Yhteyden luonti AdWordsin ja tietokannan välille

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietotekniikka

Riihimäki, kevät 2013

Timo Aarnio

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

RIIHIMÄKI
Tietotekniikan koulutusohjelma

Tekijä Timo Aarnio **Vuosi** 2013

Työn nimi Yhteyden luonti AdWordsin ja tietokannan välille

TIIVISTELMÄ

Työn tarkoituksena oli suunnitella ja toteuttaa sovellusrajapinta AdWordsin ja MySQL-tietokannan välille. Työn toimeksiantajana oli Zoined Oy, jolle sovellus tulee mahdollisesti täysipäiväiseen asiakaskäyttöön. Sovellus luotiin toimimaan Zoinedin palvelinympäristössä. Projektin ohjelmointikielenä oli PHP.

Ohjelmoinnissa käytettiin apuna Googlen luomaa ohjelmointikirjastoa, jolla AdWordsin rajapinnan komennot saadaan PHP:n ymmärtämään muotoon. Apuna käytettiin myös Googlen referenssimateriaalia, josta saatiin ohjeita sovelluksen luontiin. Lisäksi ohjelmassa käytetään XML-lukijaa, jolla AdWordsista saatu tieto luetaan ja muunnetaan MySQL:ään vietävään muotoon. Asiakkaiden autentikoitumiseen käytetään OAuth 2.0 todentamisprotokollaa.

Raportin teoriaosuudessa käsitellään projektissa käytettyjä tekniikoita ja kuvaillaan niiden toiminnallisuutta. Lisäksi teoriaosuudessa kuvaillaan vaihtoehtoisia ohjelmointikieliä, joille Google on kehittänyt ohjelmointikirjaston. Tiedonkeruuseen käytettiin eri tekniikoihin keskittyneitä kirjoja ja verkkosivustoja.

Nykyisellään sovellus saa aikaan halutut toiminnot ja se toimii hyvänä runkona jatkokehitykseen. Jatkossa sovellusta voidaan laajentaa kattamaan kaiken AdWordsin tarjoaman tiedon. Laajennus onnistuu ilman, että tämän hetkisiä toimintoja tarvitsee muuttaa huomattavasti.

Avainsanat PHP, AdWords, MySQL, Oauth, verkko-ohjelmointi

Sivut 28 s. + liitteet 8 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Riihimäki
Degree Programme in Information Technology

Author Timo Aarnio **Year** 2013

Subject of Bachelor's thesis Creating a connection between AdWords and MySQL

ABSTRACT

The purpose of the project was to design and create an application programming interface between AdWords and MySQL database. The work was done for Zoined Oy, who might take the final product to customer use. The software was created to run on the servers of Zoined. The software was programmed in PHP.

To help the programming process, a client library was used. The client library is provided by Google and it is used to convert the AdWords API commands into PHP format. Google's reference library for AdWords was used as a help service during programming. The program also uses an XML reader to read data gathered from AdWords and convert it to the format used by MySQL. Authentication of the users is done with OAuth 2.0 framework.

The theoretical part covers the basics of the techniques used during the project. Their functionality is also explained. There is also a part explaining alternative programming languages which have the client library made by Google. Various books and websites focused on the techniques were used as help in gathering the information.

At its current state the software works as intended and provides the required functionalities. It works as a good base for further development. In future the software can be expanded to include all of the data provided by AdWords. The expansion is possible to make with just minor adjustments to the existing program.

Keywords PHP, AdWords, MySQL, OAuth, web programming

Pages 28 p. + appendices 8 p.

KÄSITTEITÄ

Access token	Vahvistustunnus	OAuthin käyttämä kirjautumistunnus, joka korvaa perinteisen käyttäjänimen ja salasanan
Application programming interface (API)	Ohjelmointirajapinta, rajapinta	Määritelmä, jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja eli keskustella keskenään.
Array	Taulukko	Ohjelmointikielissä yleisesti ilmenevä dynaaminen taulukko, johon voidaan väliaikaisesti tallentaa tietoa.
Callback URL	Paluuosoite	Osoite, johon OAuth palauttaa käyttäjän, kun ensimmäinen todennus on suoritettu loppuun.
Client Id	Sovellus id	OAuth todentamisvaiheessa tarvittava id numero, jolla sovellus tunnistetaan Googlen päässä.
Client Secret	Sovelluksen piilokoodi	OAuthin todentamisvaiheessa tarvittava piilokoodi, jolla vahvistetaan sovelluksen id aidoksi.
Comma-Separated Values (CSV)		Tiedostomuoto, jolla tallennetaan taulukkomuotoista tietoa tekstitiedostoon. Taulukkorakenteen kentät on eroteltu toisistaan pilkulla.
Document Object Model (DOM)		Standardi, jonka avulla voidaan manipuloida dynaamisesti dokumenttipuuta.

Hypertext Transfer Protocol (HTTP)	Hypertekstin siirtoprotokolla	Protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.
Java Database Connectivity (JDBC)		Java-ohjelmointikielen rajapinta, joka määrittelee tavan, millä asiakassovellus voi käyttää tietokantaa.
Linux		Avoimen lähdekoodin käyttöjärjestelmäperhe. Se on maailman käytetyin palvelinkäyttöjärjestelmä.
Object Oriented Programming	Olio-ohjelmointi	Ohjelmoinnin lähestymistapa, jossa koodi pilkotaan lyhyempiin ja helpommin ymmärrettäviin osasiin.
Portable Document Format (PDF)		Adoben kehittämä ohjelmistoriippumaton tiedostomuoto.
Refresh token	Päivitystunnus	OAuthin käyttämä tunnus, jolla uusitaan vahvistustunnus.
Server	Palvelin	Tietokone, joka tarjoaa siinä suoritettavien palvelinohjelmistojen välityksellä erilaisia palveluja muille ohjelmille.
Simple API for XML (SAX)	Yksinkertainen rajapinta XML:lle	Ohjelmistotekniikassa käytetty sarjamuotoinen XML-muotoisen tiedon käsittelyrajapinta.
Simple Object Access Protocol (SOAP)		XML-kieleen pohjautuva tietoliikenneprotokollakoneiden väliseen kommunikointiin tietoverkoissa.
Structured Query Language (SQL)		IBM:n kehittämä standardoitu kyselykieli relaatiotietokannoille.
The AdWords Query Language (AWQL)	AdWords kyselykieli	SQL:n tapainen kieli, jota käytetään AdWords rajapinnan raporttikyselyihin.

Web Service Description Language (WSDL)		XML pohjainen kieli, jota käytetään web-palveluiden rajapintojen määrittämiseen. Erittelee abstraktin ja sisällön toisistaan.
World Wide Web Consortium (W3C)		Kansainvälinen yhteisö, joka kehittää avoimia standardeja verkkoympäristöön.

SISÄLLYS

1 JOHDANTO

Verkkomainostamisen määrä on kasvanut vuosi vuodelta. IAB:n mukaan heinä-syyskuun aikana vuonna 2012 verkkomainonnan osuus koko mediamainonnasta oli 12,2 prosenttia (IAB Finland 2012.). Verkkomainonnassa Google on selvästi kärjessä. Vuonna 2010 Googlella oli 44.1 prosentin markkinaosuus verkkomainonnassa (ZenithOptimedia 2011). Kasvaneen kysynnän takia kilpailu mainospaikoista on entistä suurempaa. Mainosten tarkka kohdentaminen oikealle lukijakunnalle on elintärkeää, jotta mainostaminen pysyy tuottavana.

Kohdentamista helpottamaan verkkomainontasivustot tarjoavat paljon tietoa, jota käyttäjät voivat tutkia ja käsitellä tarpeidensa mukaan. Yleensä palveluiden omat käsittelyohjelmat eivät kuitenkaan ole tarpeeksi tehokkaita yritysten tarpeisiin. Google on tarjonnut AdWords-asiakkailleen mahdollisuuden tehdä omia ohjelmia tiedon käsittelemiseen. Tämä on mahdollistettu AdWords API:lla, joka luo rajapinnan AdWords-palveluun.

Tiedon nouto omille palvelimille mahdollistaa, että sitä voidaan tarkastella syvemmin. Tiedon pohjalta voidaan luoda erilaisia kaavioita ja sitä voidaan verrata muiden palvelujen tietojen kanssa ristiin. Omilla sovelluksilla tästä prosessista saadaan helpompaa ja nopeampaa.

Työn toimeksiantaja Zoined Oy on vuonna 2011 perustettu IT-alan yritys. Yrityksen toimitilat sijaitsevat Helsingissä. Zoined on erikoistunut analytiikkaan ja tarjoaa asiakkailleen johtamisen palveluja. Palvelut toimivat verkkoympäristössä, ja ovat keskitettynä yhteen paikkaan. Täältä asiakkaat näkevät helposti tietoja yrityksensä tuottavuudesta ja tehokkuudesta. Palvelun perusideana on muuntaa asiakkaan antama data, kuten myyntimäärä, ymmärrettävään ja havainnollistavaan muotoon. Opinnäytetyön yhtenä perusideana olikin lisätä uusi tiedonkeruukanava vanhojen rinnalle.

Työn tavoitteena oli suunnitella ja toteuttaa rajapinta AdWordsin ja tietokannan välille. Ohjelmointikieli oli vapaa. Ainoana rajoituksena oli, että ohjelman tuli toimia Zoinedin palvelin- ja tietokantaympäristössä. Zoinedissa oli toteutusvaiheessa käytössä Linux-pohjainen Apache-palvelin. Tietokantaympäristönä oli MySQL. Ohjelmoitaessa käytettiin apuna Googlen tarjoamaa ohjelmistokirjastoa, joka tarjosi tehokkaat työkalut AdWords rajapinnan käyttöön. Kirjasto rajoitti kielivalikoimaa kuuteen kieleen, joista lopulta päädyttiin PHP-kieleen.

Vaatimuksena oli, että ohjelma luodaan mahdollisimman automaattiseksi. Näin Zoinedin ja asiakkaan kontakti ohjelmaan minimoitaisiin. Tämä tarkoitti sitä, että ohjelman tuli osata luoda tarvittavat tietokannat automaattisesti. Ohjelman piti myös osata hakea kaikkien asiakkaiden tiedot mahdollisimman itsenäisesti. Uusien asiakkaiden mukaantulon täytyi olla mahdollisimman helppoa ja vaivatonta. Tähän käytettiin OAuth 2.0 -protokollaa, jolla asiakkaat todennettiin palveluun.

2 TEKNISEN TOTEUTUKSEN VALINTA

Oikean tekniikan valitseminen projektiin oli tärkeää, sillä väärien ratkaisujen teko heti alkuvaiheessa olisi voinut hidastaa toteutusta ja aiheuttaa lisätyötä loppuvaiheessa. Olisi esimerkiksi ollut huono ratkaisu valita ohjelmointikieleksi sellainen, jota Zoinedin palvelimet eivät tukisi. Vastaavasti erilaisen tiedon noudon tavan valitseminen olisi saattanut osoittautua kohtalokkaaksi jatkokäsittelyvaiheessa.

2.1 AdWords

AdWords on Googlen tarjoama verkkomainostuspalvelu. Sen kautta asiakkaat voivat luoda mainoksia, joita näytetään verkkosivuilla asiakkaan asettamien toiveiden mukaan.

Palvelu käynnistettiin vuonna 2000. Alun perin palvelun toiminta perustui kuitenkin hinta per tuhatta näyttöä malliin. Tällöin asiakas maksoi sivuille latautuneista mainoksista, riippumatta siitä, klikattiinko niitä kertaakaan. Vuonna 2002 AdWords kuitenkin muutti nykyiseen maksu per klikkaus malliin (Geddes 2012, 39).

Maksu per klikkaus -mallissa asiakas maksaa vain, kun mainosta klikataan. Mainoksen näyttäminen sivulla ei kuitenkaan maksa mitään. Klikkauksen hinta määräytyy pääosin sen mukaan, paljonko avainsanalla on arvoa ja kuinka hyvät laatupisteet mainos saa. Paremmilla laatupisteillä mainos pääsee halvemmalla ylemmäs tarjouskilpailussa. Näin todellinen klikkaus voi maksaa eri mainostajilla eri hinnan riippuen mainoksen osuvuudesta (Jacobson 2011, 12).

Mainosten näkyvyys määräytyy laatupisteiden mukaan. Googlen mukaan laatupisteiden määrään vaikuttavat muun muassa avainsanan osuvuus, avainsanan aiempi napsautussuhde, kotisivujen laatu sekä maantieteellinen kohdistus (Google 2012).

2.2 AdWords API

AdWords API on Googlen tarjoama rajapinta, jolla voidaan luoda yhteys AdWordsin ja oman sovelluksen välille. Rajapinta toimii verkkosovellusrakenteen keskikerroksessa, jossa luodaan yhteys käyttäjän sovelluksen ja palvelimen välille. AdWords API perustuu SOAP 1.1 tietoliikenneprotokollaan. Se käyttää apuna XML-pohjaista WSDL kieltä tietojen siirtämiseen (Google Developers 2012a). Tämän ansiosta rajapinta on käyttöliittymäriippumaton. Näin sitä voidaan käyttää missä tahansa ohjelmointiympäristössä. SOAP, XML ja WSDL ovat kuitenkin melko haastavia kieliä ymmärtää, etenkin jos rajapintaa käyttää ensimmäistä kertaa.

Käyttöönottoa helpottamaan Google on luonut erillisen käyttäjäkirjaston. Kirjastoon Google on luonut itse SOAPiin tarvittavat komennot ja muuntanut ne kirjaston ohjelmointikielelle. Kirjasto on saatavilla muun muassa PHP:lle, C#:lle ja Javalle. Kirjasto on rakennettu olio-ohjelmoinnin sään-

töjen mukaan. Näin rajapinta on helppo ottaa käyttöön, sillä ohjelmoijan ei tarvitse luoda itse kaikkia toimintoja, vaan apuna voidaan käyttää Googlen luomia komentoja. Se helpottaa ohjelmoijan työtä ja nopeuttaa sovelluksen kehittämistä huomattavasti.

Rajapinta mahdollistaa AdWordsin monipuolisemman käytön. Sillä voidaan toteuttaa kaikki AdWordsin sivustoilla olevat ominaisuudet, mutta perusominaisuuksia voidaan kehittää omissa sovelluksissa eteenpäin. Esimerkiksi palveluun voidaan luoda automatisoidusti uusia ja muokata olemassa olevia mainoksia edellisten päivien tulosten perusteella.

AdWordsin rajapinta poikkeaa maksullisuudellaan muista Googlen rajapinnoista. Googlen mukaan (Google Developers 2012b) rajapinnan käytön maksullisuudella varmistetaan, että sitä käytetään tehokkaasti ja vastuullisesti. Hinnat vaihtelevat eri palveluista riippuen. Raporttien lataus on kuitenkin ilmaista.

Rajapinnan uusimmat versiot työstövaiheessa olivat v201209 ja v201302. Koodi ja kuvaukset seuraavat näitä versioita.

2.2.1 Tiedon nouto AdWordsista

AdWordsissa on kolme tapaa, jolla tietoa voidaan hakea kampanjoista. Tieto voidaan hakea suoraan taulukkoon ohjelman muistiin, tai se voidaan tallentaa tiedostoon joko ohjelmallisesti tai AWQL:ää apuna käyttäen.

Jos tiedot halutaan näyttää suoraan ruudulla, tiedon haussa käytetään rajapinnan kampanjapalvelua. Sen avulla tiedot haetaan array-taulukkoon. Näin tiedot ovat heti ohjelman muistissa käyttövalmiina. Taulukosta tietoa voidaan ohjelmallisesti hakea tarpeen mukaan (Liite 1). Tämä hakutapa on paras, jos halutaan nopeasti saada pieni määrä tietoa näytölle. Kampanjapalvelun käyttäminen maksaa jonkin verran, joten sitä ei kannata käyttää isoihin hakuihin.

Isoja hakuja varten AdWordsiin on lisätty raportointipalvelu, jolla voidaan hakea tietoa suoraan tiedostoon. Tuettuja tiedostomuotoja ovat muun muassa CSV, XML ja PDF (Google Developers 2012c). PDF-muodossa tiedoista saadaan helposti luettava lista, CSV ja XML ovat tarkoitettu jatkokäsittelyä varten.

Raportointipalvelu on ilmainen käyttää. Näin sen avulla voidaan helposti hakea kaikki tilin tiedot ilman lisäkuluja. Raportointipalvelun etuna on myös se, että se muuntaa luvut automaattisesti oikeaan muotoon. Esimerkiksi prosenttiluvut näkyvät heti prosenttilukuina. Kampanjapalvelua käytettäessä nämä muunnokset pitää tehdä itse, mikä lisää hieman ohjelmointityötä.

Raportin tiedot voidaan hakea puhtaasti koodimäärittelyllä (Liite 2). Tällöin määriteltävät kentät ja tiedot luodaan koodissa. Esimerkiksi valittavat kentät laitetaan itse array-taulukkoon. Myös muut tiedot tallennetaan ensin

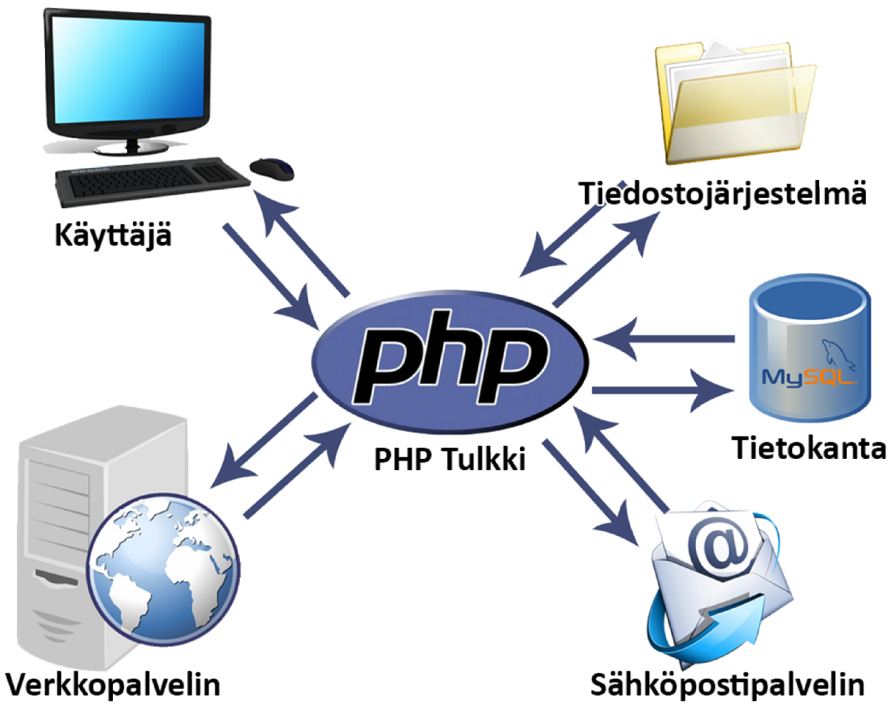
itse muuttujiin, josta sovellus hakee ne. Tämä on pidemmän päälle raskasta tehdä, sillä hausta tulee sekava.

Tätä helpottamaan Google on kehittänyt AdWordsia varten oman AWQL-kielen. Kieli vastaa SQL:ää rakenteeltaan. Kielen sanoitus on kuitenkin tehty AdWordsin ja sen rajapinnan ympärille. Koska AWQL vastaa SQL:n tyyliä, on se helppo ottaa käyttöön, jos on ennemmin työskennellyt tietokantojen parissa. AWQL myös lyhentää itse koodin määrää huomattavasti. Periaatteessa koko kysely voidaan luoda yhdellä komennolla. Käytännössä se kannattaa kuitenkin paloitella muutamaan osaan selkeyden vuoksi (Liite 3).

2.3 PHP

PHP on vuonna 1995 julkaistu skriptauskieli, joka luotiin alun perin helpottamaan verkko-ohjelmointia. PHP:n koko nimi oli aluksi Personal Home Page Tools, mutta vuosien saatossa se vaihtui muotoon PHP: Hypertext Preprocessor. Kielen tavoitteena oli minimoida tarvittavan koodin määrä tulosten saamiseksi (Hudson 2007, 1).

PHP on avoimen lähdekoodin kieli ja sitä kehitetään jatkuvasti. Avoimuuden takia kieli on ilmainen käyttää ja siihen voidaan tarvittaessa itse luoda lisäominaisuuksia. Ilmaisuuden ja avoimuuden ansiosta PHP:sta on tullut yksi WWW-ohjelmoinnin johtavista kielistä. W3Techs (2013) ilmoittaa 78,8 % verkkosivuista käyttävän PHP:ta. PHP toimii kaikilla yleisimmillä käyttöjärjestelmillä. Se myös tulee yleensä verkkopalvelimen mukana. Näin koodi on helppo ottaa käyttöön ja sitä voidaan ajaa lähes missä tahansa ympäristössä.



Kuvio 1. PHP:n yhdistyminen eri palveluihin esitetty kuviona.

PHP:ssa on sisäänrakennettuna suuri kokoelma luokkakirjastoja eri palveluihin (Kuvio 1). Mainittavimpana on vahva yhteys tietokantoihin. PHP:ssa on valmiina käskyt yhteyden luontiin sekä tietokantojen käsittelyyn. Tämä suoraviivaistaa tietokannan hallintaa ja helpottaa niiden käyttöä.

PHP muuntaa kirjoitetun koodin sarjaksi käskyjä, joita se suorittaa kunnes päästään koodin loppuun. PHP toimii siksi eri lailla verrattuna tyypillisiin ohjelmakieliin, joissa koodista ensin käännetään erillinen ajettava tiedosto ja käytetään sitä. Koska tätä välivaihetta ei ole, on PHP:n ohjelmoiminen nopeampaa. Normaalisti tällainen ajamismuoto vaatisi koneelta paljon resursseja, mutta tämä on korjattu PHP:ssa välimuistin käytöllä ja sen tehokkaalla automatisoidulla tyhjentämisellä (Hudson 2007, 3).

```
<html>
    <head>
        <title>PHP for-loop esimerkki</title>
    </head>
    <body>
        <?php
        // Tallennetaan int-tyyppinen arvo muuttujaan.
        $arvo = 1;
        /* Laskuri alkaa nollasta. Niin kauan, kun i on
         * pienempi kuin 5,
         * aja koodi ja lisää 1 laskuriin.
         */
        for($i=0 ; $i<5 ; $i++){

            //Tulosta muuttuja.
            echo $arvo;

            //Lisää muuttujaan 1.
            $arvo++;

        }
        ?>
    </body>
</html>
```

Kuvio 2. For-loopin ja muuttujien käyttö PHP-sovelluksessa.

Edellä oleva ohjelmakoodi (Kuvio 2) näyttää ruudulla tekstin ”12345”. Kahdella kauttaviivalla sekä kauttaviivalla ja tähdellä alkava osio ovat niin sanottuja kommenttirivejä. Nämä rivit eivät osallistu itse koodin ajoon millään tavalla. Sen vuoksi niihin voidaan kirjoittaa mitä halutaan. Koska PHP-koodi ajetaan ennen HTML:ää, myös HTML lähdekoodissa näkyisi PHP rivien tilalla ”12345”.

Työhön valittiin PHP, sillä siitä ja sen yhdistymisestä MySQL:ään oli jo aiempaa kokemusta. Toteutushetkellä PHP5 oli kielen uusin versio. Opinnäytetyön sisältö perustuu kyseiseen versioon.

2.4 Vaihtoehtoiset kielet

Google tarjoaa AdWordsin käyttäjäkirjastoa myös muilla kielillä. Nämä ovat Java, C#, Python, Ruby ja Perl. Kaikkien näiden käyttäjäkirjastot ovat käytännössä samanlaisia. Tässä opinnäytetyössä esitettyjen koodiesimerkkien pitäisi siksi toimia myös muilla kielillä pienten muokkauksien jälkeen.

2.4.1 C# ja .NET

C# on Microsoftin kehittämä ohjelmointikieli. Se yhdistelee C++ ja Java kieliä, sekä Microsoftin omaa Visual Basicia. Kielellä pystytään luomaan verkko-, työpöytä- ja konsolisovelluksia. C# on tarkoitettu .NETin, Microsoftin oman ohjelmistokehyksen, ohjelmointiin (Mojica 2003, ix).

C# on olio-ohjelmointiin keskittynyt kieli. Kieli tukee myös muita ohjelmointitapoja, mutta normaalisti nekin liitetään olio-ohjelmointitavan luomaan runkoon. Kieli sisältää vahvan luokkarakenteen. Jokaisessa C# kielellä tehdyssä ohjelmassa on vähintään yksi luokka, jota käytetään ohjelman ajamiseen.

.NET on tarkoitettu verkko-ohjelmointiin. .NET mahdollistaa sovellusten luonnin verkkoympäristöön. Ohjelmistokehys on tehty kokonaan verkkomaailman ehdoilla. Tämän takia esimerkiksi funktioiden ja toiminnallisuuden siirtäminen verkkosivulta toiselle on osaltaan automatisoitu. Näin ohjelmoijan ei tarvitse itse luoda näitä toimintoja, vaan hän voi käyttää toimintoihin tarkoitettuja käskyjä, jotka automaattisesti suorittavat tarvittavat koodit, jotta siirto onnistuu. Tämä helpottaa ohjelmoijan työtä, sillä aikaa ei kulu epäoleelliseen osaan. Toisaalta tällä myös varmistetaan, että siirrossa ei ole ongelmia, sillä koodi on jo ennestään testattu ja todettu toimivaksi Microsoftin puolesta (Powell 2002).

2.4.2 Java

Java on luokkapohjainen olio-ohjelmointikieli. Se pohjautuu C ja C++ -kieliin. Javan ominaisuutena on, että sama koodi toimii alustalla kuin alustalla, kunhan se vain sisältää Java tuen. Näin ohjelmaa ei tarvitse kääntää kuin kerran. Tämä on saatu aikaiseksi kehittämällä virtuaalikone, joka asennetaan oman käyttöjärjestelmän päälle. Virtuaalikone on puolestaan ohjelmoitu käyttöjärjestelmäkohtaisesti. Java on tehty mahdollisimman turvalliseksi. Se ei esimerkiksi sisällä vaarallisia rakentajia, joilla ohjelma saattaisi tehdä suunnittelemattomia asioita (Gosling 2000, 1).

Java tarvitsee MySQL Connectorin toimiakseen MySQL-tietokantojen kanssa. Connector muuntaa Javan luomat JDBC pyynnöt MySQL:n käyttämään verkkoprotokollamuotoon (MySQL 2013a). Jotta Connector toimisi, se ladataan ja sijoitetaan oman ohjelman luokkarakenteeseen mukaan

(MySQL 2013b). Tähän tiedostoon viitataan koodissa ja koodi hoitaa lopun automaattisesti.

2.4.3 Python

Python on yksinkertaisuuteen tähtäävä ohjelmointikieli. Tätä on tavoiteltu jättämällä monia muista kielistä tuttuja merkkejä kokonaan pois. Myös komennot ovat selvälukuisia englanninkielisiä sanoja muiden kielien epämääräisempien käskyjen sijaan. Se tukee Olio-ohjelmointia, mutta sitä ei ole pakko käyttää (Donaldson 2008, 2). Yksinkertaisuudella tähdätään ohjelmoinnin helppouteen ja suoraviivaisuuteen. Samalla ohjelmointiprosessia nopeutetaan, sillä ohjelmoijalla ei kulu yhtä pitkää aikaa käskyjen luonnissa, kuin esimerkiksi C#-kieltä ohjelmoitaessa.

Yksinkertaisuutensa ansiosta Python on parhaimmillaan lyhyitä skriptejä ja verkkosivustoja luotaessa. Ymmärrettävyytensä ansiosta kieltä käytetään myös peliohjelmoinnissa. Kieli ei toisaalta sovellu isompiin projekteihin, kuten käyttöjärjestelmien luontiin (Donaldson 2008, 3).

2.4.4 Ruby

Ruby on avoimen lähdekoodin olio-ohjelmointikieli. Kieli lainaa rakenteensa Perliltä ja yhdistää sen esimerkiksi Pythonin kaltaiseen yksinkertaiseen ohjelmointimuotoon. Rubyyn on helppo siirtyä, jos on aiempaa kokemusta jommasta kummasta ympäristöstä (Berman, J. 2). Rubylla tehtyjä ohjelmia ei ensin käännetä ajettavaan muotoon. Sen sijaan ohjelman koodi ajetaan käyttämällä ohjelmointikielen tulkkia, joka lukee koodia rivi riviltä. Näin Rubylla tehdyt ohjelmat toimivat kaikkialla, missä on jokin Rubya tukeva tulkki asennettuna. Samoin on myös Linux Apache-palvelimilla. Lisäosa on kuitenkin varsin usein jo oletuksena asennettuna palvelimille. Suurena syynä tähän on Ruby on Rails -ohjelmistokehys. Ruby soveltuukin parhaiten yksinkertaiseen skriptaukseen sekä verkkoympäristössä ajamiseen.

2.4.5 Perl

Perl on vanhin kielivaihtoehdoista. Sen ensimmäinen versio kehitettiin jo vuonna 1987. Kieltä on kuitenkin vuosien varrella kehitetty eteenpäin, ja nykyään se on jo viidennessä versionumerossa. Kuudetta versiota ollaan parhaillaan tekemässä. Kieli oli alun perin kehitelty tekstinkäsittelyyn, mutta uusien ominaisuuksien myötä se soveltuu laajempaan käyttöön. Kielen kehittämisessä on tähdätty sen tehokkuuteen tyylikkyyden sijaan (Perldoc n.d.). Perl soveltuukin parhaiten pinnan alla tehtäviin prosesseihin. Sillä voidaan kuitenkin luoda myös käyttöliittymiä.

Perlillä tehtyä koodia ei ensin käännetä ajettavaan muotoon. Sen sijaan skriptiä luetaan tulkilla, joka lukee ja toteuttaa koodia rivi kerrallaan. Ikänsä ja pitkäaikaisen suosionsa ansiosta kyseinen lukija tulee mukana

lähes jokaisessa Linux-versiossa. Toimiakseen Apachen kanssa, täytyy koneelle kuitenkin asentaa mod_perl. Tämä toimii jatkuvasti toimivana Perl-tulkkina verkkopalvelimella. Mod_perlin avulla Perl-koodin ajaminen nopeutuu, sillä aikaa ei kulu itse Perlin käynnistämiseen. Lisäosan asentaminen on kuitenkin yksinkertaista ja sekin on asennettuna lähes jokaiselle kaupalliselle palvelimelle oletuksena.

2.5 XML

XML on W3C:n suosittelema standardi, jolla määritellään rakenteellisia merkkauskieliä. XML luo perustan, jonka päälle voidaan luoda sovelluksia ja uusia kieliä. Sitä voidaankin kutsua kehykseksi, jonka sisälle varsinainen sisältö laitetaan. XML itsessään on pelkkää tekstidataa. XML on tarkoitettu tiedonsiirtoon ja -vaihtoon internetissä. XML:n avulla ei kuitenkaan pystytä pelkästään näyttämään mitään, vaan siihen täytyy käyttää muita kieliä, kuten HTML:ää (Nykänen 2003).

XML on vapaasti määriteltävä. Näinpä sen toiminnot voidaan määrittää ohjelmakohtaisesti vastaamaan sen omia tarpeita. XML toimii yhdessä verkkoprotokollien ja mekanismien kanssa mahdollistaen komentojen toimimisen laitteesta riippumatta. XML:ää suunniteltaessa tähdättiin kielen ymmärrettävyyteen. Koska dokumentin rakenne voidaan itse määrittää, on sen rakenne luettavissa ja siksi helposti ymmärrettävissä (Kuvio 3).

```
<?xml version="1.0" encoding="UTF-8"?>
<esimerkki>
    <varit>
        <vari>sininen</vari>
        <vari poikkeus=”vaalea”>vihrea</vari>
    </varit>
    <muodot>
        <muoto>nelio</muoto>
        <muoto>kolmio</muoto>
    </muodot>
</esimerkki>
```

Kuvio 3. Esimerkki XML dokumentista.

2.6 MySQL

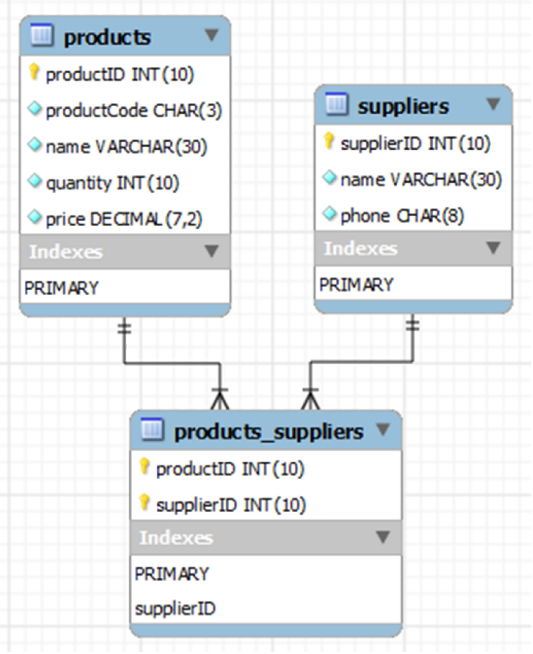
MySQL on ilmaiseksi käytettävä avoimen lähdekoodin tietokanta. Ilmaisuudestaan huolimatta MySQL on kuitenkin erittäin tehokas tietokantapohja. Se skaalautuu hyvin, mahdollistaen jopa 50 miljoonan tallenteen kantoja ilman huomattavaa suorituskyvyn laskua (Stucky 2001, 4). Kannan koolle ei olla määritelty ylärajaa. MySQL on saatavilla kaikille yleisimmille käyttöjärjestelmille. Se myös tulee yleisesti Apache-verkkopalvelimien mukana.

MySQL on relaatiotietokanta. Siinä kanta koostuu tauluista. Tauluihin on lisätty sarakkeita, joilla määritellään mitä tietoa tauluun halutaan. Jokaiselle sarakkeelle on myös määritelty tietotyyppi, jolla määritellään millaista

tietoa siihen voidaan syöttää. Näitä tauluja voidaan yhdistää halutulla tavalla. Tavoitteena on kuitenkin, että yksi tieto ilmentyy vain yhdessä paikassa, ja se lähetetään muihin tauluihin relaatioiden avulla (Sarja 2006).

Normaalisti relaatiokantaan voi luoda kolmea erilaista yhteyttä; yhdestä yhteen, yhdestä moneen ja monesta moneen. Yhdestä yhteen yhteydessä yksi taulun tieto voi yhdistyä vain yhteen tietoon toisessa taulussa. Hyvänä esimerkkinä tästä on ihmisen henkilötunnus. Samaa henkilötunnusta ei voi olla kuin yhdellä ihmisellä, joten tietojen väliin tulisi yhdestä yhteen linkki. Yhdestä moneen yhteydessä yhdellä päätiedolla voi olla useita alatietoja, mutta ei toisinpäin. Esimerkiksi yhdellä taiteilijalla voi olla useita maalauksia, mutta yleensä yhdellä maalauksella ei ole useita taiteilijoita. Vastaavasti monesta moneen yhteyttä käytetään, jos monella päätiedolla voi olla monta alatietoa, mutta alatieto voi olla yhteydessä moneen päätietoon. Asunnossa voi olla useita esineitä, mutta toisaalta sama esine voi olla useammassa asunnossa.

MySQL:ssä monesta moneen yhteys saadaan aikaiseksi luomalla kahden taulun välille uusi taulu, joka yhdistää taulut toisiinsa. Liitos luodaan tekemällä yhdestä moneen liitokset väliin jäävään tauluun (Kuvio 4).



Kuvio 4. Esimerkki monesta moneen yhteydestä.

Taulujen yhdistäminen luodaan avaimilla. Pääavain, yleensä rivin id-numero määrittää taulun rivit erilleen. Näihin saadaan yhteys toisesta taulusta käyttämällä viiteavainta. Viiteavain on yleensä oma kenttänsä, johon haettavan tiedon vastaava id-numero laitetaan.

MySQL noudattaa SQL-standardia, mutta siihen on lisätty myös omia laajennuksia. Standardoinnin ansiosta tietokannan käyttöön on helppoa siirtyä toisesta SQL-kannasta. Toisaalta lisätyt ominaisuudet mahdollistavat kannan helpomman ja tehokkaamman käsittelyn. Esimerkiksi REPLACE komento on hyvä lisä MySQL:ssä, jota ei ole SQL-standardissa. Komennon avulla tieto joko lisätään suoraan kantaan, tai jos vanha samaa vastaava

tieto on jo kannassa, se poistetaan ja korvataan uudella. Näin koko tarkastus- ja korvausprosessi on tehty yhdellä käskyllä.

2.7 OAuth

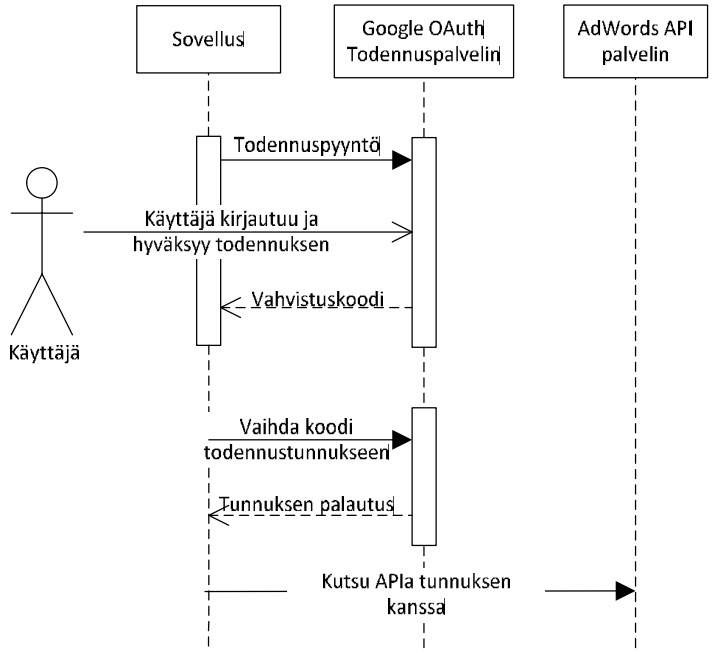
OAuth on vuonna 2007 julkaistu avoimen standardin todennusprotokolla (Hueniverse 2011). OAuthin toiminta poikkeaa sitä edeltäneistä protokollista sillä, että todennusvaiheessa käyttäjän ei tarvitse syöttää omia käyttäjätunnuksia palveluun. Sen sijaan OAuth toimii väliaikaisena avaimena, jonka avulla sovellukset voivat toimia käyttäjän puolesta. Lopullinen versio OAuth 1.0 protokollasta julkaistiin huhtikuussa 2010.

Vuonna 2012 julkaistu OAuth 2.0 ei ole nimestään huolimatta taaksepäin soveltuva 1.0:n kanssa. Sen perusidea tokenien vaihtamisesta on sama, mutta se käyttää hieman eri tekniikoita ja tapoja niiden siirtämiseen.

OAuth 1.0 oli hankala ottaa käyttöön verkkosivustoilla, sillä se vaati haastavan salausmekaniikan joka piti luoda omille sivuille. Tämä oli huomattavasti hankalampaa vanhaan käyttäjätunnuksiin perustuneeseen todennukseen. OAuth 2.0:ssa tämä ratkaistiin pakottamalla koko todennusprosessi tapahtumaan HTTPS-ympäristössä. Alkuperäinen OAuth rajoittui vain verkkosivustoihin. OAuth 2.0 lisäsi mahdollisuuden todentamiseen sovelluksissa ja multimedialaitteilla. OAuth 2.0 myös skaalautuu paremmin, sillä se mahdollistaa todennuksen ja API-kutsujen erittelyn. Näin palvelimien kuormitus jakautuu tasaisemmin (Parecki 2012).

2.7.1 Todentamisprosessi verkkosovelluksissa

OAuth 2.0:n todentaminen voidaan jakaa viiteen vaiheeseen. Sovellus aloittaa toiminnon siirtämällä käyttäjän todennuspaikkaan. Tänne mennessä mukana kulkevat tunnistautumiseen tarvittavat käyttäjätiedot sekä osoite, johon todennuspalvelin uudelleenohjaa käyttäjän, kun todennus on hyväksytty. Todennuspalvelin todentaa käyttäjän tiedot palveluntarjoajan tietojen kanssa ja päättää, voidaanko sovelluksen todennusyritys hyväksyä. Jos se hyväksytään, todennuspalvelin uudelleenohjaa käyttäjän takaisin aiemmin annettuun uudelleenohjausosoitteeseen. Osoitteeseen lisätään myös vahvistuskoodi. Kun sivulle on palattu, sovellus pyytää todennuspalvelimelta todennustunnusta. Tätä varten sovellus todentuu palveluun ja liittää mukaan aiemmin saadun vahvistukoodin sekä uudelleenohjausosoitteen. Todennuspalvelin vahvistaa koodin ja tarkastaa, että uudelleenohjausosoite vastaa alussa käytettyä osoitetta. Jos kaikki on kunnossa, todennuspalvelin palauttaa todennustunnuksen ja päivitystunnuksen (RFC 6749, 24).



Kuvio 5. OAuthin ja AdWordsin toiminta.

Edellä on näytetty todentaminen Googlen rajapintaan käyttämällä OAuth 2.0:aa(Kuvio 5). Todennuksessa käytetään Googlen omaa todennuspalvelinta, jolla myös käyttäjien tiedot ovat tallennettuna. Autentikoinnin jälkeen rajapintaan otetaan yhteyttä todennustunnuksen avulla. Google on määritellyt todennustunnuksen kestoksi 3600 sekuntia, minkä jälkeen se pitää uusia käyttämällä päivitystunnusta. Päivitystunnus ei vanhene koskaan (Google Developers 2013d).

3 TYÖN SUUNNITTELU

AdWords API:n käytön maksullisuuden takia oli erityisen tärkeää, että ohjelman rakenne suunniteltiin huolella. Näin ohjelma ei tee turhia kutsuja luokkiin, mikä lisäisi kustannuksia. Toisaalta hyvin suunniteltua ohjelmaa olisi helpompi toteuttaa ja tarvittaessa kehittää eteenpäin.

3.1 Esitarpeet toteutukseen

AdWordsin kaupallisuudesta johtuen API-avaimien saaminen AdWords API:n oli monimutkaisempaa kuin muihin Googlen rajapintoihin. Normaalisti API-avaimet saadaan heti rekisteröityessä käyttämään sitä. AdWordsin tapauksessa alkoi kuitenkin viikkojen selvitystyö.

Aluksi oli luotava uusi Gmail-tili, jota käytettiin My Client Center(MCC) tilin luontiin. Normaalisti sillä hallitaan muita AdWords tilejä, mutta tässä vaiheessa projektia tili kuitenkin luotiin vain API-avaimia varten. Kun tili oli vahvistettu, sinne oli annettava yrityksen toimiva verkko-osoite ja laskutustiedot. Ilman näitä API-avainta ei voitaisi hyväksyä. Tietojen laitta-

misen jälkeen siirryttiin itse avaimen pyytämiseen. Avainta pyydettäessä vastattiin seuraaviin kysymyksiin:

- Describe the uses of your API application or tool. Please be specific. For example, will your tool be used for account management, bid optimization, or something else?
- Who is or will be using your API application or tool? For example, colleagues in your company or advertisers or agencies to whom you are selling the tool.
- Please attach screenshots of your API application or tool. If the application or tool is yet to be developed, please provide relevant design documentation.
- Please provide a list of clients who will be using your API application or tool in an automated way.

Lomakkeen täyttämisen jälkeen se lähetettiin Googlelle tarkastettavaksi. Google arvioi tarkastamisen kestävän 5-6 viikkoa. Todellisuudessa siinä meni huomattavasti pidempään, sillä Google ei hyväksynyt ensimmäisiä vedoksia raporteista. Lopulta Google vaatikin nähdä itse sovelluksen toiminnassa, mikä jälleen hidasti tarkastusta.

Testitunnusten saamiseksi oli kuitenkin luotava uusi mainostajatili My Client Centerissä. Tilin luomisen jälkeen oli täytettävä lomake, jolla haettiin lupaa testaustiliin. Lomakkeeseen täytettiin Developer Tokenin neljä viimeistä merkkiä, My Client Centerin asiakas id ja testaustilin asiakas id. Testaustilin vahvistamiseen meni seitsemän päivää.

3.2 Sovelluksen rakenne

Sovelluksen rakenteen suunnittelu alkoi AdWords API:a tutkimalla. Ilmeni, että sillä voidaan ladata raportteja tiedostoon. API:n raporttipalvelu on lisäksi ilmainen, joten sitä kannatti tästäkin syystä käyttää mahdollisimman paljon.

Vaatimuksena oli, että uusille käyttäjille luodaan parametrisesti uudet tietokannat. Näin asiakkaita pystytään hallitsemaan erillään toisista. Tämä nopeuttaa tiedon hakua kannasta, koska haun pitää kulkea pienemmän rivimäärän läpi. Tämä puolestaan parantaa itse sovelluksen käyttöä, sillä odotusajat lyhenevät.

Näiden tietojen pohjalta alkoi ohjelman toimintajärjestyksen suunnittelu. Koska sovellusta piti pystyä tarvittaessa ajamaan komentolinjasta, oli tärkeää, että sovellus toimisi mahdollisimman automatisoidusti. Näin käyttäjän tarvitsee muistaa vähemmän komentoja. Ensimmäisessä suunnitteluvaiheessa tarvittavien komentojen määrä saatiin tiivistettyä yhteen, yrityksen nimeen. Yrityksen nimeä käytettäisiin hakemaan kannasta sitä vastaava refresh token sekä muut tarvittavat kentät raportin hakemiseen ja tallentamiseen.

Tämän jälkeen haettaisiin tiedot AdWordsista käyttäen rajapinnan raporttipalvelua. AdWords jakaa raportin kampanjoihin, mainosryhmiin ja mai-

noksiin, joten ne kaikki pitää hakea erikseen. Raportit tallennettaisiin joko XML-tiedostoon tai CSV-tiedostoon riippuen kummasta tieto tulee helpommin haettavaksi. Seuraavaksi tiedot tallennetaan kantaan ja lopuksi ladatut tiedostot poistetaan.

Aivan kaikkea ei kuitenkaan voinut liittää samaan prosessiin. Tämä johtui siitä, että AdWords vaatii käyttäjiltä OAuth2.0-autentikointia. Näin ensimmäiselle käyttökerralle piti suunnitella erillinen prosessi, joka annettaisiin asiakkaalle. Siinä asiakas täyttäisi tilinsä käyttäjätiedot ja ilmoittaisi yrityksen nimen. Näillä tiedoilla saataisiin luotua uusi tietokanta ja tarvittavat taulut tulevaa tiedonsyöttöä varten.

Vaihtoehtoisesti asiakkaita voisi pyytää liittymään Zoinedin My Client Centeriin, jolloin asiakkaan autentikointia ei tarvittaisi. Asiakkaan sähköpostiosoite riittäisi tilin id-numeron hakuun. Tällöin asiakkaan lisäys tietokantaan olisi Zoinedin ja pääskriptin työnä.

Ajojärjestyksen jälkeen oli hyvä luoda itseä varten myös alustava lista tarvittavista funktioista. Näin tarvittavat ominaisuudet ja funktiot eivät unohtuisi toteutusvaiheessa (Liite 4).

Koska AdWords päivittää tiedot kerran päivässä, oli skripti myös hyvä suunnitella ajettavaksi kerran päivässä. Näin tieto saadaan mahdollisimman uutena kantaan. Toisaalta sinne saadaan tasaiselta väliajalta olevaa tietoa, jota on helpompi jälkeenpäin käsitellä. Tiheämpi, lähes täysin reaaliaikainen haku olisi kuitenkin varsin turha, sillä osa tiedosta jäisi kuitenkin vanhaksi.

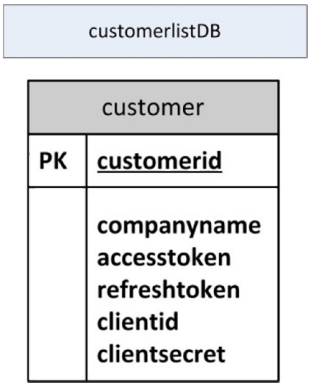
3.2.1 Rakenteen jatkokehitys

Kun AdWords-rajapinnan toiminta oli tullut hieman tutummaksi, tuli selväksi, että ohjelman rakenne ei ollut optimaalinen. Siksi suunnitteluvaiheeseen piti palata uudestaan. Alkuperäisen version yksi parametri tuntui lopulta turhalta. Toiminta suunniteltiin uudelleen niin, että skriptiä ajettaessa kaikkien kannassa olevien asiakkaiden tiedot haetaan automaattisesti ajoon mukaan. Näin ajon aloittamisesta tulee niin helppoa kuin mahdollista, eikä ohjelman ajajan tarvitse vahtia skriptin toimintaa sen ajon aikana. Tämä myös mahdollistaa ohjelman automatisoidun ajamisen päivittäin. Näin työntekijän ei välttämättä tarvitse käynnistää ohjelmaa manuaalisesti vaan ajo voidaan laittaa ajoitetuksi toiminnoksi tietokoneelle. Tämä varmistaa, että ohjelma ajetaan joka päivä, vaikka työntekijöitä ei olisikaan paikalla. Muutos ei kuitenkaan vaikuttaisi koodiin paljoakaan, joten kyseessä oli suuri hyöty pienellä haitalla.

3.3 Tietokannan rakenne

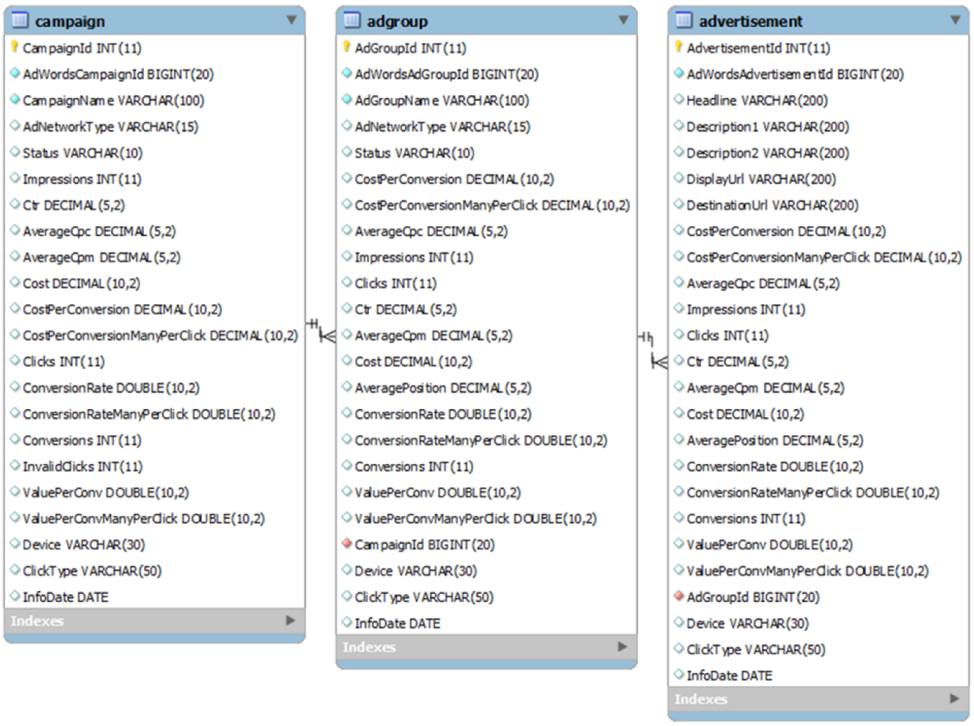
Kun ohjelman perusrakenne oli tiedossa, oli hyvä vaihe siirtyä tietokannan rakenteen suunnitteluun. Zoinedin toiveena oli, että sovellus luo automaattisesti jokaiselle asiakkaalle uuden kannan, kun uusi asiakas tuodaan mu-

kaan palveluun. Toisaalta sovelluksen toiminta vaatii asiakkaan tietoja toimiakseen. Paras ratkaisu oli luoda käyttäjien tiedoille oma kanta. Näin ne ovat erillään asiakkaiden kannoista, mutta helposti saatavilla ja muutettavissa yhdessä paikassa. Näiden tietojen kanta on lopulta vain yhden taulun kokoinen (Kuvio 6). Tässä vaiheessa suunnittelua tarkat vaatimukset kirjautumista varten eivät olleet vielä tiedossa. Nämä selviäisivät vasta siinä vaiheessa, kun ohjelmaan tuodaan OAuth mukaan.



Kuvio 6. Suunnitelma asiakaskannasta.

AdWordsin puolella kampanjaraportti on jaettu kolmeen osaan, kampanjoihin, mainosryhmiin ja mainoksiin (Kuvio 7). Näin myös tietokantaan tulee kolme taulua samoilla nimillä. Jotta tauluja voidaan käyttää helpommin keskenään, yhdistyvät ne toisiinsa viiteavaimilla. Mainostauluun tulee viiteavain mainosryhmistä. Vastaavasti mainosryhmiin tulee viiteavain kampanjoista. Vaikka useaan kenttään näyttää tulevan sama sarake, on ne järkevämpi toteuttaa kuvan mukaisella tavalla. Yhteisiin kenttiin tulee muista riippumatonta tietoa, pääosin numeroarvoja, joita ei kannata lähettää erilliseen tauluun. Toisaalta näin tehdessä tarvittavien hakujen ja lisäysten määrä laskee huomattavasti, sillä nyt tehdään yksi haku per kanta useamman sijaan.



Kuvio 7. Alkuperäinen suunnitelma tietokannan rakenteesta.

Tietokannassa pakollisia kenttiä ovat id- ja nimikentät. Tämä siksi, että näillä kentillä tietoa todennäköisimmin haetaan jatkossa kannasta. Toisaalta ilman näitä olisi mahdotonta yhdistää esimerkiksi hintaa mihinkään varsinaiseen kampanjaan. Muut kentät sisältävät lähinnä numeroarvoja eivätkä siksi ole pakollisia. Tämä johtuu suurimmaksi osaksi siitä, että AdWords ei anna aina kaikille kentille arvoja. Yleensä oletuksena arvoksi tulee nolla, mutta välillä AdWords jättää kentät tyhjiksi. Tästä johtuen myös tietokannassa kenttien tulee voida olla tarpeen mukaan tyhjiä.

Jokaisessa kolmesta taulusta ensimmäinen id on se id, jonka tietokanta luo automaattisesti uudelle tietoriville. Vastaavasti AdWords-alkuinen id on se id, jonka AdWords on luonut kohteelle. Nämä kaksi id:tä täytyy pitää erillään jo senkin takia, että jos tietoa halutaan lisätä joka päivältä kantaan, ei sama id voisi olla perusavaimena useaan kertaan.

Tekstiä vastaanottavien kenttien tyypiksi on määritelty VARCHAR. Lisäksi niille on määritelty sallittu merkkimäärä. Merkkimääriin jätettiin tarkoituksella ilmaa, jotta käyttäessä ei ilmenisi ikäviä yllätyksiä. Kokonaislukuja saavat kentät ovat määritelty INT-tyyppisiksi. Näihin mahtuu yli kahden miljardin kokoinen arvo, jonka sisälle eri kenttien lukujen koko pitäisi mahtua. Poikkeuksena ovat AdWordsin puolelta tulevat id-numerot, joissa ollaan yli neljän miljardin luvuissa. Näille kentille on annettu BIGINT-tyyppi, johon mahtuu yli 18 triljoonan luku. Rahaa ja prosentuaalisia arvoja käsittelevät kentät ovat määritelty DECIMAL-tyypillä. AdWords antaa näiden tiedot kahden desimaalin tarkkuudella, joten myös kannassa ne ovat kahden desimaalin tarkkuudella. DECIMAL on kuitenkin DOUBLE-tyyppiä tarkempi, joten näissä kentissä sitä on järkevämpi käyttää. DOUBLE-tyyppiä käytetään kentissä, joissa esitetään muun-

nosarvoja. AdWords antaa myös nämä kahden desimaalin tarkkuudella. Näissä DOUBLE on riittävän tarkka arvojen näyttämiseen.
Lisäksi jokaisessa taulussa on Date-kenttä, johon merkitään päivä, jolloin tieto on lisätty kantaan. Tämän avulla tietoa voidaan helposti järjestellä ja piirtää myöhemmin tarvittaessa kaavioihin.

Koodia kokeillessa ilmeni, että AdWordsin puolelta tulevat id-numerot ovat niin isoja numeroita, ettei normaalin INT-tyypin koko riitä. Siksi niiden kenttien tyypiksi tuli BIGINT.

4 TYÖN TOTEUTUS

4.1 Työkalut

Sopivien työkalujen valinta on erityisen tärkeää sovelluskehityksessä. Huonoilla ja vajailla ohjelmilla hidastetaan turhaan omaa luovuutta ja varsinaista ohjelmointia.

4.1.1 Ohjelmointityökalu

Vaikka PHP-kieltä voisi ohjelmoida millä tahansa tekstieditorilla, oli kuitenkin järkevämpää valita ohjelmointiin tarkoitettu työkalu. Näiden ohjelmien suurimpana etuna on mukaan liitetty kielikirjasto, joka osaa automaattisesti näyttää ja esitäyttää koodia tarpeen mukaan. Kirjastoa voi myös selata, ja sitä kautta valita sopivimman komennon eri vaiheisiin. Koska kirjasto on jo ohjelmassa itsessään, ei tarvitse turhaan pitää toista tiedostoa tai verkkosivustoa auki, jolta etsii sopivia käskyjä. Ohjelmointityökalujen laajuudessa ja monipuolisuudessa on kuitenkin huomattavia eroja. Jo suunnitteluvaiheessa tuli selväksi, että skripti tulisi koostumaan useammasta tiedostosta, joita yhdistellään yhteen keskustiedostoon. Tämän takia ohjelmassa tuli olla sisäinen tiedoston hallintamahdollisuus. Näin eri tiedostojen välillä voi vaihdella helposti ja yksinkertaisesti ja ohjelmiston kokonaiskuva pysyy paremmin muistissa.

Koska PHP on palvelinpäässä toimiva koodikieli, ei koodia voi ajaa niin yksinkertaisesti kuin esimerkiksi C-kieltä. Kaikki ajettavat tiedostot pitää ensin siirtää tiettyyn paikkaan, tässä tapauksessa omalle palvelimelle, ja ajaa sieltä. Ohjelmoitaessa on kuitenkin tärkeää, että itse luova vaihe ei katkea siksi, että tiedostoja pitää manuaalisesti siirtää omalta koneelta palvelimelle jokaisen tallennuksen jälkeen. Näinpä ohjelmointisovelluksen piti pystyä tekemään tämä vaihe automaattisesti.

Näiden omien vaatimuksien pohjalta etsin sopivaa työkalua. Vertailujen jälkeen ohjelmointityökaluksi tuli NetBeans, koska se täytti tarpeet parhaiten. Toisaalta olin käyttänyt ohjelmaa jo aiemmissa projekteissa, joten sen käyttöliittymä oli tuttu. Näin toteutusvaiheessa ei kulunut aikaa työkaluun tutustumiseen.

4.1.2 Tietokantatyökalu

Tietokantojen hallintaan käytettiin PHPMyAdminia. Hallintatyökalun valintaan ei ollut vaihtoehtoja, sillä hallinta tehtiin palvelimen päässä. Palvelimelle oli puolestaan asennettu tämä ohjelmisto. Tästä ei kuitenkaan ollut haittaa, sillä PHPMyAdminia oli käytetty jo aiemmissa projekteissa. Näin sen sisältö ja toiminta oli jo ennestään tiedossa. PHPMyAdmin on monipuolinen hallintatyökalu, jolla voidaan tehdä kaikki toiminnot, joita MySQL-kielestä löytyy. Se tarjoaa graafisen käyttöliittymän, jolla helposti nähdään luodut taulut ja niiden sisältö. Suurin osa PHPMyAdminin parissa käytetystä ajasta kului tauluja tutkiessa. Siksi tietojen näyttö graafisesti oli työkalun tärkein ominaisuus. Varsinainen tietokantojen hallinta tehtiin kuitenkin pääosin koodillisesti oman sovelluksen kautta.

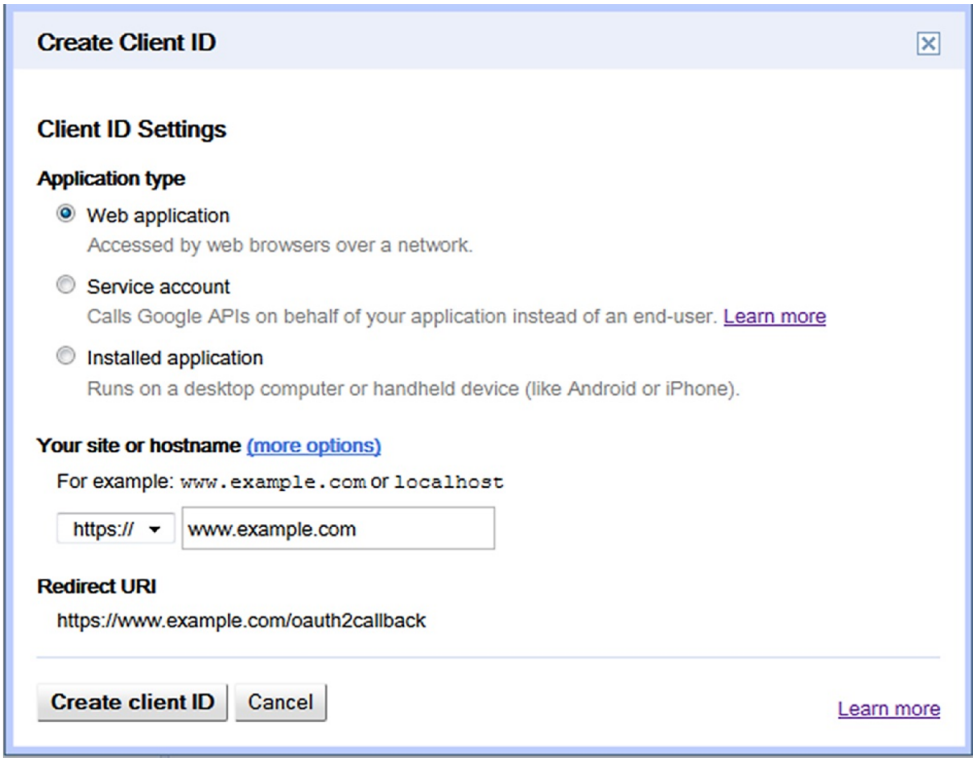
4.2 AdWords APIn käyttöönotto

Aluksi tehtiin lyhyt testiskripti. Skriptin päätarkoituksena oli tarkastaa, että testitunnukset sekä kehittäjän tunnisteet toimivat. Samalla saatiin hyvä katsaus rajapinnan toimintaan. Ensimmäisessä skriptissä haettiin kampanjoiden tietoja (Liite 5). Kun skripti toimi odotetusti, vastaukseksi saatiin, että tililtä ei löytynyt yhtäkään kampanjaa. Ohjelma toimi oikein, sillä tiliin ei ollut vielä lisätty kampanjoita.

Testitilille piti luoda uusia kampanjoita, mainosryhmiä ja mainoksia, jotta tililtä voidaan hakea tietoa. Testitunnuksilla oli kuitenkin vain lukuoikeus, joten skripti ei toiminut odotetusti. Sovellus ei kuitenkaan kertonut suoraan ongelmasta, vaan ilmoitti pelkästään, että koodia ajaessa on tapahtunut virhe. Apua haettiin Googlen apuryhmästä, ja ongelman ratkaisuun kului useampi viikko aikaa. Oikeuksien muuttamisen jälkeen kaikki sujui jälleen normaalisti ja tilille saatiin lisättyä tarvittavat tiedot.

4.3 Sovellus id:n luonti

Jotta ohjelman todennus saadaan toimimaan, täytyy sille luoda sovellus id. Tunnukset luodaan Googlen rajapintakeskuksessa osoitteessa code.google.com/apis. Kun tällä sivustolla painaa create client id -näppäintä, aukeaa alla oleva ruutu (Kuva 1).



Kuva 1. Sovellus ID:n luonti OAuthia varten.

Koska opinnäytetyössä tehtävä ohjelma toimii internetissä, myös tässä valitaan ohjelmatyypiksi verkkosovellus. Tällöin kirjautumistiedot siirtyvät automaattisesti URL-rivillä sovelluksen sivustolle, josta ne voidaan kerätä talteen koodilla. Näin asiakkaan ei tarvitse käsin kopioida tietoja, hänen tarvitsee vain hyväksyä todentuminen. Ainoa pakollinen tieto sovellus id:n saamiseksi on sivuston osoite, josta todentumiskutsu tehdään ja jonne jälkeenpäin palataan. Google generoi automaattisesti paluuosoitteen. More options -linkkiä painettaessa tälle kuitenkin voidaan määrittää haluttu kohde. Kun create client id -näppäintä painetaan, Google luo todentamiseen tarvittavat tiedot.

4.4 Pääohjelma

Pääpainona ohjelmassa on tietojen nouto AdWordsista ja niiden tallennus tietokantaan. Ohjelma ajetaan yhdellä PHP-tiedostolla. Selvyyden vuoksi eri funktiot on kuitenkin eritelty yhdeksään erilliseen PHP-tiedostoon. Tiedostoissa ovat ohjelman eri ajovaiheet. Lisäksi yhdistäminen tietokantoihin on laitettu erillisiin tiedostoihin. Tiedostot tuodaan päätiedostoon require_once-käskyllä. Lisäksi AdWordsin ohjelmistokirjasto linkitetään ohjelmaan require_once-käskyllä. Näin ohjelma osaa käyttää ohjelmistokirjaston sisällä olevia toimintoja.

Ohjelmasta tehtiin täysin automatisoitu. Tämä tarkoittaa sitä, että Zoinedin tarvitsee vain käynnistää skripti ja muu tapahtuu automaattisesti. Tämä saatiin aikaan noutamalla ensin jokainen käyttäjä käyttäjäkannasta ja käyttämällä näistä saatua tietoa kehyksenä muille toiminnoille. Kannasta haetaan kaikkien asiakkaiden tiedot talteen, sillä niitä tarvitaan myöhemmissä vaiheissa. Loppuskriptiä ajetaan niin kauan kuin tietokannasta löytyy läpi-

käymättömiä asiakkaita. Näin skripti toimii juuri niin kauan kuin sille on tarvetta.

4.4.1 Todennustunnuksen uusiminen

Koska OAuth 2.0 todennustunnukset vanhentuvat tunnissa, täytyy ne päivittää uudelleen ennen AdWordsiin yhdistämistä. Tätä varten muistista noudetaan asiakaskohtaiset tiedot. Todennustunnuksen uusimiseen tarvitaan OAuthin puolelta annettu sovelluksen id-tunnus, sovelluksen salaustunnus, vanha todennustunnus ja päivitystunnus. Lisäksi asiakkaan id-numero ja yrityksen nimi tallennetaan muuttujiin myöhempää käyttöä varten.

Muuttujiin tallennuksen jälkeen alustetaan uusi AdWords-käyttäjä ohjelmaan. Alustus on parasta tehdä jo tässä vaiheessa, sillä sitä tarvitaan myös tunnuksien uusimisessa. Toisaalta Google painottaa hyvissä toimintamalleissaan (Google 2013b), että toimintoja pitäisi ryhmittää mahdollisimman tehokkaasti. Myös käyttäjän alustus kuului tähän ryhmään. Nyt, koska käyttäjä luodaan jo näin aikaisessa vaiheessa, sitä voidaan käyttää useampaan kertaan ajon aikana. Alustus ei kuitenkaan vaikuta 10 000 operaation päivärajoitukseen, joten alustuksen uusiminen jokaisen uuden asiakkaan kohdalla oli järkevä idea.

Tietokannasta haetut vanhat käyttäjätiedot liitetään käyttäjään. Näitä tietoja käyttäen päivitetään käyttäjän todennustunnus. Uusiminen tapahtuu käyttäjäkirjaston ansiosta yhdellä rivillä. Päivitetyt tiedot noudetaan takaisin käyttöön ja todennustunnus tallennetaan tietokantaan. Lopuksi uudet OAuth tiedot tallennetaan käyttäjälle. Näillä tiedoilla päästään AdWordsin palvelimille.

4.4.2 Raporttien lataaminen

Ohjelmaan tuli kolme eri funktiota, jotka hakevat AdWordsista raportteja. Yhdellä haetaan kampanjan, toisella mainosryhmien ja kolmannella mainoksien tehokkuustiedot. Hakuprosessi jaettiin kolmeen osaan toimintojen selkiyttämiseksi. Raportit piti joka tapauksessa ladata eri tiedostoihin, joten oli parempi myös eritellä lataukset toisistaan. Näin toimintoja pystyisi jatkossa tutkimaan ja muokkaamaan nopeammin.
Haussa käytetään apuna AdWords rajapinnan raportointipalvelua. Hakukäskyt on tehty AWQL –kielellä (Kuvio 8).

```
 $reportQuery = 'SELECT CampaignId, CampaignName, AdNet-
workType1, Status, Impressions, Ctr, AverageCpc, Aver-
ageCpm, Cost, CostPerConversion, CostPerConversionManyPer-
Click, Clicks, ConversionRate, ConversionRateManyPerClick,
Conversions, InvalidClicks, InvalidClickRate, Value-
PerConv, ValuePerConvManyPerClick, Device, ClickType FROM
CAMPAIGN_PERFORMANCE_REPORT DURING ' . $dateRange;
```

Kuvio 8. Kampanjaraportin tietojen hakuun käytetty AWQL-kysely.

Edellä on näytetty esimerkkinä kampanjan tietojen hakuun tarvittava kysely. Kyselyssä määritellään kaikki halutut kentät sekä aikaväli, jolta tiedot haetaan. Koska skripti on suunniteltu ajettavaksi kerran päivässä, haetaan myös raporttiin vain edellisen päivän tiedot. Tämä myös helpottaa jatkossa kaavioiden luontia, sillä nyt tiedot ovat jo valmiiksi eritelty päivittäin. Edellä olevassa kyselyssä aikaväli on asetettu erilliseen muuttujaan lauseen selkiyttämiseksi. Kyselyssä pyydettyjen kenttien järjestyksellä ei ole merkitystä. Järjestelin kuitenkin tässä projektissa alkupäähän kaikki kampanjaa kuvaavat kentät ja loppuun numeroarvoja antavat kentät. Viimeiset kaksi kenttää, eli Device ja ClickType on lisätty jälkeenpäin Googlen minimivaatimusten muututtua. Mainosryhmissä ja mainoksissa kyselyt eroavat lähinnä tarvituissa kentissä. Kyselyn runko pysyy kuitenkin pääosin samana.

Raportit ladataan omiin XML-tiedostoihin. Tiedostot seuraavat WSDL-rakennetta. Näissä on ensin listattuna abstrakti osa, eli sarakkeiden nimet. Abstraktin jälkeen tulee konkreettinen osa, eli rivit. Riveissä on kuvattu erikseen jokainen AdWordsista saatu kampanja, mainosryhmä ja mainos. Näille myös näytetään edelliseltä päivältä saatu aktiivisuustieto. WSDL-rakenteen ansiosta XML-tiedosto on helpompi lukea, sillä lukijalle kerrotaan heti alkuun, mitä mikäkin sarake tarkoittaa ja listaa voidaan käyttää referenssinä rivejä lukiessa. Toisaalta rakenne myös helpottaa tiedon prosessointia koodissa, sillä tällainen eritelty tyyli on helpompi muuntaa toistettaviksi koodinpätkiksi (Liite 6).

4.4.3 Tietojen siirto tietokantaan

Kun XML-tiedostot on ladattu talteen, voi ohjelma aloittaa tietojen siirron tietokantaan. Luku suoritetaan tiedosto kerrallaan järjestyksessä kampanjat, mainosryhmät ja lopuksi mainokset. Näin varmistetaan, että alemman tason tiedoille on jo olemassa ylempien tasojen id-numerot, jotta viiteavaimet voivat liittyä oikein.

Tiedostojen lukemiseen käytetään PHP:n XMLReaderia. Ohjelmassa käytetään XMLReaderia, sillä se on SAX-muotoinen tiedon käsittelyrajapinta. SAX toimii eventien aikana, joka sallii suurempien tiedostojen lukemisen. Tämä johtuu siitä, että tällainen toiminta vaatii vähemmän muistia. Muistin optimointi onkin tässä vaiheessa tarpeen, sillä suuremmilla yrityksillä etenkin mainoksien määrä voi olla todella suuri. Esimerkiksi toinen yleinen lukutyyppi DOM olisi voinut kuluttaa muistit loppuun ennen kuin tiedosto on luettu kokonaan. Tämä joko hidastaisi ohjelman ajon matelunopeuteen tai voisi jopa keskeyttää koko ohjelman ajon. Suurimpana haittana SAX-lukijassa on, että jos jokin tieto halutaan saada uudelleen, se pitää myös lukea uudelleen tiedostosta. Tätä haittaa ei kuitenkaan ole tässä ohjelmassa, sillä tiedot luetaan joka tapauksessa vain kerran läpi.

Tiedoston läpiluku suoritetaan while-loopin sisällä. While tarkastaa, että XML-tiedostossa on vielä uusia tietorivejä, ja ajaa ohjelmaa uudelleen, kunnes rivit loppuvat. Tiedoston lukuvaiheessa WSDL-muotoisen XML-tiedoston tehokkuus on huomattava. Koska sarakkeet ja rivit on eroteltu

toisistaan, voidaan ne myös hakea tiedostosta erillään. Näin koodaus helpottuu, kunhan lukurakenteen ymmärtää. Tässä ohjelmassa luku on tehty kahdella erillisellä if-lauseella.

```
if ($reader->nodeType == XMLREADER::ELEMENT &&
$reader->localName == 'column') {
            //Sarakkeet haetaan xml:stä niiden nimiatri-
            buutin mukaan.
            $columnName = $reader->getAttribute("name");
            //Nimet siirretään arraylistaan.
            $columnNames[] = $columnName;
        }
```

Kuvio 9. Sarakkeiden lukeminen XML tiedostosta.

Ensimmäisellä if-lauseella haetaan tiedoston sarakkeet niiden nimiatribuutin mukaan (Kuvio 9). Tiedostossa (Liite 6) on sarakkeille annettu myös display-atribuutti, mutta se on lähinnä tarkoitettu ohjelmille, jotka näyttävät tiedon suoraan ruudulla. Sitä ei siis tarvita tässä ohjelmassa. Haetut tiedot tallennetaan array-listaan. Listaa käytetään seuraavassa if-lauseessa apuna rivien haussa.

```
if ($reader->nodeType == XMLREADER::ELEMENT && $reader-
>localName == 'row') {
            $query = "INSERT INTO advertisement (
            Tietokannan kentät) VALUES (null, ";

            foreach ($columnNames as $columnName) {
                $val =
               $reader->getAttribute($columnName);
                $query .= "'" . $val . "', ";
}
            $queryfix = str_replace('%', '', $query);
            $queryfix .= "CURDATE())";
            $insert = mysql_query($conn, $queryfix);
            if (!$insert) {
                die('Error: ' . mysql_error() . "Errno" .
               mysql_errno());
            }
        }
```

Kuvio 10. Rivien lukeminen XML-tiedostosta ja tietokantaan vienti.

Seuraavalla if-lauseella haetaan tiedoston rivit (Kuvio 10). Ajon alkuun on laitettu query-muuttuja, jossa on alkuosa tarvittavasta MySQL-kyselystä. Siihen on listattu kaikki kentät, tässä tapauksessa mainostaulun kentät. Muuttujan määrittäminen tässä varmistaa sen, että sille ei jää edellisen rivin tietoja. Seuraavaksi riveiltä haetaan tiedot foreach-lauseella sarakkeittain ulos. Apuna tässä käytetään edellisestä if-lauseesta saatua array-listaa. Saadut tietopätkät lisätään yksitellen query-muuttujan perään, millä saadaan rakennettua täydellinen MySQL-kysely. Kun kysely on pääosin valmis, poistetaan siitä prosenttimerkit pois. Numerokentät eivät tunnista prosenttimerkkiä, joten se on pakko jättää pois kannasta. Se lisätään jälkeenpäin koodillisesti, jos kyseisiä kenttiä käytetään tiedon näyttämiseen. Vii-

meiseksi lauseeseen lisätään vielä CURDATE(), joka antaa tiedon keräyspäivän. Lopuksi tiedot viedään kantaan ja uusi kierros alkaa.

4.4.4 Tiedostojen poistaminen

Kun koko tiedonsiirtoprosessi on suoritettu, kannattaa väliaikaisiksi tarkoitetut XML-tiedostot poistaa. Näin samaa tietoa ei turhaan säilytetä kahdessa paikassa. Tällä myös tarkistetaan, että raportin lataus toimii varmasti oikein. Lataustoiminto saattaisi yksinkertaisesti lisätä uuden tiedon vanhan perään, jolloin tietokantaan vietäisiin samaa tietoa useaan kertaan. Toisaalta lataus saattaisi hajota, jos se ei pystyisi korvaamaan vanhaa tiedostoa. Koska poisto kehitettiin skriptiin jo aikaisessa vaiheessa, ei mahdollisia latausongelmia ilmentynyt.

```
function removeFiles(){
    //Alusta arraylista, sisällä poistettavien tiedostojen nimet.
    $file = array("campaign.xml","adgroup.xml","advertisement.xml");


    //Tiedostojen määrän ajan
    for($i = 0; $i<3; $i++){

        //Sulje tiedosto
        fclose($file[$i]);
        //Poista tiedosto
        unlink($file[$i]);

        echo "Tiedosto ". $file[$i]. " Poistettu.\n";
    }
}
```

Kuvio 11. Tiedostojen poistaminen PHP:lla

Poistettavat tiedostot on listattuna array-listaan (Kuvio 11). Tätä listaa käytetään seuraavaksi tulevan for loopin sisällä. For loop kierrättää poistoskriptin kolmeen kertaan, eli niin kauan, kun array-listassa on tiedostoja jäljellä. Jotta poistaminen onnistuu varmasti, tiedostot suljetaan fclose-komennolla. Näin tiedosto ei ole enää käytössä ja mahdollisesti linkitettynä muuttujiin. Riski tähän on pieni ohjelmassa jo muutenkin, mutta tämän kohdalla on parempi pistää sulkeminen varman päälle, kuin mahdollisesti rikkoa koko skripti.

4.5 Autentikoituminen

Toinen pääosio ohjelmassa on asiakkaan autentikoituminen. Tällä yhdistetään käyttäjän AdWords-tunnukset sovelluksen tietokantaan ja sallitaan niiden käyttäminen automaattisesti uudelleen. Todentamisessa käytetään OAuth 2.0-todennusprotokollaa joka mahdollistaa kirjautumistunnusten käytön ilman, että varsinaiset tunnukset ovat kuitenkaan sovelluksen tai muiden ihmisten nähtävillä. Todentaminen on ainoa hetki, jossa asiakas on kontaktissa tiedonkeruuohjelmaan.

4.5.1 Asiakkaan lähettäminen todennusosoitteeseen

Todentamisen ajaminen aloitetaan esimerkiksi Zoinedin etusivulle sijoitetulla linkillä. Linkki johtaa alla olevaan skriptiin, joka suorittaa todentamisen ensimmäisen vaiheen (Kuvio 12).

```
<?php
require_once 'ad-
words_api/src/Google/Api/Ads/AdWords/Lib/AdWordsUser.php';

//Alusta muuttujat
$clientId = "idnumero";
$clientSecret = "secret";
$callbackUrl = "palautusosoite";

// Alusta uusi AdWords käyttäjä ja aseta tarvittavat oAuth
tiedot.
$user = new AdWordsUser();
$user->SetOAuth2Info(array(
    "client_id" => $clientId,
    "client_secret" => $clientSecret
));

// Luo todennusosoite tiedoista.
$authUrl = $user->GetOAuth2AuthorizationUrl($callbackUrl,
true);
 //Siirry osoitteeseen
header("Location: $authUrl");
?>
```

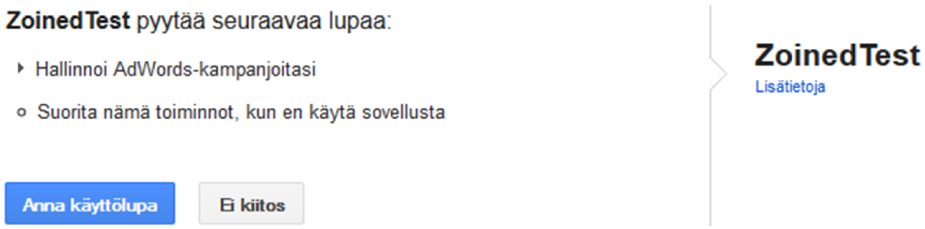
Kuvio 12. Ajon aloittamiseen tarvittava koodi

Skriptissä käytetään apuna AdWordsin ohjelmistokirjastoa. Kirjastossa tulee mukana myös OAuthin käyttöä auttavia ominaisuuksia, jotka suoraviivaistavat ohjelmointia. Kuviossa muuttujien arvot on piilotettu tietoturvan vuoksi. Tiedot näihin saadaan Googlen API-sivustolla tehdystä OAuth-tunnuksesta. Muuttujien määrittämisen jälkeen alustetaan uusi AdWordsUser-metodi. Tälle asetetaan aiemmin määritetty client id ja client secret. Tietoja tarvitaan autentikointiin, sillä näillä yhdistetään käyttäjän tiedot tähän skriptiin.

Seuraavaksi luodaan osoite, johon asiakas siirretään. Osoite on kuin pätkä koodia, johon on liitetty mukaan edellä annetut client id ja client secret. Lisäksi siihen liitetään vielä callback url, johon asiakas palaa autentikoitumisen jälkeen. Sulkeiden sisällä oleva true parametri määrittää, että sovellus toimii offline tilassa. Näin Google antaa myös päivitystunnuksen, jota tarvitaan varsinaista skriptiä ajettaessa. Jos parametri olisi false, sitä ei annettaisi, eikä ohjelma siten toimisi. Lopulta osoitteesta luodaan ajokäsky, jolla itse siirtyminen tapahtuu.

4.5.2 Autentikoitumisen hyväksyminen

Sovelluksen päässä luotu linkki johtaa Googlen todennussivustolle. Jos asiakas ei ole kirjautunut googlen tunnuksilla, tulee ensin kirjautumissivu. Jos kirjautuminen on jo tehty ennestään, tulee seuraava sivu automaattisesti (Kuva 2).



Kuva 2. Googlen todennussivusto.

Sivu näyttää lupaa pyytävän ohjelman nimen ja listan toiminnoista, joiden käyttö sallitaan hyväksyttäessä. Lisätietoja-linkkiä painettaessa nähdään kehittäjän sähköpostiosoite sekä osoite, johon todennuspalvelu palauttaa asiakkaan. Jos asiakas antaa käyttöluvan, generoi Googlen autentikointipalvelin asiakkaalle todennustunnuksen jonka avulla sovellus pääsee tietoihin käsiksi. Jos asiakas painaa ”ei kiitos” -nappulaa, pysähtyy todentaminen siihen. Kieltäytymistä ei kuitenkaan tallenneta muistiin, eli todentaminen voidaan suorittaa yöhemmin.

4.5.3 Todennustietojen nouto ohjelmaan

Hyväksytyn ja läpi menneen todentamisen jälkeen Googlen sivusto palauttaa asiakkaan aiemmin annettuun paluuosoitteeseen. Paluuosoitteessa käytetään jälleen apuna AdWordsin ohjelmistokirjastoa. Lisäksi alkuun on luotu muuttujat client id:lle, secretille ja palautusosoitteelle. Ensin koodissa alustetaan uusi käyttäjä, jolle asetetaan client id ja secret. Palauttaessaan käyttäjän takaisin sovelluksen sivulle, Google lisää URLiin mukaan tunnistautumiskoodin, jonka avulla päästään käsiksi kirjautumiseen vaadittaviin tietoihin.

```
$authCode = $_REQUEST["code"];
$user->GetOAuth2AccessToken($authCode, $callbackUrl);
    $oauthInfo = $user->GetOAuth2Info();
    $_SESSION["clientid"] = $oauthInfo["client_id"];
    $_SESSION["clientsecret"] = $oauthInfo["client_secret"];
    $_SESSION["accesstoken"] = $oauthInfo["access_token"];
    $_SESSION["refreshtoken"] = $oauthInfo["refresh_token"];
```

Kuvio 13. Todennuskoodin muunto kirjautumistietoihin.

Todennuskoodi saadaan erilleen URL-rivistä $_REQUEST[”code”]-komennolla (Kuvio 13). Tällä haetaan osoitteesta pätkä, joka alkaa code= -pätkän jälkeen ja loppuu joko itse osoitteen loppuun tai kysymysmerkkiin,

josta alkaa uusi parametri. Koodia ja paluuosoitetta käyttämällä saadaan haettua käyttäjän todennustunnus ja päivitystunnus OAuth-rajapinnasta. Lisäksi varmuudeksi noudetaan client id ja client secret, jotta tiedot ovat täydellä varmuudella yhtenäisiä. Haetut tiedot tallennetaan tässä vaiheessa sessioon, sillä niitä ei voida vielä tässä vaiheessa tallentaa tietokantaan. Paluuosoitteessa kysytään asiakkaan yrityksen nimeä. Tätä tietoa käytetään tietokannan luonnissa. Kun tieto on annettu ja asiakas painaa lähetänäppäintä, siirretään kaikki tieto seuraavaan PHP-tiedostoon.

4.5.4 Tietokannan ja taulujen luonti

Ensimmäiseksi edellisiltä sivuilta kerätyt tiedot tallennetaan käyttäjäkantaan. Kantaan tulee id-numero, yrityksen nimi, client id, client secret, todennustunnus ja päivitystunnus. Kantaan viennin jälkeen luodaan uusi tietokanta käyttäjälle yrityksen nimen mukaan. Lisäys suoritetaan create database -komennolla, jolla kanta luodaan. Kannan luomisen jälkeen luodaan taulut kampanjoille, mainosryhmille ja mainoksille (Liite 7). SQL-lauseen perään lisätään rivi DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci. Tällä varmistetaan, että mahdolliset skandinaaviset kirjaimet menevät normaalisti kantaan.

4.6 Jatkokehitys

Ohjelman nykyisellä versiolla haetaan vain kampanjoiden, mainosryhmien ja mainosten tiedot talteen. Google kuitenkin tarjoaa esimerkiksi käyttäjän sijaintiin ja toimintatapoihin perustuvaa tietoa, josta voi olla suurta hyötyä grafiikoita luotaessa. Näitä eri tietoja kerättäessä voidaan luoda tarkempi kuva käyttäjästä ja siten myös Zoinedin asiakkaat voisivat kiinnostua palvelun käytöstä enemmän.

Toisaalta sovellus on nyt melko staattinen. Esimerkiksi tietokannan kolumneja ja sisältöä joutuu päivittämään manuaalisesti. Tällaiset muutokset pitäisi pystyä jatkossa tekemään automaattisesti sitä mukaa, kun Google lisää tai poistaa toimintoja palvelustaan.

Jatkovaiheessa myös ohjelman optimointi voi olla tarpeellista, jos sille tulee erittäin suuret käyttäjämäärät. Nyt skriptiä on testattu vain pienillä määrillä, joilla mahdollista suurempaa stressiä ei pääse kokeilemaan.

5 YHTEENVETO

Opinnäytetyössä toteutettiin yhteys Googlen AdWords-palvelun ja Zoinedin MySQL-tietokannan välille. Opinnäytetyön raportissa keskitytään pääosin itse toteutuneen ohjelman toimintojen kuvaamiseen ja koodiesimerkkeihin. Ohjelmoitaessa käytettiin PHP-kieltä, mutta saman toiminnallisuuden saa aikaiseksi myös muilla kielillä, jos käyttää Googlen luomaa ohjelmistokirjastoa, joka on luotu AdWords-rajapinnan päälle. Esimerkki-

en ideana on auttaa lukijaa pääsemään helpommin alkuun AdWords APIn kanssa, sillä haastavin vaihe toteutuksessa oli juuri rajapinnan saaminen toimintaan ja sen sisällön ymmärtäminen. Hankala ymmärtäminen johtui pääosin Googlen tavasta opastaa rajapintansa käyttämistä koodiesimerkein. Esimerkit kuitenkin rakentuvat todella monimutkaisista ja syvälle esimerkkikirjaston syövereihin pureutuvista koodilinkityksistä, joiden takia koodin lukemisesta tuli haastavaa. Toisaalta AdWords APIn sivusto oli vielä toteutusvaiheessa puolitiessään, eikä kaikkea tietoa ollut vielä lisätty sivuille. Yhteenvetoa kirjoitettaessa tilanne on jo hieman parantunut, mutta se ei ole täydellisellä tasolla.

Esimerkkejä tai ohjeita ei ollut myöskään paljoa saatavilla muualla verkossa. AdWordsin kehitysympäristö on niin suljettu, että harrastelijaohjelmoijilla ei ole suurta halua tutustua siihen. Jo testiavaimien saanti on huomattavasti suurempi prosessi AdWordsissa, kuin esimerkiksi muissa Googlen palveluissa. Näinpä suurin osa AdWordsille ohjelmoitavista projekteista on yhtiöiden sisäisiä ja tietotaitokin jää yhtiöiden sisälle.

Hyvin tehty toiminnallisuusrakenteen suunnittelu heti projektin alussa oli tärkeässä osassa sovellusta ohjelmoitaessa. Tämä piti varsinaisen toteutusvaiheen ajantasalla ja aina oli tiedossa mitä pitää tehdä seuraavaksi. Ohjelman toteutus veikin lopulta vain noin kolme kuukautta tehokkaan suunnitelman ansiosta.

Opinnäytetyö oli tärkeä osa omaa oppimisprosessia. PHP:sta oli jo aiempaa osaamista muutaman verkkosivustoprojektin kautta, mutta tässä projektissa siihen piti syventyä huomattavasti enemmän. Erityisesti tehokas toimintojen automatisointi oli hyvää uutta tietoa, jota pystyy käyttämään myös jatkossa muissa projekteissa. Lisäksi ohjelmassa käytetyt XML ominaisuudet ovat yleisesti käytettyjä myös muualla, joten niiden oppiminen oli kätevää tietoa tulevaisuutta varten.

OAuth 2.0:n oppiminen oli myös hyödyllistä, sillä se on tällä hetkellä varsin suosittu tapa hallita kirjautumista. Koska OAuth sisältää ainakin pääosin saman rakenteen sivustosta riippumatta, voi nyt opittuja tietoja soveltaa myös niissä jatkossa.

Ohjelmointioppien lisäksi opinnäytetyössä pystyi kehittämään projektityötaitoja. Omien aikataulujen luonti ja niissä pysyminen oli hyvää oppia tulevaisuutta varten. Harmittavasti projekti venyi suunniteltua pidemmäksi Googlen tarkastuksista johtuen.

LÄHTEET

3Q: Verkkomainonnan merkitys brändin rakentajana ja myynnin tekijänä vahvistuu. 2012. IAB Finland. Viitattu 11.1.2013. http://www.iab.fi

Berman, J. 2008 Ruby: The Programming Language. Sudbury: Jones and Bartlett Publishers

Best Practices. 2013b. Google. Viitattu 17.4.2013. https://developers.google.com

Donaldson, T. Python: Visual QuickStart Guide. 2008. Berkeley, California: Peachpit Press

Download Formats. 2012c. Google Developers. Google. Viitattu 28.1.2013. https://developers.google.com

FAQ. 2012b. Google Developers. Google. Viitattu 21.1.2013. https://developers.google.com

Geddes, B. 2012. Advanced Google AdWords Second Edition. Indianapolis: John Wiley & Sons.

Gosling, J. 2000. The Java Language Specification. California: First Printing.

Historical trend. 2013. Usage statistics and market share of PHP for websites. W3Techs. Viitattu 17.1.2013. http://w3techs.com

Hudson,P. 2007. PHP in a Nutshell. Sebastopol: O'Reilly Media, Inc.

Installing Connector/J from a Binary Distribution. 2013b. MySQL. Viitattu 3.6.2013. http://dev.mysql.com

Jacobson, H. 2011. AdWords for Dummies. Hoboken: Wiley Publishing, Inc.

Laatupisteiden tarkastaminen ja tulkitseminen. 2012a. Google. Viitattu 15.1.2013. http://support.google.com

Mojica, J. 2003. C#:Web Development With Asp.Net. Berkeley, California: Peachpit Press

Nykänen, O. 2003 XML 10 kohdan tiivistelmänä. Viitattu 9.4.2013. http://www.w3c.tut.fi

Overview of the AdWords API. 2012a. Google Developers. Google. Viitattu 21.1.2013. https://developers.google.com

Parecki, A. 2012. Differences from OAuth 1.0. Viitattu 8.4.2013. http://aaronparecki.com

Powell, R. Richard, W. 2002. C Sharp and the .NET Framework: The C++ Perspective. Yhdysvallat: Sams Publishing

Quadrennial events to help ad market grow in 2012 despite economic troubles. 2011. The Zenith Optimedia Blog. Viitattu. 11.1.2013. http://zenithoptimedia.blogspot.fi

Relaatiotietokanta. 2006. Jari Sarjan materiaalia verkossa. Jari Sarja. Viitattu 22.1.2013. http://www.verkkopedagogi.net

RFC 6749. The OAuth 2.0 Authorization Framework. Obtaining Authorization. 2012. Internet Engineering Task Force. Viitattu 9.4.2013. http://tools.ietf.org

Schwartz, R. Foy, B. Phoenix, T. 2011. Learning Perl, Sixth Edition. Sebastopol, California: O'Reilly Media, Inc.

Stucky, M. 2001. MySQL: Building User Interfaces. USA: New Riders Publishing.

The OAuth 1.0 Guide. 2011. Hueniverse. Viitattu 8.4.2013. http://hueniverse.com/OAuth/guide/

Using MySQL With Java. 2013a. MySQL. Viitattu 3.6.2013. http://dev.mysql.com

Using OAuth 2.0 to Access Google APIs. 2013d. Google Developers. Viitattu 9.4.2013. https://developers.google.com

What is Perl? Kirrily, R. Perldoc. Viitattu 5.6.2013. http://perldoc.perl.org

AdWords kampanjatietojen haku array taulukosta

```
    // Display results.
    if (isset($page->entries)) {
      foreach ($page->entries as $campaign) {
        printf("Campaign with name '%s' and id '%s' had the fol-
lowing stats "
            . "during the last week:\n", $campaign->name, $cam-
paign->id);
        printf("  Impressions: %d\n", $campaign->campaignStats-
>impressions);
        printf("  Clicks: %d\n", $campaign->campaignStats-
>clicks);
        printf("  Cost: $%.2f\n", $campaign->campaignStats-
>cost->microAmount
            / AdWordsConstants::MICROS_PER_DOLLAR);
        printf("  CTR: %.2f%%\n", $campaign->campaignStats->ctr
* 100);
      }
    } else {
      print "No matching campaigns were found.\n";
    }
```

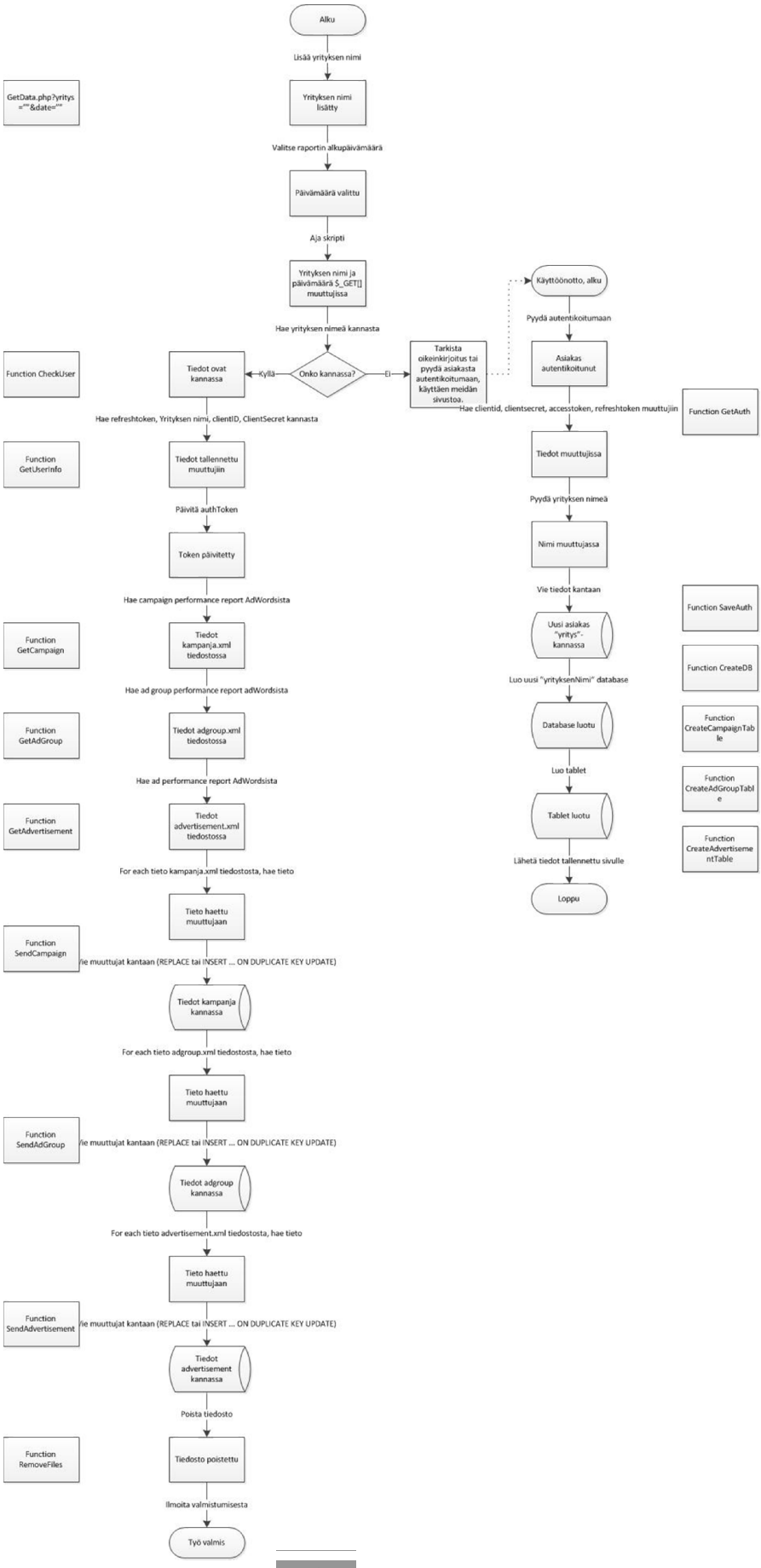
Esimerkki raportin hausta ilman AWQL:ää

```
function DownloadCriteriaReportExample(AdWordsUser $user,
$filePath) {
  // Load the service, so that the required classes are availa-
ble.
  $user->LoadService('ReportDefinitionService',
ADWORDS_VERSION);

  // Create selector.
  $selector = new Selector();
  $selector->fields = array('CampaignId', 'AdGroupId', 'Id',
'Criteria',
      'CriteriaType', 'Impressions', 'Clicks', 'Cost');

  // Filter out deleted criteria.
  $selector->predicates[] = new Predicate('Status', 'NOT_IN',
array('DELETED'));

  // Create report definition.
  $reportDefinition = new ReportDefinition();
  $reportDefinition->selector = $selector;
  $reportDefinition->reportName = 'Criteria performance report
#' . uniqid();
  $reportDefinition->dateRangeType = 'LAST_7_DAYS';
  $reportDefinition->reportType = 'CRITERIA_PERFORMANCE_REPORT';
  $reportDefinition->downloadFormat = 'CSV';

  // Exclude criteria that haven't recieved any impressions over
the date range.
  $reportDefinition->includeZeroImpressions = FALSE;

  // Set additional options.
  $options = array('version' => ADWORDS_VERSION, 'returnMoneyIn-
Micros' => TRUE);

  // Download report.
  ReportUtils::DownloadReport($reportDefinition, $filePath, $us-
er, $options);

  printf("Report with name '%s' was downloaded to '%s'.\n",
      $reportDefinition->reportName, $filePath);
}
```

Esimerkki raportin hausta AWQL:n avulla

```
function DownloadCriteriaReportExample(AdWordsUser $user,
$filePath) {
  // Load the service, so that the required classes are availa-
ble.
  $user->LoadService('ReportDefinitionService',
ADWORDS_VERSION);

  // Create selector.
  $selector = new Selector();
  $selector->fields = array('CampaignId', 'AdGroupId', 'Id',
'Criteria',
      'CriteriaType', 'Impressions', 'Clicks', 'Cost');

  // Filter out deleted criteria.
  $selector->predicates[] = new Predicate('Status', 'NOT_IN',
array('DELETED'));

  // Create report definition.
  $reportDefinition = new ReportDefinition();
  $reportDefinition->selector = $selector;
  $reportDefinition->reportName = 'Criteria performance report
#' . uniqid();
  $reportDefinition->dateRangeType = 'LAST_7_DAYS';
  $reportDefinition->reportType = 'CRITERIA_PERFORMANCE_REPORT';
  $reportDefinition->downloadFormat = 'CSV';

  // Exclude criteria that haven't recieved any impressions over
the date range.
  $reportDefinition->includeZeroImpressions = FALSE;

  // Set additional options.
  $options = array('version' => ADWORDS_VERSION, 'returnMoneyIn-
Micros' => TRUE);

  // Download report.
  ReportUtils::DownloadReport($reportDefinition, $filePath, $us-
er, $options);

  printf("Report with name '%s' was downloaded to '%s'.\n",
      $reportDefinition->reportName, $filePath);
}
```

Ensimmäinen versio suunnitellusta ajojärjestyksestä



AdWords API:n ensimmäinen skripti

```
<?php

//Luo oletusreitin includelle.
$path = dirname(__FILE__) . '/../../../src';
     set_include_path(get_include_path() .
PATH_SEPARATOR . $path);

     //Lataa AdWordsUser.php. Tätä kautta tulevat
kaikki komennot käyttöön.
     require_once ('Ad-
Words_api/src/Google/Api/Ads/AdWords/Lib/AdWordsUser.p
hp');

     //Hae kampanjan tiedot
     function GetCampaigns($user){

          //Alusta kampanjapalvelu
          $campaignService =
          $user->getCampaignService('v201209',
'https://AdWords.google.com');

          //Alusta kampanjapalvelun selector (valinta)
palvelu.
          $selector = new Selector();
          //Valitse näytettävät kentät (Id, Nimi)
          $selector->fields = array('Id', 'Name');
          //Päätä kampanjoiden järjestys. (Nimi, nouse-
va)
          $selector->ordering[] = new OrderBy('Name',
'ASCENDING');

          //Alusta kampanjoiden paging (sivutus) palve-
lu. (Aloitusnumero, lopetusnumero)
          $selector->paging = new Paging(0, 5);

          do {
    // Luo Get haku.
    $page = $campaignService->get($selector);

    // Näytä tulokset
    if (isset($page->entries)) {
      foreach ($page->entries as $campaign) {
        printf("Campaign with name '%s' and id '%s'
was found.\n",
            $campaign->name, $campaign->id);
      }
    } else {
      print "No campaigns were found.\n";
    }

    // Siirry seuraavalle sivulle
    $selector->paging->startIndex += 5;
```

```
  } while ($page->totalNumEntries > $selector->paging-
>startIndex);
      }

      try {

          //Alusta käyttäjän muuttujat
      $email = "testitilin sähköpostiosoite";
      $password = "salasana";
      $developerToken = "23 merkkinen avain";
      $userAgent = "Ohjelman nimi";
      $clientId = "tilinIDnumero";

      //Alusta käyttäjä
      $user = new AdWordsUser(NULL, $email, $password,
$developerToken,
          $auth, $userAgent, $clientId);
      $user->LogAll();

  // Aja funktio
  GetCampaigns($user);
} catch (Exception $e) {
  printf("An error has occurred: %s\n", $e-
>getMessage());
}


?>
```

AdWordsista saatava XML tiedosto mainoksille

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<report><report-name name="AD_PERFORMANCE_REPORT"/>
<date-range date="Feb 15, 2013-Feb 21, 2013"/>
<table>
<columns>
<column name="adID" display="Ad ID"/>
<column name="ad" display="Ad"/><column name="descriptionLine1" dis-
play="Description line 1"/>
<column name="descriptionLine2" display="Description line 2"/>
<column name="displayURL" display="Display URL"/>
<column name="destinationURL" display="Destination URL"/>
<column name="costConv1PerClick" display="Cost / conv. (1-per-
click)"/>
<column name="costConvManyPerClick" display="Cost / conv. (many-per-
click)"/>
<column name="avgCPC" display="Avg. CPC"/>
<column name="impressions" display="Impressions"/>
<column name="clicks" display="Clicks"/>
<column name="ctr" display="CTR"/>
<column name="avgCPM" display="Avg. CPM"/>
<column name="cost" display="Cost"/>
<column name="avgPosition" display="Avg. position"/>
<column name="adGroupID" display="Ad group ID"/>
<column name="convRate1PerClick" display="Conv. rate (1-per-click)"/>
<column name="convRateManyPerClick" display="Conv. rate (many-per-
click)"/>
<column name="conv1PerClick" display="Conv. (1-per-click)"/>
<column name="valueConv1PerClick" display="Value / conv. (1-per-
click)"/>
<column name="valueConvManyPerClick" display="Value / conv. (many-per-
click)"/>
</columns>
<row valueConvManyPerClick="0.0" valueConv1PerClick="0.0"
conv1PerClick="0" convRateManyPerClick="0.00%" con-
vRate1PerClick="0.00%" adGroupID="6313749082" avgPosition="0.0"
cost="0.00" avgCPM="0.00" ctr="0.00%" clicks="0" impressions="0"
avgCPC="0.00" costConvManyPerClick="0.00" costConv1PerClick="0.00"
destinationURL="http://www.example.com" displayURL="www.example.com"
descriptionLine2="Low-gravity fun for everyone!" descrip-
tionLine1="Visit the Red Planet in style." ad="Cruise #51262407ec158"
adID="21152264962"/>
<row valueConvManyPerClick="0.0" valueConv1PerClick="0.0"
conv1PerClick="0" convRateManyPerClick="0.00%" con-
vRate1PerClick="0.00%" adGroupID="6313749082" avgPosition="0.0"
cost="0.00" avgCPM="0.00" ctr="0.00%" clicks="0" impressions="0"
avgCPC="0.00" costConvManyPerClick="0.00" costConv1PerClick="0.00"
destinationURL="http://www.example.com" displayURL="www.example.com"
descriptionLine2="Low-gravity fun for everyone!" descrip-
tionLine1="Visit the Red Planet in style." ad="Cruise #51262407ec53a"
adID="21152265082"/>
</table>
</report>
```

”Mainos” taulun luontiin käytettävä MySQL komento

```
CREATE TABLE advertisement(
        AdvertisementId int NOT NULL AUTO_INCREMENT,
        PRIMARY KEY(AdvertisementId),
        AdWordsAdvertisementId bigint NOT NULL,
        Headline varchar(200),
        Description1 varchar(200),
        Description2 varchar(200),
        DisplayUrl varchar(200),
        DestinationUrl varchar(200),
        CostPerConversion decimal(10,2),
        CostPerConversionManyPerClick decimal(10,2),
        AverageCpc decimal(5,2),
        Impressions int,
        Clicks int,
        Ctr decimal(5,2),
        AverageCpm decimal(5,2),
        Cost decimal(10,2),
        AveragePosition decimal(5,2),
        ConversionRate double(10,2),
        ConversionRateManyPerClick double(10,2),
        Conversions int,
        ValuePerConv double(10,2),
        ValuePerConvManyPerClick double(10,2),
        AdGroupId bigint NOT NULL,
        CONSTRAINT fk_AdGroupIds FOREIGN KEY (AdGroupId)
REFERENCES adgroup(AdGroupId),
        InfoDate date
        )DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci
```