistik, och till sist JavaScript som används för att skapa det informationspaket om användaren som skickas till servern för användning vid filtreringen av innehåll.

Många av de val som gjorts i planeringen av den tekniska arkitekturen bakom SmartElement baserar sig på en möjlighet att kunna skala systemet för att hantera förändrade användarmängder. De grundläggande principerna har varit att det skall vara lätt att utveckla, det skall vara lätt att underhålla, det skall vara lätt att installera och det skall vara lätt att skala upp om det skulle behövas.

### 4.1 PHP

PHP är ett skriptspråk som utvecklats med målet att snabbt kunna skapa dynamiska webbsidor. (PHP Manual 2013) Språket valdes för att det är enkelt att arbeta med och det var språket som företagets andra produkter var skrivna med.

PHP är ett av de mest populära språken för webbutveckling (TIOBE Programming community index 2014) och det finns en mängd information och färdiga kodpaket till förfogande för programmerare, vilket underlättar och försnabbar utvecklingen. Genom sin popularitet har språket även den fördelen att det är lätt att hitta utvecklare som kan fortsätta utvecklingen av projekt skrivna i PHP.

## 4.2 Databas

Databas lagret inom SmartElement är delat splittrat i dels en relationsdatabas och dels en dokumentdatabas. Denna lösning valdes för att dra nytta av de goda sidorna med båda teknikerna, var vissa data stöds bättre av den stabila datastruktur som en relationsdatabas ger medan annan data drar nytta av flexibiliteten som en dokumentdatabas ger.

Eftersom SmartElement hanterar dels intern data som är starkt bunden till systemet och påverkar funktionaliteten av systemet, och dels extern data som är närmare kopplad till användarens webbsida och påverkar hur en användares sida fungerar så finns det två tydliga data-set som båda har egna krav.

### 4.2.1 MySQL

För lagring av intern data används relationsdatabasen MySQL. Databasen lämpar sig bra för lagring av data med starka lenkar mellan objekten, vilket passar bra in på den användardata som SmartElement behandlar, det finns en klar hierarki bland objekten och länkar mellan dessa.

Relationsdatabasen gör sig bra i denna roll då den ger en viss garanti på data strukturen bakom de olika objekt som bygger upp SmartElement. Genom att binda modellerna till en databasstruktur har man en sorts garanti att existerande data uppdateras och transformeras i samband med att processer i systemet uppdateras, på så vis ansågs en rigidare databasmodell bättre anpassad för denna uppgift.

### 4.2.2 MongoDB

För lagring av extern data, relaterad till besökare av användares webbsidor, valdes dokumentdatabasen MongoDB. MongoDB använder inte ett strikt schema för datalagring utan dokument i samma samling kan ha skiljande attribut associerade med sig, detta underlättar vidareutveckling av systemet genom att tillåta legacy data att existera i databasen tillsammans med ny tills den gamla uppdateras och kompletteras då besökare återvänder

till en användares webbsida.

I fallet av denna data så är den strikta datastrukturen inte lika viktig, datan föråldras dels snabbt samt att designen är sådan att datan inte skall behöva vara komplett och innehålla strikta datastrukturer. Tanken med besökarobjekten är att de skall vara levande och öppna för modifikation och utvidgning, om ett nytt filter läggs till i systemet så skall det inte kräva en migration av legacy data, utan statistiken för det nya filtret läggs helt enkelt till på objektet och om inte datan finns på objektet så matchar filtret ej. Denna flexibilitet är lättare att åstadkomma med en dokumentdatabas eftersom att scheman just inte behöver uppdateras, vilket potentiellt skulle vara en väldigt tung operation i fallet av besökarobjektet som är den överdrivet snabbast växande samlingen inom systemet.

## 4.3 Memcache

Eftersom SmartElements datamodell består av många länkade objekt är det relativt tungt att bygga upp ett sidobjekt från databasen då det måste göras många förfrågningar. På grund av detta samt för att minska risken för (n+1)-problem (var man gör en databasförfrågan för varje element i en samling som man itererar över) så bestämdes det att ett caching lager skulle användas för att hålla dessa datamodeller och på så vis låta mjukvaran gå direkt till filtrerings logiken då en förfrågan från tagen behandlas.

Målet med cache lagret är att hålla sidobjekten i minnet på servern så mycket som möjligt, och endast läsa från databasen när ett objekt byggs om i samband med en uppdatering. För detta ändamål valdes Memcache, en väletablerad mjukvara för caching. Memcache valdes för att det var en teknik som utvecklarna var vana vid, den var redan i bruk i andra projekt samt att dess enkla modell av horisontell skalning skulle tillåta systemet att snabbt reagera på en växande kundskara.

## 4.4 JavaScript

För att kunna identifiera så mycket information som möjligt om besökaren samt för att leverera innehållet som valts ut för beökaren är det nödvändigt att köra kod på klienten. För att stöda detta krav valdes scriptspråket javascript, vilket stöds av så gott som godtyckliga webbläsare och tillåter logik att köra på besökarens maskin i samband med en sidvisning.



## 5 PROCESSER

För att kunna leverera anpassat webbinnehåll till besökaren av en webbsida, måste systemet ha ett sätt att kunna identifiera information om denne som sedan kan användas för att filtrera bland det tillgängliga innehållet.

### 5.1 Information om användaren

SmartElement använder sig till stor del av statistik som systemet samlar ihop med hjälp av en JavaScript tag som körs i besökarens webbläsare då denne besöker en sida som använder systemet. Denna information skickas till back-end systemet som vidare processerar datan och sparar den i en dokument databas under ett beökarobjekt. På detta viset kan systemet komma ihåg användare mellan sessioner och på så vis skapa historik om användarens beteende och vanor, vilket i sin tur är till nytta för att hitta det bästa innehållet att presentera.

#### 5.1.1 Hänvisningsinformation

Den första informationen som registreras om användaren är varifrån denne anländer till sidan. Webbläsare skickar oftast en så kallad referrer variabel när man navigerar till en webbsida genom att klicka på en länk, denna variabel innehåller adressen på sidan som visade länken.

Genom att spara denna information får man dels en indikation av varifrån besökare hittar till sidan, men även en indikation av vilken typs sidor den aktuella besökaren ofta besöker.

En nackdel är att hänvisningsinformation inte alltid finns tillgänglig, och avsaknad av denna information ger en inte ett definitivt svar om hur användaren hittat till sidan. Detta på grund av att hänvisningsinformation inte skickas t.ex. när man navigerar till webbsidan från en sida med krypterad anslutning, om användarens webbläsare har konfigurerats

att inte sända informationen men även om användaren helt enkelt skrivit in webbsidans address i sin webbläsare och är en så kallad direkt träff. (Fielding et al. 1999)

### **5.1.2 IP Geolokalisering**

Nästa bit av information som samlas in om bseökaren är uppskattnng av dennes position på basis av den information som registrerats för IP-adressen som förfrågan kommer från.

SmartElement använder sig av en databas som köps in av ett företag som specialiserar sig i att uppehålla så noggrann information som möjligt om var i världen IP-adresser egentligen är registrerade. Från denna databas söker systemet sedan fram information om besökarens läge. Information som är tillgänglig är bland annat vilket land användaren befinner sig i, vilken region inom landet samt vilken stad.

Trots att informationen i databasen ar av rätt hög kvalitet, kan man inte helt och hållet lite på informationene som genereras genom IP geolokalisering. Dels så är IP-adressen som sänds till servern inte nödvändigtvis besökarens egna IP-adress då denne kan vara uppkopplad via en VPN-anslutning eller eventuellt använda sig av en proxy-server som inte vidare förmedlar ursprungsadressen, detta betyder i sin tur att man får platsen var VPN- eller proxy-serverns IP är registrerad. Ett annat problem är att den information som finns tillgänglig beror på vad besökarens internet leverantör rapporterar, så även om IP-adressen är användarens egna så kan leverantören ha registrerat adressen på en annan plats än den var användaren befinner sig.

### **5.1.3 Besökarstatistik**

För varje sidvisning på en webbsida med SmartElement sparas det statistik. Användarobjektet som sparas i systemet innehåller information om hur ofta besökaren varit på webbplatesen, hur många sidor denne besökt inom webbplatsen samt hur ofta denne sett en specifik sida.

Genom att skapa statistik som är kopplad till användaren kan man generera information om användarens beteende då denne använder webbsidan. Man kan räkna hur ofta denne besöker specifika delar av sidan, om den inte sett en viss sida på länge, om den börjat besöka en specifik del mer eller mindre o.s.v. All denna information kan hjälpa att söka fram information som kan tänkas relevant för besökaren.

De största problemen med statistik är att den är beroende av möjligheten att identifiera besökaren då denne återvänder till webbsidan. Som systemet är byggt för tillfället hänger detta på användningen av kakor som innehåller en unik id-nummer, vilket betyder att systemet tappat användaren så fort denne raderar denna kaka. Det finns sätt att rundgå detta genom

#### **5.1.4 Tid**

Vad är den lokala tiden för sidan, vad är den lokala tiden för användaren, hur stor tidskillnad är det mellan de två, hur länge har användaren varit på sidan.

#### **5.1.5 Användardefinierad information**

Information som webbsidan själv tillhandahåller.

### **5.2 Filtrering**

Hur kan man filtrera på basis av den insamlade informationen?

## **6 ARKITEKTUR**

Förklaring av arkitekturen Arkitekturen designades utgående från några bas idéer, systemet skall vara lätt att utveckla dels i funktionalitet och dels i

### **6.1 Back-End system**

Back-end systemet är det som gör den egentliga processeringen av information och väljer ut innehållet som levereras.

#### **6.1.1 Matcher**

Matchern är den kod som tar emot ett Visitor objekt som innehåller information om användaren och kör sedan igenom filtren för sidan med detta objekt.

#### **6.1.2 Site objektet**

Objektet som innehåller all information om webbsidan, elementen, filtren och innehållen.

#### **6.1.3 Element objekten**

Element objekten motsvarar ett HTML element på webbsidan, dessa har filter med vilka det kan definiera vilket innehåll det skall visa.

#### **6.1.4 Filter objekten**

Filter objekten innehåller regler som definierar en målgrupp och ett innehåll som det returnerar.

### **6.1.5 Content objekten**

Content objekten är innehåll som returneras, detta kan vara dels HTML innehåll eller javascript kod.

## **6.2 Javascript tagen**

Javascript tagen är en kort kod som laddas i samband med sidan och samlar ihop information om användaren och sedan skickar denna information till back-end systemet för processering.

## 7 SAMMANFATTNING OCH DISKUSSION

Sammanfattning av arbetet, tillbakatitt på produkten. Diskussion om förbättringar som skulle kunna göras i systemet. Diskussion om tekniker som kunde bytas ut för att eventuellt förbättra prestanda. Diskussion om svårigheter med produkten.

## 8 FÖR UPPGIFTEN UTVIDGAD DISPOSITION

Istället för att skapa ett skilt dokument valde jag att skriva använda mig av dokumentet som kommer att bli mitt slutarbete. Här kan man se en grund för arbetet och ett preliminärt skelett för dokumentet.

Jag har påbörjat arbetet med att skriva en kort bakgrund om projektet samt en kort introduktion av systemet som hoppeligen kommer att hjälpa läsaren förstå olika val som kommer att förklaras i arkitektur delen. Jag har även försökt skriva ned några korta meningar under alla rubriker för att ge en bild av det tänkta innehållet.

Källförteckningen är kanske lite kort men den innehåller två relativt tunga verk, den så kallade Gang of Four boken (Design Patterns: Elements of Reusable Object-Oriented Software) och Introduction to algorithms, vilka i samband med The Pragmatic Programmer och High Performance MySQL ganska långt täcker arkitekturen av projektet. Referenserna sträcker sig inte så långt ännu pga de skrivna kapitlens natur, bakgrunden har någon referens till tidsskrifter som behandlat personalisering. Jag skulle dock säga att process och arkitektur kapitlen kommer vara de som behöver mest referenser, eftersom det är där jag gjort ställningstaganden och val som kan behöva stödas. Stilen är Harvard.

Av källorna jag valt är Design Patterns och Introduction to Algorithms båda rätt kompletta verk inom respektive fält så de kan anses vara rätt trovärdiga. The pragmatic programmer är en bok som titt som tätt rekommenderas inom mjukvarubranschen, den innehåller många beprövade metoder för mjukvaruutveckling som är i bruk världen över. Utöver att söka böcker så har jag använt mig av Association for Computing Machinerys söktjänst för att söka information samt försökt använda nelliportalen, ACM kan anses vara ganska

trovärdig källa som en av världens äldsta föreningar för datavetenskap. Den information som tagits från webben är närmast profilering av produkter, enligt deras egen utsago. Informationen är ju inte nödvändigtvis referentgranskad, men den används ej heller i sådan utsträckning var det skulle behövas.

## KÄLLOR

- Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. & Stein, Clifford. 1990, *Introduction to Algorithms*, Massachusetts: MIT press.
- Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P. & Berners-Lee, T. 1999, *RFC 2616, Hypertext Transfer Protocol – HTTP/1.1*. Tillgänglig: <http://www.rfc.net/rfc2616.html>.
- Gamma, Erich; Helm, Richard; Johnson, Ralph & Vlissides, John. 1994, *Design Patterns: Elements of Reusable Object-Oriented Software*, Massachusetts: Addison-Wesley Professional.
- Guy, Ido; Zwerdling, Naama; Ronen, Inbal; Carmel, David & Uziel, Erel. 2010, Social Media Recommendation based on People and Tags, I: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, New York: ACM, s. 194–201.
- Hunt, Andrew & Thomas, David. 1999, *The Pragmatic Programmer*, Massachusetts: Addison-Wesley Professional.
- Mulvenna, Maurice D.; Anand, Sarabjot S. & Büchner, Alex G. 2000, Personalization on the Net using Web mining: introduction, *Communications of the ACM*, årg. 43, , s. 123–125.
- 2014, *Nosto Website*. Tillgänglig: <http://www.nosto.com>, Hämtad: 23.3.2014.
- 2013, *PHP Manual*. Tillgänglig: <http://fi2.php.net/manual/en/>, Hämtad: 2.3.2014.
- Schwartz, Baron; Zaitsev, Peter; Vadim Tkachenko, Jeremy D. Zawodny; Lentz, Arjen & Balling, Derek J. 2008, *High performance MySQL*, 2 uppl., Sebastopol: O'Reilly Media, Ltd.
- 2014, *TIOBE Programming community index*. Tillgänglig: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>, Hämtad: 26.4.2014.