Examensarbete

Plugin till e-handelssystem: En utredning av e-handelssystem åt Prisjakt AB

Av

Robert Kvant

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Sammanfattning

Prisjakt AB har sedan starten 2002 utvecklat en komplett informations- och jämförelsetjänst där man som konsument får möjlighet att sortera produkter utifrån egenskaper och pris. För att kunna tillhandahålla sin tjänst krävs aktuella pris- och produktuppgifter och en stor del av Prisjakts arbete består därför av att inhämta information om nätbutikers sortiment. Detta utförs antingen genom att Prisjakt skriver mjukvara som tolkar en butiks hemsida eller att butiken genererar en prisfil som innehåller en lista på butikens kompletta sortiment. Eftersom det är arbetskrävande att utveckla egen mjukvara är Prisjakts strävan hela tiden att få butikerna att skapa en prisfil.

Idag använder många butiker på nätet sig av e-handelssystem och det finns idag system som automatiskt kan generera en prisfil medan många saknar stöd. Målet med examensarbetet är att presentera det vanligaste e-handelssystemet som i nuläget saknar funktionalitet för att generera en prisfil. Inom ramen för examensarbetet har det även utvecklats ett plugin för att kunna generera en prisfil för det valda e-handelssystemet.

En analys utfördes där statistik togs fram över hur fördelningen av e-handelssystem såg ut för de nätbutiker som i nuläget inte levererade sina priser via en prisfil. Med hjälp av analyseringsmjukvaran Wappalyzer kunde en lista på de mest vanligaste e-handelssystemen tas fram. Dessutom inhämtades information från Prisjakts supportavdelning om vilket som var det vanligaste e-handelssystemet. Därefter gjordes en sammanvägning av de två resultaten.

E-handelssystemet Shopify valdes då det var vanligast av de system som saknade stöd för att skapa en prisfil. Därefter utvecklades ett plugin till e-handelssystemet Shopify som kunde skapa en prisfil.

Nyckelord: E-handelssystem, OAuth 2.0, JSON, Python

Abstract

Since 2002 Prisjakt AB has developed a complete information and product comparison service where you as a consumer get the opportunity to sort products based on characteristics and price. To be able to provide their service the first step is to get hold of current price and product information. A major portion of Prisjakts work is therefore dedicated to gather information about a stores range of products. One way to accomplish this is by writing software that parses and extracts information from the stores website. Another way is to let the store create a pricefile that lists the stores complete range of products. Because there is a lot of work involved in writing software Prisjakts endeavor is to get the stores to create their own pricefile.

Today many online-stores uses some kind of ecommerce-system and some of them can automatically generate a pricefile but many misses this functionality. The goal was to find the most common system that lacked this functionality. Within the scope of this thesis a plugin was also developed that was able to generate a pricefile for the chosen e-commerce system.

An analytic process was carried out with the purpose to gather information about the distribution of e-commerce systems for the customers not using pricefiles. Using the analysis software Wappalyzer, a list of the most common e-commerce systems could be created. Information about the most common e-commerce systems was also gathered from the support department of Prisjakt. Then a comparison of the two results was performed.

The e-commerce system Shopify was chosen because it was the most common system among those lacking support for creating a pricefile. A plugin was developed that was able to create a pricefile.

Keywords: Ecommerce, OAuth 2.0, JSON, Python

Förord

Detta examensarbete utfördes i samarbete med Prisjakt AB och bidrog för egen del till utökade kunskaper inom olika webb-tekniker och e-handelssystem.

Jag vill tacka alla berörda på Prisjakt AB och då framförallt Max Noack som varit behjälplig under arbetets gång.

Jag vill även rikta ett tack till min handledare Christin Lindholm och min examinator Christian Nyberg.

Terminologi

Ticket-system

Det journalsystem som supportavdelningen på Prisjakt använder för att registrera och skriva ner information om supportärenden.

Kommandotolk

Ett gränssnitt där en användare av en dator kan skriva in kommandon och köra program-skript.

Innehållsförteckning

1. Inledning

1.1. Bakgrund

Prisjakt AB har sedan starten 2002 utvecklat en komplett informations- och jämförelsetjänst där mottot är att man ska kunna hitta rätt produkt till rätt pris. Förutom möjligheten att sortera produkter utifrån pris och egenskaper tillhandahåller Prisjakt även betygsättning av butiker för att öka konsumenternas möjligheter till ett säkert köp [1].

Prisjakt fungerar översiktligt genom att nätbutiker tillhandahåller aktuell information om sitt produktutbud genom någon form av teknisk metod. I sin tur erbjuder Prisjakt konsumenter verktyg för att jämföra och sortera produkter utifrån pris och egenskaper. Dessa verktyg innefattar en webbsida samt applikationer för Android, Mac OS X, iPhone och iPad.

1.1.1. Inhämtning av data

För att kunna tillhandahålla tjänsten behöver Prisjakt i ett första steg tillgång till information om en nätbutiks produkter. Detta utförs på något av följande sätt:

1. Prisjakt skriver mjukvara som tolkar en nätbutiks hemsida och hämtar ner information om nätbutikens sortiment cirka tre till fem gånger per dygn
2. Nätbutiken genererar utifrån sin produktdatabas en prisfil som följer ett av Prisjakt förutbestämt format [2]. Prisjakt hämtar prisfilen från nätbutikens server cirka tre till fem gånger per dygn.

Eftersom metod 1 medför extra arbete för Prisjakt är strävan hela tiden att förmå nätbutikerna att själva generera en prisfil. Prisjakt ger därför på sin hemsida olika förslag på hur man som nätbutik kan skapa en prisfil [2]. Man kan enligt Prisjakt:

- Exportera en produktlista i form av en statisk textfil från ett produkt-, order- eller ekonomisystem.
- Installera ett programskript som skapar en prisfil utifrån en produktdatabas.
- Använda ett e-handelssystem som har stöd för att generera en prisfil.

Oavsett vilken metod som väljs så måste prisfilen följa de grundläggande kraven som innefattar rätt format och aktuell produktinformation.

1.1.2. E-handelssystem

E-handel som begrepp definieras som det som sker när någon köper eller säljer en vara från en butik på Internet [3]. Mellan år 2006 och 2016 har den svenska e-handelns omsättning ökat med över 300 procent och från att i början av år 2000 ha utgjort 1.1 procent av den totala detaljhandeln slutade år 2016 med en andel på 7.7 procent [3][4].

En nätbutik kan ses som en naturlig utveckling av äldre tiders postorder och skulle kunna innefatta en standardmässig hemsida med listade produkter och en funktion för att göra en beställning. Men med tanke på den digitala utvecklingen ställs helt andra krav idag. Förutom ett tilltalande utseende och möjlighet till enkel och effektiv administration av butiken krävs flera olika typer av tekniska lösningar för att klara sig i den ökade konkurrensen. Man måste exempelvis kunna tillhandahålla flera betalningssätt, ha goda sökmöjligheter och funktioner för att skapa merförsäljning [5]. I e-barometerns helårsrapport för 2016 kan man läsa att under det fjärde kvartalet gjorde i genomsnitt 35 procent av e-handelskonsumenterna ett köp med sin mobiltelefon [4]. Detta medför även ett ökat krav på anpassning för olika typer av enheter.

Denna utveckling ställer allt högre krav på de som agerar på marknaden och idag använder därför många nätbutiker sig av e-handelssystem för att få en komplett plattform för sin online-försäljning.

Mängden av e-handelssystem kan delas upp i ett antal olika kategorier. En kategori av system består av de leverantörer som tillhandahåller sin produkt enligt SaaS-modellen (Software as a service) vilket innebär att de står för både utveckling och drift av butiken. Detta innebär att den som köper tjänsten inte behöver ha några tekniska kunskaper överhuvudtaget. Exempel på leverantörer av den här typen av tjänst är det kanadensiska företaget Shopify [7].

En annan kategori innefattar de företag som säljer sin produkt under licens men där kunden själva får stå för driften av butiken. Detta medför att kunden själv får ha egna eller hyrda servrar och antingen egen eller inhyrd kompetens för att driftsätta och underhålla systemet. Exempel på leverantörer är Intershop som tillhandahåller e-handelssystemet Intershop Commerce Suite Enterprise [8].

Ytterligare en kategori innefattar de e-handelssystem som är av typen öppen-källkod och som kan användas fritt utan kostnad. Som föregående kategori kräver även den här typen av system egen eller inhyrd kompetens för driftsättning och underhåll. Ett exempel på den här typen av e-handelssystem är Magento som finns tillgängligt på Github [9].

När det gäller möjligheten att kunna automatiskt generera en prisfil utifrån en produktdatabas till ett av Prisjakt angivet format har vissa typer av e-handelssystem inbyggt stöd medan andra kräver anpassningar. Prisjakt har idag en lista över e-handelssystem som har inbyggd funktionalitet när det gäller att kunna generera en prisfil [6].

1.2. Syfte

Syftet med arbetet är att genom en övergripande undersökning analysera de e-handelssystem som det i nuläget inte finns stöd för hos Prisjakt och utifrån den informationen utveckla ett plugin för ett valt system.

1.3. Mål

Målet är att genom en övergripande utredning hitta det e-handelssystem som ännu inte är optimerat för Prisjakt, är vanligast och som det finns möjligheter till att vidareutveckla. Under

utredningsfasen ska förutom övergripande marknadsstatistik även intern information och nätbutikernas behov analyseras. Till det e-handelssystem som väljs - utifrån den övergripande utredningen - ska ett plugin utvecklas som kan utifrån nätbutikens kompletta produktdatabas generera en prisfil som följer Prisjakts format.

1.4. Problemformulering

Nedan listas de problemformuleringar som besvaras i examensarbetet:

- Får supportavdelningen några frågor angående plugin för olika typer av e-handelssystem?
- För vilket e-handelssystem som ännu inte stöds ska ett plugin utvecklas?
- Finns det restriktioner när det gäller att utveckla plugin till ett specifikt e-handelssystem?

1.5. Motivering av examensarbetet

Genom att utöka mängden av e-handelssystem som har stöd för prisfil kommer arbetsbördan minska för Prisjakt samt underlätta för de nätbutiker som i nuläget skapar en prisfil genom andra mer omständliga metoder.

Med fler butiker som tillhandahåller produktinformation med hjälp av en prisfil kan Prisjakt presentera aktuell och korrekt information. Genom att Prisjakt tillhandahåller uppdaterad information om nätbutikers sortiment kan samhällets konsumenter få tillgång till ett bättre verktyg för att jämföra priser och produkter. Detta kan i slutändan gynna konsumenter, nätbutiker och samhället i stort.

Ett intressant tekniskt ämne och företag gjorde att valet föll på att utföra ett examensarbete i samarbete med Prisjakt.

1.6. Avgränsningar

Det plugin som utvecklas ska endast kunna generera en prisfil som följer Prisjakts angivna format. Inget fokus på utseende eller andra funktioner ska ske.

2. Teknisk bakgrund

I detta kapitel beskrivs de programmeringsspråk, verktyg och tekniker som använts under examensarbetets genomförande.

2.1. JSON

JSON (JavaScript Object Notation) är ett textbaserat, programspråks-oberoende format för utbyte av data och bygger på JavaScripts notation för objekt och arrayer. Ett objekt - som definieras med två klammerparenteser - innehåller noll eller flera namn/värde-par där namn är en textsträng (inom citationstecken) och det associerade värdet en textsträng (inom citationstecken), ett tal, ett booleskt värde, ett objekt, en array eller värdet null [14]. Varje namn och tillhörande värde åtskiljs av ett kolontecken. Möjligheten att använda objekt och arrayer som värde i ett namn/värde par gör att man kan skapa nästlade strukturer.

JSON har blivit ett mycket populärt format när det gäller att utbyta information över Internet och används bland annat till webbtjänster och konfigurationsfiler [15].

```
{
            "ID": 32345,
            "name": "Scott Anderson",
            "married": true,
            "born": 1890,
            "children": [
                        "David Anderson",
                        "Emily Anderson
            ]
}
```

Fig. 1. Exempel på JSON-dokument.

JSON förekom och tolkades under den tekniska analysfasen som returvärde från mjukvaran Wappalyzer (avsnitt 2.2) samt under utvecklingsprocessen för pluginet som returvärde från Shopify API [51].

2.2. Wappalyzer

Wappalyzer är ett verktyg för att identifiera vilka bakomliggande tekniker som används på en specifik webbsida och kan enligt deras hemsida bland annat detektera innehållshanteringssystem, e-handelssystem, webb-ramverk och servermjukvara [10].

Programmet finns fritt tillgängligt som tjänst direkt på deras hemsida men kan även laddas ner och köras fristående genom kommandotolken. Wappalyzer finns även som tillägg för webbläsarna Firefox och Chrome. Eftersom Wappalyzer är av typen öppen-källkod kan vem som helst bidra till utvecklingen och all källkod finns därför tillgänglig på Github [11].

Enligt dokumentationen på deras hemsida så används reguljära-uttryck för att matcha kännetecken i bland annat html-kod, JavaScript-variabler och http-headers [20]. Alla reguljära-uttryck anges i filen *apps.json* [26].

```
"RxJS": {
   "cats": ["12"],
   "env": "^Rx$\\;confidence:20",
   "icon": "RxJS.png",
   "script":
"rx(?:\\.\\w+)?(?:\\.compat)?(?:\\.min)?\\.js",
   "website": "http://reactivex.io"
},
```

Fig. 2. Exempel ur apps.json

Som tillägg till varje reguljärt uttryck kan ytterligare information anges. Ett av de mer intressanta är *confidence* som anger hur säker identifieringen är, där 100 definieras som högsta värde. Är inte *confidence* definierat kommer värdet 100 att antas som standard [20].

I Fig. 2 matchas en global JavaScript-variabel *Rx*. Utvecklaren har angivit ett *confidence* på 20 som indikerar ett mindre pålitligt reguljärt uttryck.

Under examensarbetets genomförande användes Wappalyzer som analysverktyg för att ta reda på vilka e-handelssystem som används av de nätbutiker som saknar prisfil.

2.3. Docker

Docker är namnet på ett företag som tillhandahåller en så kallad *container-plattform* där en *container* är ett fristående exekverbart paket med mjukvara. Genom att paketera mjukvaran i *containers* får man enligt Dockers hemsida en isolerad mer säker miljö med målet att minska risken för kompabilitetsproblem oavsett bakomliggande infrastruktur [34]. Under den tekniska analysfasen användes Docker för att exekvera Wappalyzer då det finns tillgängligt som *container* [35].

2.3.1. Att använda Wappalyzer tillsammans med Docker

Efter att ha först installerat Docker kunde Wappalyzer installeras genom följande kommando:

```
docker pull wappalyzer/cli
```

Fig. 3. Installation av Wappalyzer

Med Wappalyzer installerat kunde det nu köras via kommandotolken med en internetadress som argument:

```
docker run --rm wappalyzer/cli http://www.cg.no/
```

Fig. 4. Exekvering av Wappalyzer genom Docker

Efter att kommandot som visas i Fig. 4 exekverats returnerades ett svar som utgjorde ett JSON-dokument som innehöll information om vilka tekniker som identifierats.

```
[...
{"name":"Magento","confidence":"100","version":"","ic
on":"Magento.png","website":"http://www.magentocommer
ce.com","categories":[{"6":"Ecommerce"}]} ...]
```

Fig. 5. Svar från Wappalyzer

I Fig. 5 visas ett utdrag från ett returnerat JSON-dokument. Förutom namnet på e-handelssystemet som identifierats finns även *confidence* angivet som anger hur säker identifieringen är (avsnitt 2.2).

2.4. MySQL Workbench

MySQL Workbench är ett visuellt verktyg som används för att designa, utveckla och administrera MySQL-databaser [12]. Med hjälp av programmet kan man via ett grafiskt gränssnitt ansluta sig till en databasserver som kör MySQL och:

- Designa och skapa databaser genom att visuellt rita upp E/R-diagram.
- Skapa, exekvera och optimera SQL-kommandon.
- Administrera databasservrar och användare.
- Analysera prestanda.
- Migrera data från andra databassystem till MySQL.

Under den tekniska analysfasen användes MySQL Workbench för att ansluta till och hämta ner information från Prisjakts databas om de nätbutiker som saknade prisfil.

2.5. SQLite

SQLite är ett databassystem som till skillnad från traditionella databashanterare inte körs som en separat process på en server. SQLite skriver direkt till filer på hårddisken och allt innehåll i en SQLite-databas ryms därför i en enda fil. Trots dess enkelhet tillhandahålls avancerade funktioner som transaktioner och triggers [13].

Under examensarbetet användes SQLite under den tekniska analysfasen för att spara ner uppgifter om vilka e-handelssystem som identifierats. SQLite användes även som lagringsmetod för det plugin som utvecklades.

2.6. Python

Python definieras ofta som ett objekt-orienterat skriptspråk [16]. Programspråket går idag att installera på Windows, Mac, Linux och ett antal mer specialiserade operativsystem [17].

Som standard kommer Python utrustat med ett stort kodbibliotek som innehåller allt ifrån funktioner för textmanipulation till kod för att skapa grafiska gränssnitt [18].

Under den tekniska analysfasen användes Python som programspråk för att skriva ett automationsskript som analyserade och sparade ner information om vilka e-handelssystem som identifierats för de nätbutiker som saknade prisfil.

2.6.1. Subprocess

Under den tekniska analysfasen uppkom behovet att via Python exekvera Wappalyzer genom Docker. För att lösa det problemet användes modulen subprocess som finns tillgänglig i Pythons standardbibliotek [19].

```
#!/usr/bin/env python3
import subprocess
proc=subprocess.run(["docker","run","--
rm","wappalyzer/cli","http://www.example.com"],
          stdout=subprocess.PIPE)
if proc.returncode == 0:
          print(proc.stdout)
```

Fig. 6. Anrop av Docker från Python med modulen subprocess

I Fig. 6 visas hur Docker anropas med hjälp av funktionen *run()* i modulen subprocess. JSON-svaret från Wappalyzer extraheras genom att anropa *stdout* på objektet *proc*.

2.7. OAuth 2.0

OAuth 2.0 är en teknisk metod för att delegera begränsad åtkomst för applikationer till en användares data utan att behöva ta det osäkra steget att lämna ut fullständiga inloggningsuppgifter [49]. Exempel på tjänster som använder sig av OAuth 2.0 för delegerad åtkomst är Twitter [50]. OAuth 2.0 möjliggör att en användare på Twitter kan ge åtkomst till en tredjeparts-applikation för att exempelvis hämta ner information om användarens Twitter-följare.

Innan en klientapplikation kan få tillgång till en resurs skyddad av OAuth 2.0 måste den delegeras åtkomst av en användare. OAuth 2.0

definierar fyra primära sätt som detta kan utföras på som utgörs av *authorization-code, password, client-credentials* och *implicit* [49]. I den här texten kommer endast *authorization-code* att beskrivas då den metoden användes under examensarbetets genomförande.

Oavsett metod är målet att som klientapplikation få tillgång till en *access-token*. Som innehavare av en giltig *access-token* kan man som klientapplikation komma åt resurser skyddade av OAuth 2.0. De vanligaste metoderna är att skicka med en *access-token* i en http-header eller som parameter i en URL. [49]

Under examensarbetet användes OAuth 2.0 under utvecklingsprocessen för pluginet då det krävdes för att kunna få tillgång till en nätbutiks data.

2.7.1. Authorization code

I protokollflödet för OAuth 2.0 förekommer följande aktörer:

- *Resursägare* som är användaren av tjänsten. Äger data på resursservern och kan delegera åtkomst åt andra till sin data. Exempel på en resursägare kan vara en person som har konto på Twitter och som vill dela med sig av information om sina Twitter-följare till en tredjeparts-applikation.
- *Resursserver* som lagrar resursägarens data skyddat av OAuth 2.0
- *Klient* är den applikation som vill få tillgång till en resursägares data på resursservern.
- *Auktoriseringsserver* ger ut *access-tokens* och tar emot godkännanden från resursägare gällande åtkomst till deras data.

I ett första steg registrerar utvecklaren sin applikation *Klient* hos *Auktoriseringsservern*. Utvecklaren anger en URL som fastställer var *Resursägaren* ska vidarebefordras efter att den har godkänt åtkomst för *Klient*. Ett *client_id* och en *client_secret* tilldelas *Klient* där *client_secret* är en hemlig nyckel.

Auktoriseringen utförs därefter i följande förenklade steg:

1. *Resursägaren* vidarebefordras av applikationen *Klient* till *Auktoriseringsservern*.
2. *Resursägaren* tillfrågas av *Auktoriseringsservern* om den godkänner åtkomst för *Klient*.
3. Om resursägaren godkänner kommer den att vidarebefordras till den URL som angavs av *Klient* under registreringen av applikationen. Som parameter i webbadressen kommer det att finnas med en *authorization-code.*
4. Applikationen *Klient* kan nu använda *authorization-code* för att erhålla en *access-token*. Kravet är att den tillsammans med *authorization-code* skickar med *client_secret* som den tilldelades vid registreringen av applikationen.

Efter att ha fått tillgång till en *access-token* kan den nu användas för att få åtkomst till *resursägarens* data.

2.8. PHP

PHP är ett skriptspråk av typen open-source som är speciellt lämpligt vid utveckling av webb-applikationer och var det språk som användes under utvecklingsprocessen för pluginet. Språket är serverbaserat vilket innebär att koden exekveras på webb-servern innan den returneras till klienten [64]. Med språket kommer det ett kodbibliotek med ett stort antal färdiga funktioner [65].

3. Metod

I följande avsnitt beskrivs de faser som examensarbetet innefattar.

- **Inläsning och faktainsamling**

  Denna del i examensarbetet handlade mycket om att bekanta sig med Prisjakts arbete och om e-handelssystem i allmänhet. Denna fas innehöll mycket kommunikation med anställda på Prisjakt som gav tips om olika verktyg och webbsidor som kunde vara till hjälp i arbetet. Det funderades mycket kring om det gick att undersöka vilket e-handelssystem som en nätbutik använder och denna del innefattade därför en stor del experimenterande med olika typer av färdig och egenskriven mjukvara. I denna fas initierades även en mailkonversation med supportavdelningen som i ett senare skede resulterade i ett möte.

- **Analys av e-handelssystem**

  Under den här delen i examensarbetets genomförande utfördes en analysprocess som bestod av två delar. I den här fasen av examensarbetet genomfördes ett möte med supportavdelningen som resulterade i information kring deras arbetsmetoder. Mjukvara för analys av e-handelssystem användes för att på teknisk väg försöka ta fram statistik över vilka e-handelssystem som användes av de nätbutiker som saknade prisfil. Utifrån resultatet från den tekniska analysmetoden togs de 10 vanligaste e-handelssystemen som skickades över till Prisjakts supportavdelning. De genomförde därefter sökningar i sitt ticket-system och noterade antalet träffar för varje system under de senaste 8 månaderna. Resultaten från den tekniska analysmetoden och supportavdelningens ticket-system sammanvägdes därefter för att ta fram en sammanslagen lista på de vanligaste e-handelssystemen. Utifrån denna lista valdes därefter det vanligaste e-handelssystem som saknade stöd för att skapa en prisfil. För det e-handelssystem som valdes utvecklades det därefter ett plugin som kunde skapa en prisfil.

- **Utveckling och test av plugin**

  Denna fas innefattade en utvecklingsprocess som resulterade i ett plugin för det vanligaste e-handelssystemet som saknade stöd för att generera en prisfil. En stor del av arbetet bestod först och främst i att bekanta sig med dokumentationen för att bilda sig en uppfattning om hur det fungerade att utveckla plugin för det valda systemet. Efter att ha studerat dokumentationen kunde en översiktlig modell över det plugin som skulle utvecklas ritas upp. I nästa steg användes programspråket PHP för att implementera den modell som ritades upp i ett tidigare skede. Det plugin som utvecklades installerades därefter och testades på en nätbutik. I detta steg kommunicerades det även med Prisjakt som verifierade att prisfilen skapades enligt rätt format.

3.1. Analysprocessen

Målet med examensarbetet var att hitta det mest vanligaste e-handelssystemet som i nuläget saknade stöd för att generera en prisfil och därefter utveckla ett plugin till det valda systemet. För att kunna ta reda på vilket e-handelssystem som det skulle utvecklas ett plugin till krävdes någon form av statistik över hur fördelningen av e-handelssystem såg ut för de nätbutiker som saknade prisfil. Det påbörjades därför en analysprocess som innefattade två övergripande delar Fig. 7.

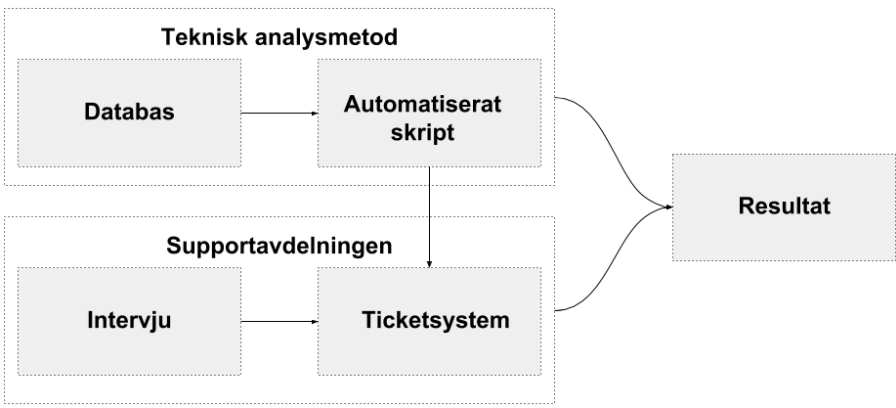

Fig. 7. Översiktlig bild av analysprocessen.

Den ena delen innefattade en teknisk analysmetod som bestod av följande delmoment:

- **Databas** som innebar att plocka ut internetadresserna till e-handelssystemen för de nätbutiker som saknade prisfil ur Prisjakts databas.
- **Automatiserat skript** där internetadresserna till e-handelssystemen användes som indata till ett egenutvecklat automatiserat skript som utnyttjade mjukvara för analys av e-handelssystem.

Den andra övergripande delen inriktades mot supportavdelningens arbete och bestod av följande delmoment:

- **Intervju** som utfördes som en *öppet riktad* intervju med hjälp av ett antal förutbestämda frågor om supportavdelningens arbete.
- **Ticket-system** där resultatet från det automatiserade skriptet under den tekniska analysmetoden användes för att göra sökningar i supportavdelningens ticket-system. Detta utfördes genom att utifrån den tekniska analysmetoden ta namnen på de 10 vanligaste e-handelssystemen och därefter göra sökningar och notera antalet träffar de senaste 8 månaderna för varje system i supportavdelningens ticket-system. Denna statistik vägdes därefter mot resultatet från den tekniska analysmetoden.

Målet var därefter att utifrån resultaten från de båda övergripande delarna ta fram ett sammanslaget resultat som kunde användas för att ta beslut om vilket e-handelssystem som det skulle utvecklas ett plugin till.

3.2. Teknisk analysmetod

Tidigt under examensarbetets genomförande undersöktes det om det med någon teknisk metod gick att komma fram till vilket e-handelssystem som en nätbutik använder. Vid diskussion med anställda på Prisjakts agentavdelning framkom att det ofta gick att hitta kännetecken i källkoden som avslöjade vilket e-handelssystem som används.

Som exempel analyserades ett antal nätbutiker som använder sig av e-handelssystemet Shopify [7]. Genom att studera källkoden kunde man se några gemensamma kännetecken.

Det förekom i källkoden för samtliga nätbutiker en JavaScript-variabel deklarerad som *Shopify*:

```
<script type="text/javascript">
var Shopify = Shopify || {};
Shopify.shop = "butikens-namn.myshopify.com";
...
```

Man kunde även på flertalet ställen i källkoden hitta samma URL: *cdn.shopify.com*.

I början av den tekniska analysfasen var därför tanken att skriva egen mjukvara som kunde användas för att leta efter kännetecken i källkoden. Men efter att ha studerat mängden av e-handelssystem som finns på marknaden framstod detta som ett allt för omfattande arbete. Detta bekräftas genom att studera Builtwith och deras topplista för e-handelssystem där det finns 19 (Cart Functionality ej inräknat) olika leverantörer representerade [21]. Det undersöktes därför om det fanns färdig analyseringsmjukvara tillgänglig.

Builtwith

Det första alternativet som utvärderades var Builtwith som är ett australiensiskt företag som specialiserat sig på att tillhandahålla statistik över teknologier på Internet [22]. Detta val grundade sig i en rekommendation från en anställd på Prisjakt.

På deras hemsida finns ett gratis verktyg för att analysera vilka tekniker som används på en webbsida [22]. Genom att skriva in en adress till en hemsida kan man som svar få tillbaka en lista på vilka tekniker som identifierats.

Genom att skapa ett gratis konto på deras hemsida kan man få tillgång till fler funktioner. Till exempel kan man ladda upp ett dokument med internetadresser till olika webbsidor och utifrån det generera en rapport som visar fördelningen av tekniker. Problemet var att på grund av den stora mängden webbsidor som skulle analyseras krävdes det en uppgradering av kontot. Detta kom att medföra en kostnad som

Prisjakt inte var intresserade av att betala. Detta alternativ fick därför uteslutas.

Netcraft

Netcraft är ett engelskt företag inriktat på säkerhet och internetstatistik som likt Builtwith erbjuder en gratis tjänst på sin hemsida för att identifiera webbtekniker, inkluderat e-handelssystem [23]. Inga möjligheter fanns att ladda upp ett dokument med internetadresser och det undersöktes därför om det gick att automatisera ett skript som kunde tolka Netcrafts webbsida. Deras juridiska avtal tillät inte detta [24] och därför fick Netcraft även uteslutas som alternativ.

Wappalyzer

Det alternativ som slutligen valdes var Wappalyzer som beskrivs under avsnitt 2.2. Wappalyzer valdes eftersom det var det enda alternativet kvar då övriga alternativ fick uteslutas på grund av kostnadsmässiga eller juridiska begränsningar.

3.2.1. Databasen

För att kunna gå vidare med den tekniska analysen krävdes tillgång till internetadressen till e-handelssystemet för varje nätbutik som saknade prisfil. Vid diskussion med anställda meddelades att den informationen fanns tillgänglig i Prisjakts databas.

Efter att ha fått tillgång till inloggningsuppgifter användes MySQL Workbench för att ansluta till databasservern. Ett sql-kommando exekverades som filtrerade ut alla de nätbutiker som saknade prisfil. Detta resulterade i en lista på 9162 rader innehållandes bland annat internetadressen till nätbutikens e-handelssystem. Denna lista exporterades därefter till hårddisken som en kommaseparerad fil.

3.2.2. Automatiserat skript

På grund av den stora mängden webbsidor som skulle analyseras undersöktes det hur man bäst kunde automatisera hela processen. Tidigare erfarenhet av programspråket Python (avsnitt 2.6) gjorde att valet föll på att använda det för att skriva ett automationsskript enligt

nedanstående modell. För att underlätta filtrering och gruppering av resultatet valdes att spara ner allt i en SQLite-databas [13].

För varje rad i den kommaseparerade filen:

- Plocka ut adressen till e-handelssystemet
- Anropa Wappalyzer (avsnitt 2.2) via Docker (avsnitt 2.3) med adressen till e-handelssystemet som argument.
- Från det returnerade JSON-svaret kontrollera om det har gått att identifiera något e-handelssystem. I så fall extrahera vilket eller vilka e-handelssystem som identifierats tillsammans med dess tillhörande värde på *confidence*.
- För varje e-handelssystem som identifierats:

  Spara ner namnet på e-handelssystemet och det tillhörande värdet på *confidence* i en databas tillsammans med företagsnamn och internetadress.
- I de fall som inget e-handelssystem kunnat identifierats:

  Spara ner företagsnamn och internetadress i databasen med kolumnen e-handelssystem satt till ’-’ för att markera ett oidentifierat system..

3.3. Supportavdelningen

För att få insikt i supportavdelningens arbete bokades det in ett möte med en anställd för intervju ansikte mot ansikte. Intervjun var av *öppet riktad* typ [68]och utgick från ett mindre antal fastställda frågor där svaren skriftligen nedtecknades. De frågor som ställdes var mer av en övergripande natur och användes främst som ett sätt att få igång en diskussion.

Som första del av intervjun ombads den anställda att berätta allmänt om arbetsuppgifterna som utförs på avdelningen. Det framkom att deras huvudsakliga arbete består av support till både användare av Prisjakts tjänster (konsumenter) och de företag som listar sina priser på deras hemsida (nätbutiker). Arbetet kunde bestå av allt ifrån att hjälpa till med inloggningsproblem till att felsöka och ge tips om prisfiler.

Vid en ytterligare fråga om vad de specifikt hjälper till med för problem med prisfilerna meddelades att de lokaliserar fel och ger därefter förslag på vad som ska åtgärdas för att filen ska kunna tolkas av Prisjakts system. En viktig del i arbetet består också av att tipsa de nätbutiker som saknar prisfil att skaffa plugin till sitt e-handelssystem i de fall som det är möjligt. Som kommentar nämndes att många nätbutiker undviker att skaffa plugin till sitt e-handelssystem då de tycker det är en för stor extrakostnad.

En fråga ställdes gällande om de använder något specifikt journalsystem där de för in uppgifter om sina kunder. Svaret som gavs var att de i nuläget använder en enklare form av ticket-system för att anteckna supportärenden men att de inte använde något system för att registrera fullständiga uppgifter om varje nätbutik. De hade därför inga uppgifter tillgängliga gällande vad en specifik nätbutik använder för e-handelssystem. Därför var det enda sättet att extrahera ut information genom enkla sökningar på nyckelord i deras ticket-system.

Eftersom det inte gick att få fram någon direkt statistik ur supportavdelningens system så valdes det istället att utgå ifrån den lista som togs fram under den tekniska analysfasen och därefter göra sökningar i ticket-systemet på de 10 mest vanligaste e-handelssystemen.

3.4. Källkritik

Majoriteten av de referenser som förekommer i källförteckningen är officiella förstahandskällor hämtade direkt från utvecklaren eller företagets hemsida. Eftersom uppgifterna kommer direkt från officiella källor får informationen anses som trovärdig.

De undantag som förekommer beskrivs nedan.

Referenser [3] och [5] är två artiklar hämtade från IIS som är en förkortning för Internetstiftelsen i Sverige. Förutom att vara ansvariga för topp-domänen .se arbetar de för att främja forskning, utbildning och undervisning med inriktning på Internet. Författaren till de länkade artiklarna - Henrik Rådmark - har även gett ut en bok i ämnet med titeln ”Rätt väg till lyckad e-handel” [67]. Med tanke på att det är en väl ansedd stiftelse som publicerat artiklarna och att de är

skrivna av en författare som givit ut en bok i ämnet får källan anses som trovärdig.

[4] är en rapport med titeln ”E-barometern årsrapport 2016”. Rapporten tas fram av Postnord i samarbete med Svensk Digital Handel och HUI Research. Förutom att Postnord i sig är ett välkänt företag har HUI Research kunder som utgörs av exempelvis Halmstad kommun och revisionsföretaget PwC [58]. Källan anses därför som trovärdig.

Referens [14] går till ett tekniskt dokument utgivet av IETF som är en förkortning för Internet Engineering Task Force. I det refererade dokumentet står beskrivet att texten genomgått en offentlig granskning och att det har blivit godkänt för publicering av IESG som står för Internet Engineering Steering Group. Vid en kontroll har IESG medlemmar från företag som Google, AT&T och Cisco samt en medlem från ETH Zürich [60]. Källan får därför anses som trovärdig.

Referenser [15][16] och [49] går till tre böcker utgivna av förlaget O’Reilly Media. Enligt deras hemsida har de sedan 1984 givit ut böcker inom datavetenskap och har idag kunder som Google, Amazon och Netflix [61]. Källan får därför anses som trovärdig.

Referenser [21][22][28][29][30][31] och [32] går till företaget Builtwith och deras hemsida. Builtwith är ett australiensiskt företag som specialiserat sig på att tillhandahålla statistik över teknologier på Internet. Det var svårt att hitta statistik över fördelningen av e-handelssystem från källor som ansågs pålitliga under examensarbetets genomförande. Builtwith ansågs dock vara det mest pålitliga och det grundade sig främst i att de har goda referenser från kunder som utgörs av flertalet kända företag [62]. Builtwith används även som referens i en artikel från tidningen MIT Technology Review [63].

Referens [68] går till en bok utgiven av förlaget Studentlitteratur. Enligt förlagets hemsida har de givit ut läromedel och kurslitteratur sedan 1963. Alla tre författarna är verksamma på den datavetenskapliga institutionen på Lunds Tekniska Högskola. Källan anses därför vara trovärdig.

4. Analys

Efter att ha fått fram information från både supportavdelningen och den tekniska analysmetoden analyserades resultatet med målet att komma fram till vilket e-handelssystem som det skulle utvecklas ett plugin till.

Som första steg sammanställdes resultatet från den tekniska analysmetoden genom att gruppera på e-handelssystemens namn och därefter sortera på antal. Utifrån den tekniska analysmetoden togs namnen på de 10 vanligaste e-handelssystemen som därefter skickades över till supportavdelningen. De gjorde därefter sökningar i sitt ticket-system och noterade antalet träffar för varje e-handelssystem under de senaste 8 månaderna. Resultaten från den tekniska analysmetoden och supportavdelningens ticket-system sammanvägdes därefter som resulterade i en sammanslagen lista. Utifrån den listan valdes därefter det vanligaste e -handelssystemet som saknade stöd för att skapa en prisfil.

4.1. Resultat teknisk analysmetod

Efter att med analysmjukvaran Wappalyzer gått igenom alla de 9162 raderna i den kommaseparerade filen kunde resultatet från den tekniska analysmetoden noggrant undersökas. Det första som uppmärksammades var att databasen innehöll 9185 rader. Det visade sig att 23 nätbutiker hade identifierats med två olika e-handelssystem var som totalt utgjordes av 46 rader i databasen. 29 av dessa rader hade ett värde på *confidence* som var 100.

5393 rader i databasen utgjorde nätbutiker som det inte hade gått att få fram något resultat för.

Totalt 67 rader i databasen innehöll nätbutiker som det hade gått att analysera fram ett e-handelssystem för men som hade ett värde på *confidence* som understeg 100.

För de nätbutiker som hade identifierats med två olika e-handelssystem där båda hade ett värde på *confidence* som var 100 gjordes valet att utesluta dessa ur resultatet. Detta omfattade totalt 6 butiker som utgjordes av 12 rader i databasen.

Utifrån den här informationen konstruerades ett sql-kommando (Appendix 8.1) som baserades på följande krav:

- Filtrera bort alla rader där det inte gått att identifiera något e-handelssystem.
- Filtera bort de rader som har ett värde på *confidence* som understiger 100.
- Filtrera bort de rader som innehåller nätbutiker som identifierats med två olika e-handelssystem och som har *confidence* satt till 100 i båda fallen.

Detta resulterade i totalt 3713 rader som grupperades utifrån namnet på e-handelssystemet och sorterades i fallande ordning på antal. Resultatet för de 10 mest vanliga e-handelssystemen visas i Table I.

TABLE I. RESULTAT TEKNISK ANALYSMETOD

E-handelssystem	**Antal**
Magento	1213
Prestashop	903
WooCommerce	351
OpenCart	242
Shopify	184
osCommerce	137
Salesforce Commerce Cloud	90
Textalk	69
EPages	66
Bigcommerce	56

När det gäller den stora mängden oidentifierade system kan det vara så att de nätbutikerna inte använder något e-handelssystem utan istället använder en egenutvecklad e-handelsplattform. Det skulle också kunna vara så att de använder ett för Wappalyzer känt e-

handelssystem som har skräddarsytts till den grad att det inte längre går att identifiera.

I ett värsta scenario skulle mängden av oidentifierade system kunna innehålla ett känt e-handelssystem som inte kan identifieras överhuvudtaget av Wappalyzer och som överstiger övriga identifierade system i antal.

För att studera detta närmare analyserades marknadsstatistik som sedan jämfördes mot den lista som beskriver de system som Wappalyzer ska kunna identifiera [27].

De nätbutiker som analyserades är verksamma på den svenska, brittiska, norska, irländska och nya zeeländska marknaden. Enligt Builtwith och deras topplistor för Sverige [28], Storbritannien [29], Norge [30], Irland [31] och Nya Zeeland [32] kan utav totalt 17 e-handelssystem 10 identifieras av Wappalyzer [27].

Utav de e-handelssystem som inte kan identifieras av Wappalyzer finns Squarespace Commerce och Wix Stores med på topplistorna för alla de analyserade marknaderna. Som exempel innehar Squarespace Commerce tredjeplatsen för e-handelssystem på den norska topplistan med en andel på 13.31 % [30]. Wix Stores hamnar på andra plats på den brittiska marknaden med totalt 13.65 % [29]. E-handelssystemet Mystore som inte heller kan identifieras av Wappalyzer hamnar på fjärde plats på den norska marknaden med en andel på 12.73% [30].

Utifrån denna statistik kan man konstatera att det kan i den oidentifierade mängden system förekomma kända e-handelssystem som Wappalyzer inte kan identifiera och som vid en bättre analysmetod hade kunnat placera sig på den topplista som visas i Table I.

Tankar kring kvalitén på Wappalyzer uppkom och då med tanke på att det utifrån den tekniska analysmetodens resultat förekom nätbutiker som hade identifierats med två olika e-handelssystem som vardera hade ett värde på *confidence* som var 100. Enligt dokumentationen på hemsidan för Wappalyzer så genomgår den kod man bidrar med en mängd automatiserade tester och en granskning av en ansvarig person innan den integreras i projektet [33]. Trots detta tillvägagångssätt väcks en del frågor. Vem som helst kan bidra till koden men vem ansvarar för att koden fungerar i framtiden när det

kommit en ny version av e-handelssystemet? Har den som skrivit koden och den som granskat tänkt på alla fall som kan förekomma och finns det risk för falska positiver?

För att komma fram till ett säkrare resultat hade man kunnat jämföra den topplista som visas i Table I med statistik som skapats utifrån Builtwith och deras tjänst som beskrivs i avsnitt 3.2.

4.2. Resultat supportavdelningens ticket-system

Utifrån den lista som togs fram utifrån den tekniska analysmetoden togs namnen på de 10 mest vanligaste e-handelssystemen som därefter skickades över till supportavdelningen. De gjorde därefter sökningar i sitt ticket-system och noterade antalet träffar för varje e-handelssystem under de senaste 8 månaderna Table II.

TABLE II. RESULTAT TICKET-SYSTEM

E-handelssystem	**Antal**
Textalk	192
Magento	64
WooCommerce	49
Shopify	45
Prestashop	35
Salesforce Commerce Cloud	30
OpenCart	13
EPages	8
osCommerce	5
Bigcommerce	3

4.3. Totalt resultat

För att få fram ett sammanslaget resultat summerades varje e-handelssystems placering i de båda listorna.

TABLE III. TOTALT RESULTAT

E-handelssystem	**Teknisk**	**Support**	**Totalt**
Magento	1	2	3
WooCommerce	3	3	6
Prestashop	2	5	7
Shopify	5	4	9
Textalk	8	1	9
OpenCart	4	7	11
Salesforce Commerce Cloud	7	6	13
osCommerce	6	9	15
EPages	9	8	17
Bigcommerce	10	10	20

Kolumnen *Teknísk* anger e-handelssystemets placering från den tekniska analysmetoden och kolumnen *Support* anger placeringen utifrån supportavdelningens ticket-system. De två kolumnernas värden summerades och skrevs in i kolumnen *Totalt* som därefter sorterades i stigande ordning. Resultatet visas i Table III.

Det som direkt utmärkte sig var resultatet för e-handelssystemet Textalk. Under den tekniska analysmetoden hade det placerat sig på åttonde plats men vid sökningar i supportavdelningens ticket-system på en första plats. Vid kontakt med supportavdelningen meddelade de att det redan fanns ett samarbete med Textalk som möjliggör att man som nätbutik kan skapa en prisfil [36]. Det stora antalet träffar på Textalk i ticket-systemet kunde enligt supportavdelningen förklaras med att alla nätbutiker som använder sig av Textalk länkar sin prisfil utifrån adressen *https://shop.textalk.se/shop*. Detta medför att många tickets gällande prisfiler för Textalk kommer innehålla den internetadressen och därmed öka på mängden av tickets som innehåller ordet Textalk. Nätbutiker som använder andra e-handelssystem har i många fall inte en så avslöjande adress till sin prisfil.

På grund av att det redan fanns stöd kunde Textalk uteslutas. Det borde dock undersökas av supportavdelningen varför det ändå förekommer nätbutiker som inte använder sig av prisfil.

När det gäller Magento utfördes det först en sökning på Magento Marketplace med nyckelorden ”prisjakt” och ”pricespy” [37]. Inga fria alternativ kunde hittas. Vid kontakt med supportavdelningen meddelades att de via en tredje-part hade ett plugin under utveckling.

För WooCommerce hittades ett fritt tillgängligt plugin [38][39]. Enligt en av hemsidorna var inte tillägget uppdaterat på över två år [39]. Samtidigt vittnade statistik på en av hemsidorna om att det fortfarande laddades ner [38].

När det gäller Prestashop finns ett gratis plugin som rekommenderades av supportavdelningen [40]. I det diskussionsforum där pluginet diskuteras finns inlägg som vittnar om att det fortfarande var aktivt den 16 augusti 2017 [40]. Vid en sökning med nyckelordet ”prisjakt” i deras app-store kunde dessutom tre alternativ hittas [66]. Ingen av dessa alternativ var gratis.

För Shopify fanns det inga plugin när man sökte på ”prisjakt” eller ”pricespy” i deras app-store [47].

När det gäller OpenCart finns det ett plugin som supportavdelningen tipsade om [41]. Pluginet är inte gratis. Inga fria alternativ kunde hittas vid en sökning på Google.

När det gäller Salesforce Commerce Cloud gick det inte att hitta någon information på deras hemsida som vittnade om integration med Prisjakt eller möjligheter till att utveckla plugin [42]. Detta bekräftades ytterligare vid en sökning på Google.

Gällande osCommerce så finns det redan ett plugin utvecklat av Prisjakt [2].

För EPages del hittades inget plugin vid en sökning på deras hemsida [43].

När det gäller Bigcommerce hittades inget plugin vid en sökning i deras app-store [44].

4.3.1. Val av e-handelssystem

Med information tillgänglig återstod nu att välja ett e-handelssystem.

Eftersom supportavdelningen meddelade att det fanns ett plugin för Magento på gång kunde det e-handelssystemet direkt väljas bort.

När det gäller WooCommerce finns det ett fritt plugin tillgängligt. Som tidigare beskrivits var pluginet inte uppdaterat på över två år men statistik vittnade samtidigt om att det fortfarande laddades ner [38][39]. För att få ett helt säkert svar på om pluginet fortfarande fungerade utfördes ett test.

WooCommerce är ett tillägg till det välkända CMS-verktyget Wordpress [46]. Därför installerades först Wordpress av senaste version som vid tidpunkten för testet var 4.9.1. Därefter installerades WooCommerce version 3.2.5 [45]. Slutligen installerades pluginet för att generera en prisfil [39]. Inga problem uppenbarade sig vid installationen och pluginet kunde generera en prisfil. Därför kunde även detta alternativ uteslutas.

När det gäller Prestashop finns det redan ett fritt tillgängligt plugin som rekommenderades av supportavdelningen [40]. Dessutom finns det tre stycken avgiftsbelagda alternativ tillgängliga i Prestashops app-store [66]. Utifrån den här informationen togs beslutet att även utesluta detta alternativ.

När det gäller Shopify finns inget plugin för skapande av prisfil i deras app-store [47]. Därför beslutades att välja Shopify då det samtidigt finns goda möjligheter till att utveckla plugin för det e-handelssystemet [48].

5. Resultat

Efter att ha fastställt att Shopify [7] var det e-handelssystem som det skulle utvecklas ett plugin till kunde utvecklingsprocessen påbörjas. Som första steg studerades dokumentationen för att på så vis bilda sig en uppfattning kring hur det fungerade att utveckla plugin till Shopify. Efter att ha studerat dokumentationen och valt den tekniska metod som skulle användas kunde en översiktlig modell över hur pluginet skulle fungera ritas upp.

Ett utvecklar-konto skapades på hemsidan för Shopify som gav tillgång till en butik där pluginet installerades och testades under utvecklingsprocessen.

En stor del av utvecklingsarbetet bestod av att implementera funktionalitet för att kunna installera pluginet och få åtkomst till en butiks data med OAuth 2.0. Efter att ha först implementerat OAuth 2.0 kunde därefter kod skrivas som resulterade i ett plugin till Shopify som kan generera en prisfil enligt Prisjakts format.

5.1. Utveckling av plugin

Enligt Shopifys dokumentation är ett plugin en separat webb-applikation som kommunicerar med Shopify API och som ges åtkomst till en butiks data med OAuth 2.0 [53]. Pluginet laddas därför inte upp och körs från Shopifys servrar utan måste installeras och exekveras på en egen separat webbserver och kommunicera med Shopify API via OAuth 2.0 [52].

5.1.1. Översiktlig modell

Ett grundläggande krav är att man vill kunna anropa en sida på nätbutikens domän för att få en prisfil genererad. Det beslutades att använda sig av en så kallad *Application Proxy* som innebär att man kan presentera data på en Shopify-butik som kommer från en extern källa [54]. Genom att ange en *Application Proxy* kan en förfrågan till exempelvis adressen *https://{butiks-url}/proxy-request* skickas vidare till en extern server. Svaret från den externa servern kommer därefter att returneras och presenteras på *https://{butiks-url}/proxy-request.*

Utifrån detta beslut kunde en översiktlig modell över hur pluginet ska fungera ritas upp Fig. 8.

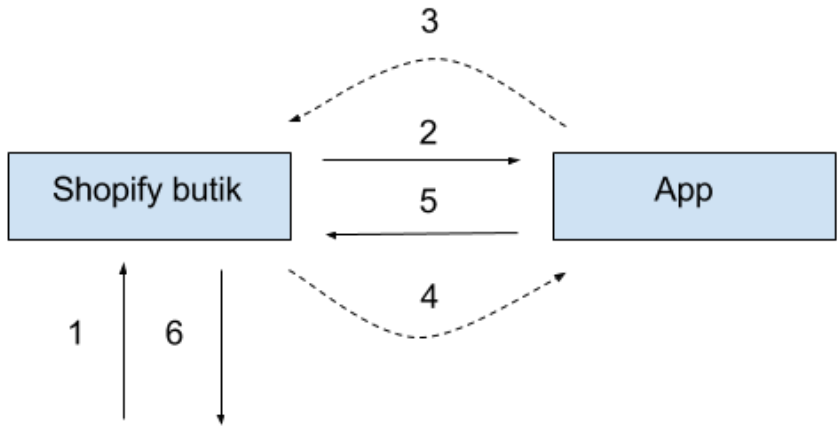

Fig. 8. Översiktlig modell över pluginet

I Fig. 8 är *App* en webb-applikation som finns på en extern webbserver och som har installerats med OAuth 2.0 på *Shopify butik*. Detta medför att *App* i det här fallet har tillgång till en *access.token* som gör att den kan hämta ner information om alla produkter som finns på *Shopify butik*. I följande exempel är det bestämt att en prisfil genereras genom att anropa *https://{url-shopify-butik}/pricefile*. I det här exemplet kan pluginet *App* nås på adressen *http://{url-app}/plugin*. En *Application Proxy* har konfigurerats som medför att anrop till adressen *https://{url-shopify-butik}/pricefile* kommer att skickas vidare till *http://{url-app}/plugin.*

1. En https-förfrågan görs till *https://{url-shopify-butik}/pricefile* som enligt tidigare beskrivning ska presentera en prisfil enligt Prisjakts format [2].
2. Med hjälp av en *Application Proxy* kommer förfrågan att skickas vidare till adressen *http://{url-app}/plugin.* I anropet till *App* kommer det att finnas med en url-parameter som anger vilken butik det gäller.
3. *App* kommer att göra en förfrågan till *Shopify API* gällande en lista på butikens alla produkter. Detta utförs genom att anropa *https://{url-shopify-butik}/admin/products.json* [56].

4. *Shopify API* returnerar en lista på butikens produkter.
5. *App* presenterar butikens produkter enligt Prisjakts format på prisfil [2].
6. Resultatet returneras och visas på *https://{url-shopify-butik}/pricefile*.

5.1.2. Förarbete

Innan utvecklingsarbetet påbörjades skapades ett utvecklar-konto på Shopifys hemsida. Med ett konto får man tillgång till en kontrollpanel där man kan registrera ett nytt plugin och en nätbutik som kan användas för att testa det plugin som utvecklas [51]. I samband med att man registrerar ett nytt plugin får man tillgång till ett *client_id* och en *client_secret* som krävs i protokollflödet för OAuth 2.0 [53]. I kontrollpanelen anges även den adress som butiksinnehavaren ska skickas tillbaka till då den godkänt pluginet [52].

Under den här delen av utvecklingsprocessen bestämdes även att använda PHP. Detta val grundade sig i tidigare erfarenhet av PHP samt att Prisjakt själva använder sig av det programspråket.

I de utdrag ur programfilerna som visas i följande avsnitt finns en fil *client.php* inkluderad som innehåller all bakomliggande funktionalitet som exempelvis anrop till Shopify API.

5.1.3. Installation

Innan ett plugin kan skicka förfrågningar till Shopify API måste det begära åtkomst och få ett godkännande av butiksinnehavaren. Detta sker med OAuth 2.0 [53].

Nedan beskrivs hur detta utfördes i programspråket PHP.

authorize.php

authorize.php innehåller en vidarebefordran till en sida på Shopifys servrar som visar en dialogruta där nätbutiksinnehavaren tillfrågas om den godkänner åtkomst för pluginet.

```
<?php
header("Location:
https://{$shop}.myshopify.com/admin/oauth/authorize?
client_id={$client_id}&
scope={$scopes}&
redirect_uri={$redirect_uri}")
?>
```

Fig. 9. Filen authorize.php

Filen *authorize.php* visas i Fig. 9. Adressen som användaren skickas vidare till innehåller följande variabler:

- *$shop* som innehåller namnet på nätbutiken som pluginet ska installeras på.
- *$client_id* är det *client_id* som tilldelades vid registreringen av pluginet (avsnitt 5.1.2).
- *$scopes* anger vilken del av nätbutikens data som pluginet vill få åtkomst till [55]. $scopes innehåller i det här fallet värdet ”read_products”.
- *$redirect_uri* anger den URL som nätbutiksägaren ska skickas vidare till då den godkänt pluginet. Denna URL måste överensstämma med den URL för vidarebefordran som angavs vid registreringen av pluginet [53].

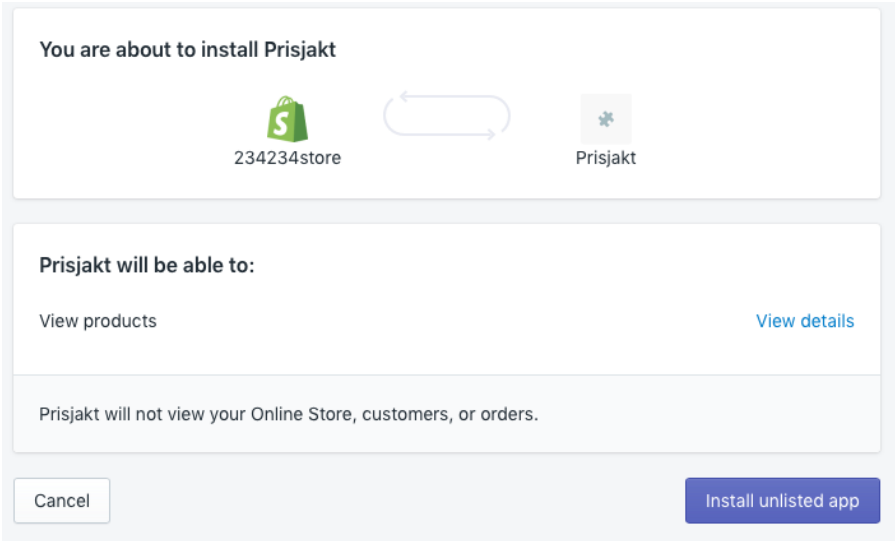

Fig. 10. Dialogruta vid installation av plugin

Genom att låta en nätbutiksägare som vill installera pluginet besöka *authorize.php* kommer den att skickas vidare till en sida på Shopifys servrar och bli presenterad för den dialogruta som visas i Fig. 10.

callback.php

När en nätbutiksägare klickar på installationsknappen på den dialogruta som visas i Fig. 10 kommer en vidarebefordran att ske till *callback.php*.

De parametrar som skickas med i anropet till filen *callback.php* är bland annat: *code*, *hmac*, *timestamp* och *shop* [53]. Parametern *code* är den *authorization-code* som beskrivs i avsnitt 2.7.1.

Innan *code* kan bytas ut mot en *access-token* ska man enligt dokumentationen utföra en verifiering av parametern *hmac*. Detta utförs genom att låta parametrarna *code*, *timestamp* och *shop* passera genom en hash-funktion SHA-256 med *client_secret* som nyckel [53].

```
<?php
require 'client.php';
$client = new Client();
if ($client->verifyHmac($code,$timestamp,
           $shop,$hmac,$client_secret)) {
  $access_token=$client- >getAccessToken(
           $shop,$client_id,$client_secret,$code);
  $client->saveToDb($shop,$access_token);
else{
 echo "Failed to verify HMAC";
}
?>
```

Fig. 11. Filen callback.php

I Fig. 11 visas filen *callback.php*. Först utförs en verifiering av parametern *hmac* genom ett anrop till funktionen *verifyHmac()*. Om verifieringen lyckas kommer ett anrop att ske till funktionen *getAccessToken()* som används för att utbyta en *access-token.* Som parametrar till funktionen *getAccesToken()* skickas:

- *$shop* som innehåller namnet på butiken.
- *$client_id* som är det *client_id* som tilldelades vid registreringen av pluginet (avsnitt 5.1.2).
- *$client_secret* som är den *client_secret* som tilldelades vid registreringen av pluginet (avsnitt 5.1.2).
- $code som innehåller *authorization-code* [53].

Slutligen kommer den access-token som hämtas med funktionen g*etAccessToken()* att sparas ner i en SQLite-databas [13].

5.1.4. Skapande av prisfil

Med tillgång till en *access-token* kunde nu pluginet hämta information om en butiks produkter från Shopify API.

Genom att skapa en *Application Proxy* i kontrollpanelen för utvecklarkontot kunde förfrågningar till *https://{butiks-url}/apps/pricefile* skickas vidare till filen *pricefile.php*.

Man kan genom Shopify API hämta ner en lista på en nätbutiks produkter genom att anropa *https://{butiks-url}/admin/products.json* [56]. Det som uppmärksammades var att man vid ett anrop endast

kunde hämta ner maximalt 250 produkter åt gången. Detta löstes genom att först anropa *https://{butiks-url}/admin/products/count.json* som returnerar det totala antalet produkter för en nätbutik. Genom att dela det totala antalet produkter med 250 kunde man fastställa hur många anrop till *https://{butiks-url}/admin/products.json* som måste utföras.

pricefile.php

Ett utdrag ur filen *pricefile.php* visas i Fig. 12. I anropet till *pricefile.php* kommer det att finnas med en parameter *shop* som anger vilken nätbutik som det ska genereras en prisfil för.

```
<?php
require 'client.php';
$client = new Client();
$access_token=getAccessTokenFromDb($_GET['shop']);
$count = $client->getProductCount($_GET['shop']);
header("Content-Type: text/plain");
$pages = ceil($count/250);
for ($page=1; $page==$pages; $page++) {
 $products=$client->getProducts(
           $_GET['shop'],$page,$access_token);
 foreach ($products['products'] as $product) {
           // Skriv ut produkter ...
           ...
?>
```

Fig. 12. Filen pricefile.php

Först hämtas nätbutikens *access-token* från databasen med funktionen *getAccessTokenFromDb()*. Därefter görs ett anrop till funktionen *getProductCount()* för att hämta det totala antalet produkter.

Antalet produkter divideras sedan med 250 för att få fram antalet anrop som ska utföras. Detta värde tilldelas variabeln *$pages*.

En for-loop används som körs utifrån värdet på *$pages*. En variabel *$page* används som räknare och anger den sida med produkter som ska hämtas. Vid varje iteration kommer ett anrop till *getProducts()* att ske där resultatet sparas ner i variabeln *$products*. Slutligen loopas alla produkter i variabeln *$products* igenom och presenteras enligt prisjakts format för prisfil [2].

5.1.5. Test av plugin

Under utvecklingsprocessen testades pluginet på den nätbutik som skapades i samband med registreringen av utvecklarkontot (avsnitt 5.1.2).

Som första steg användes butiken för att få rätt på installationen av pluginet med OAuth 2.0.

Under nästa steg i arbetet fylldes nätbutiken med ett antal demo-produkter som användes för att verifiera att prisfilen bestod av korrekt information och skapades enligt Prisjakts format.

Som slutligt steg skickades länken till nätbutikens prisfil över till Prisjakt som verifierade att den skapades enligt rätt format och kunde därmed läsas av deras system.

6. Slutsats

Genom att genomföra en övergripande utredning som bestod av både en teknisk analysmetod och information från supportavdelningen kunde fördelningen av e-handelssystem för de nätbutiker som saknade prisfil identifieras. Under analysfasen användes intern information från supportavdelningen samt Prisjakts databas och övergripande marknadsstatistik användes som jämförelsematerial under den tekniska analysmetoden. Utifrån resultatet från analysprocessen kunde de e-handelssystem som var vanligast identifieras. Resultatet användes sedan för att filtrera ut de system som i nuläget saknade stöd för att generera en prisfil och som det fanns möjligheter till att vidareutveckla. Utifrån denna analysprocess kunde ett plugin till e-handelssystemet Shopify utvecklas utifrån de krav som beskrivs i avsnitt 1.3. Detta uppfyller även syftet med examensarbetet som bestod av att genom en övergripande undersökning hitta och utveckla ett plugin för det e-handelssystem som det i nuläget inte finns stöd för hos Prisjakt (avsnitt 1.2).

Det plugin som utvecklades är i nuläget en grund att bygga vidare på och ska inte ses som en skarp produkt då det i nuläget inte uppfyller alla de krav som fastställts av Shopify [59]. Efter att ha bekräftat att pluginet är tillräckligt stabilt och följer de krav som Shopify föreskrivit behöver det installeras på en stabil webb-server.

Efter att ha säkerställt att pluginet uppfyller alla de krav som fastställts av Shopify [59] kommer koden att överlämnas till Prisjakt som kommer att installera det på sina servrar.

6.1. Problemformulering

Nedan besvaras de frågor som listades i kapitel 1.4

Får supportavdelningen några frågor angående plugin för olika typer av e-handelssystem?

Det framkom under intervjun med supportavdelningen att en del av deras arbete bestod av att hjälpa nätbutiker med prisfiler och att tipsa om plugin till e-handelssystem. Supportavdelningen fick inte frågor om plugin till specifika e-handelssystem utan deras främsta arbete

bestod av att hjälpa till med prisfiler och tipsa nätbutiker om olika plugin i de fall som det var möjligt. Det fanns inte heller möjlighet att hitta information om frågor kring plugin till e-handelssystem i deras journalsystem då deras system inte hade möjlighet att kategorisera tickets utifrån ämnesområde.

För vilket e-handelssystem som ännu inte stöds ska ett plugin utvecklas?

Under examensarbetets genomförande utfördes en analysprocess som bestod av två delar (avsnitt 3). En teknisk analysmetod utfördes där man med hjälp av mjukvaran Wappalyzer tog fram statistik över vilka e-handelssystem som används av de nätbutiker som i nuläget inte använder prisfil. Utifrån resultatet från den tekniska analysmetoden togs namnet på de 10 vanligaste e-handelssystemen som därefter skickades över till Prisjakts supportavdelning. De gjorde därefter sökningar på namnen i sitt ticket-system och noterade antalet träffar för varje e-handelssystem under de senaste 8 månaderna. Statistiken från supportavdelningen vägdes därefter med den lista som togs fram utifrån den tekniska analysmetoden. Utifrån den slutgiltiga listan valdes därefter Shopify som det e-handelssystem som det skulle utvecklas ett plugin till då det var det system som var vanligast och samtidigt saknade funktionalitet för att skapa en prisfil.

Finns det restriktioner när det gäller att utveckla plugin till ett specifikt e-handelssystem?

När det gäller restriktioner kring utveckling av plugin till specifika e-handelssystem så finns det i fallet med Shopify ett fastställt dokument som beskriver vad som krävs av ett plugin innan det får publiceras på deras app-store [59]. Restriktioner som förekommer i det dokumentet är exempelvis att pluginet inte får ha ett namn innehållande ordet ”Shopify” och att det måste tillhandahålla alla förfrågningar över HTTPS tillsammans med ett giltigt SSL-certifikat.

6.2. Reflektion över etiska aspekter

När det gäller Prisjakt kommer det plugin som utvecklades att medföra att arbetsbördan kommer att minska då färre manuella metoder måste användas. Detta kan medföra minskade kostnader för Prisjakt som istället kan lägga resurserna på att utveckla och förbättra

sin tjänst ytterligare. Detta kan i slutändan bidra till en ökad nytta för konsumenterna i samhället.

Det finns begränsningar när det gäller antalet anrop till *Shopify API* som bör tas i beaktande och som kan medföra missbruk. Enligt dokumentationen är antalet anrop begränsat och fungerar enligt en så kallad *leaky-bucket* algoritm [69]. Detta medför en *bucket-size* på 40 anrop och en *leak-rate* på 2 anrop per sekund. En person som vill illa skulle kunna skriva ett automatiserat skript som anropar en butiks prisfil med en frekvens som medför att *bucket-size* överskrids och som i sin tur gör att det plugin som utvecklats blir blockerat ifrån att hämta ner data från *Shopify API*.

6.3. Framtida utvecklingsmöjligheter

För att det plugin som utvecklades under examensarbetets gång ska kunna anses tillräckligt stabilt för att kunna släppas till allmänheten krävs en del åtgärder:

1. Under utvecklingsprocessen har en enkel webbserver avsedd för webb-utveckling använts och för att pluginet ska kunna släppas till allmänheten krävs att det installeras på en stabil webbserver.
2. Shopify har satt upp en lista som beskriver vad som krävs av ett plugin för att det ska kunna listas i deras app-store [59]. Det bör därför kontrolleras så att pluginet uppfyller alla de krav som listas i det dokumentet.
3. Pluginet måste genomgå en kontroll där det säkerställs att tillräcklig felhantering är implementerad.
4. Funktionalitet måste implementeras för att undvika missbruk av den typ som beskrivs i avsnitt 6.2. Ett alternativ skulle kunna vara att begränsa åtkomst till prisfilen baserat på IP-adresser som tillhör Prisjakt då man i anropet från en *Application Proxy* kan avläsa klientens IP-adress i fältet *X-Forwarded-For* [54]. Ett annat alternativ är att ha en process i bakgrunden som med jämna mellanrum går igenom och hämtar ner nätbutikernas produkter och sparar ner dessa i en fil eller databas. Vid ett anrop till en prisfil hade man då kunnat hämta ner produktsortimentet från en fil eller databas istället för att varje

gång göra ett anrop till *Shopify API* och därmed riskera att bli blockerad.

7. Källförteckning

[1] "*Om Prisjakt – Kunskap före köp*" https://www.prisjakt.nu/info.php?t=about_company [2017-10-27]

[2] "*Vad är en prisfil? – Kunskap före köp*" https://www.prisjakt.nu/info.php?t=for_stores_price [2017-10-27]

[3] "*Vad är e-handel? | IIS*" https://www.iis.se/lar-dig-mer/guider/ratt-vag-till-lyckad-e-handel/vad-ar-e-handel/ [2017-10-27]

[4] "*E-barometern 2016 årsrapport*" https://www.iis.se/docs/e-barometern-arsrapport-2016.pdf [2017-10-27]

[5] "*Att bygga en nätbutik | IIS*" https://www.iis.se/lar-dig-mer/guider/ratt-vag-till-lyckad-e-handel/att-bygga-en-natbutik/ [2017-10-27]

[6] "*E-handelslösningar Prisjakt*" https://www.prisjakt.nu/misc/ecommerce_solutions.php [2017-10-27]

[7] "*Shopify Press and Media*" https://www.shopify.com/press [2017-10-27]

[8] "*Intershop Commerce Suite Enterprise - Intershop Communications AG*" https://www.intershop.com/intershop-commerce-suite-enterprise [2017-11-28]

[9] "*Magento - Github*" https://github.com/magento [2017-11-28]

[10] "*Wappalyzer - Identify technologies on websites*" https://wappalyzer.com [2017-11-28]

[11] "*GitHub - AliasIO/Wappalyzer: Cross-platform utility that uncovers the technologies used on websites*" https://github.com/AliasIO/Wappalyzer [2017-11-28]

[12] "*MySQL :: MySQL Workbench*" https://www.mysql.com/products/workbench/ [2017-11-28]

[13] "*About SQLite*" https://www.sqlite.org/about.html [2017-11-28]

[14] "*RFC 7159 - The JavaScript Object Notation (JSON) Data Interchange Format*" https://tools.ietf.org/html/rfc7159 [2017-12-01]

[15] A. Rauschmayer, "*Speaking JavaScript*," Sebastopol CA 95472, O'Reilly Media, 978-1-449-36503-5, 2014.

[16] M. Lutz, "*Learning Python, Fifth Edition*," Sebastopol 95472, O'Reilly Media, 978-1-449-35573-9, 2013.

[17] "*Download Python for Other Platforms | Python.org*" https://www.python.org/download/other/ [2017-12-01]

[18] "*The Python Standard Library - Python 3.6.4 documentation*" https://docs.python.org/3/library/index.html [2017-12-01]

[19] "*17.5. subprocess — Subprocess management - Python 3.6.4 documentation*" https://docs.python.org/3/library/subprocess.html [2017-12-01]

[20] "*Wappalyzer - Developer Documentation*" https://www.wappalyzer.com/docs/specification [2017-12-01]

[21] "*Ecommerce technologies Web Usage Statistics*" https://trends.builtwith.com/shop [2017-12-01]

[22] "*Builtwith Technology Lookup*" https://builtwith.com [2017-12-01]

[23] "*Netcraft | Internet Research, Anti-Phishing and PCI Security Services*" https://www.netcraft.com [2017-12-01]

[24] "*Netcraft | Fair Use, Copyright*" https://www.netcraft.com/about-netcraft/fair-use-copyright/ [2017-12-01]

[25] "*Wappalyzer/driver.js at master · AliasIO/Wappalyzer · GitHub*" https://github.com/AliasIO/Wappalyzer/blob/master/src/drivers/npm/driver.js [2017-12-01]

[26] "*Wappalyzer/apps.json at master · AliasIO/Wappalyzer · GitHub*" https://github.com/AliasIO/Wappalyzer/blob/master/src/apps.json [2017-12-01]

[27] "*Wappalyzer - Ecommerce*" https://www.wappalyzer.com/categories/ecommerce [2017-12-01]

[28] "*Ecommerce usage in Sweden*" https://trends.builtwith.com/shop/country/Sweden [2017-12-01]

[29] "*Ecommerce usage in the United Kingdom*" https://trends.builtwith.com/shop/country/United-Kingdom [2017-12-01]

[30] "*Ecommerce usage in Norway*" https://trends.builtwith.com/shop/country/Norway [2017-12-01]

[31] "*Ecommerce usage in Ireland*" https://trends.builtwith.com/shop/country/Ireland [2017-12-01]

[32] "*Ecommerce usage in New Zealand*" https://trends.builtwith.com/shop/country/New-Zealand [2017-12-01]

[33] “*Wappalyzer - Developer Documentation*” https://www.wappalyzer.com/docs [2017-12-01]

[34] ”*What is a Container | Docker*” https://www.docker.com/what-container [2017-12-01]

[35] ”*wappalyzer/cli – Docker Hub*” https://hub.docker.com/r/wappalyzer/cli/ [2017-12-01]

[36] ”*Prisfiler från Textalk Webshop till Prisjakt.nu*” https://www.textalk.se/webshop/partners//prisjakt/ [2017-12-01]

[37] ” *Magento Extensions &Themes for Your Online Store | Marketplace*” http://marketplace.magento.com [2017-12-01]

[38] ”*WooCommerce Pricefiles*” https://managewp.org/plugins/details/woocommerce-pricefiles [2017-12-01]

[39] ”*WooCommerce Pricefiles – WordPress-tillägg*” https://sv.wordpress.org/plugins/woocommerce-pricefiles/ [2017-12-01]

[40] ” *Module PrestaCenter XML Export Free (PS 1.5.x) - Free Modules & Themes - PrestaShop*” https://www.prestashop.com/forums/topic/276093-module-prestacenter-xml-export-free-ps-15x/ [2017-12-01]

[41] ”*OpenCart - Prisjakt - Prisfil v1.04*” https://www.opencart.com/index.php?route=marketplace/extension/info&extension_id=7917 [2017-12-01]

[42] ”*CRM - ledaren inom SaaS (Leader In Software-as-a-Service) - Salesforce Sverige*” https://www.salesforce.com [2017-12-01]

[43] ”*Apps & integrations for your ePages shop*” https://www.epages.com/sell-online/apps-and-integrations/ [2017-12-01]

[44] ”*Ecommerce Apps Marketplace | BigCommerce*” https://www.bigcommerce.com/apps/ [2017-12-01]

[45] ”*WooCommerce - WordPress Plugins*" https://wordpress.org/plugins/woocommerce/ [2017-12-01]

[46] ”*Blog Tool, Publishing Platform, and CMS - WordPress*" https://wordpress.org/ [2017-12-01]

[47] ”*App Store: Ecommerce App Marketplace by Shopify*" https://apps.shopify.com [2017-12-01]

[48] ”*Shopify App Developer Program*" https://developers.shopify.com [2017-12-01]

[49] R. Boyd, “*Getting Started with OAuth 2.0*,” Sebastopol 95472, O’Reilly Media, 978-1-449-31160-5, 2012.

[50] ”*Oauth with the Twitter API — Twitter Developers* " https://developer.twitter.com/en/docs/basics/authentication/overview/oauth [2017-12-01]

[51] ”*Getting started - Developer resources - Shopify Help Center*" https://help.shopify.com/api/getting-started [2017-12-01]

[52] ”*Building an application - API tutorials - Shopify Help Center"* https://help.shopify.com/api/tutorials/building-public-app [2017-12-01]

[53] ”*OAuth - Authentication - Shopify Help Center"* https://help.shopify.com/api/getting-started/authentication/oauth [2017-12-01]

[54] ”*Application proxies - API tutorials - Shopify Help Center"* https://help.shopify.com/api/tutorials/application-proxies [2017-12-01]

[55] ”*OAuth - Authentication – Scopes - Shopify Help Center"* https://help.shopify.com/api/getting-started/authentication/oauth#scopes [2017-12-01]

[56] ”*Product - Admin API - Shopify Help Center"* https://help.shopify.com/api/reference/product#index [2017-12-01]

[57] ”*Product - Admin API - Shopify Help Center"* https://help.shopify.com/api/reference/product#count [2017-12-01]

[58] ”*HUI - Referenser"* http://www.hui.se/om-oss/referenser_1 [2017-12-05]

[59] ”*Requirements - Requirements and success criteria - Shopify Help Center"* https://help.shopify.com/api/listing-in-the-app-store/app-requirements-and-success-criteria/app-review-checklist [2017-12-05]

[60] ”*IETF | IESG Members"* https://www.ietf.org/iesg/members.html [2017-12-05]

[61] ”*About - O'Reilly Media"* http://www.oreilly.com/about/ [2017-12-05]

[62] ”*Customers - Builtwith"* https://builtwith.com/customers [2017-12-06]

[63] ” *Seeking Edge, Websites Turn to Experiments - MIT Technology Review"* https://www.technologyreview.com/s/523671/seeking-edge-websites-turn-to-experiments/ [2017-12-06]

[64] ”*PHP: What is PHP? - Manual"* http://php.net/manual/en/intro-whatis.php [2017-12-06]

[65] ”*PHP: PHP Manual - Manual"* http://php.net/manual/en/ [2017-12-06]

[66] ”*PrestaShop's Official Marketplace - Modules and templates - PrestaShop Addons"* https://addons.prestashop.com/en/ [2017-12-20]

[67] ”*Rätt väg till lyckad e-handel"* https://www.iis.se/docs/lyckadehandel_webb.pdf [2017-12-20]

[68] M. Höst och B. Regnell och P. Runesson, “*Att genomföra examensarbete*“ Lund, Studentlitteratur, 978-91-44-00521-8, 2006.

[69] ”*API call limit – Getting started – Shopify Help Center"* https://help.shopify.com/api/getting-started/api-call-limit [2018-01-12]

8. Appendix

8.1. Sql-kommando för att gruppera och sortera e-handelssystem

```
SELECT count(*),eCommerce FROM companies WHERE
eCommerce!='-' AND confidence='100' AND store_id NOT IN
(SELECT store_id FROM companies WHERE confidence='100'
GROUP BY company HAVING count(*)>1)
GROUP BY eCommerce ORDER BY count(*) DESC;
```