Data mining

Ett grafiskt tillägg för SaaS- plattformen AIMO som illustrerar SMS- och MMS- trafik samt omsättning

Alexander Sundqvist

Examensarbete för ingenjör (YH)-examen

Utbildningsprogrammet för informationsteknik

Vasa 2014

Innehållsförteckning

Förkortningar

ADO	ActiveX Data Objects
AIMO	Arena Interactive Mobile Opportunities
AJAX	Asynchronous JavaScript and XML
ASP.NET	Active Server Pages
COM	Component Object Model
CSS	Cascading Style Sheets
DAO	Data Access Object
DTO	Data Transfer Object
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IIS	Internet Information Services
JSON	JavaScript Object Notation
KDD	Knowledge Discovery and Data mining
LINQ	Language Integrated Query
MMS	Multimedia Message Service
MVC	Model View Controller
PHP	Hypertext Processor
REST	Representational State Transfer
SaaS	Software as a Service
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SVG	Scalable Vector Graphics
SVN	Subversion (Version Control System)
URL	Uniform Resource Locator
VPN	Virtual Private Network
XML	Extensive Markup Language
WAP	Wireless Application Protocol

EXAMENSARBETE

Författare:	Alexander Sundqvist
Utbildningsprogram:	Informationsteknik, Vasa
Handledare:	Mikael Jakas

Titel: *Data mining – Ett grafiskt tillägg för SaaS- plattformen AIMO som illustrerar SMS- och MMS- trafik samt omsättning*

Datum 12.04.2014 Sidantal 41 Bilagor 1

Abstrakt

Syftet med mitt lärdomsprov var att skapa ett tillägg till Arena Interactives SaaS- plattform AIMO. Projektet kunde klassas som ett data mining- projekt och tillägget skulle visa flödet av meddelanden och omsättningen i systemet under givet tidsintervall. Målet var att bygga en modul som kunde visa grafisk statistik som illustrerar Arena Interactives SMS- och MMS trafik. Den skulle även visa olika trender och en kurva på omsättningen skulle presenteras. Resultatet är idag ett tillägg som personalen på Arena Interactive använder för vecko- och månadsrapporter.

Språk: svenska Nyckelord: data mining, statistik, informationsbehandling

---

BACHELOR'S THESIS

Author:	Alexander Sundqvist
Degree Programme:	Information Technology, Vaasa
Supervisor:	Mikael Jakas

Title: *Data mining – Ett grafiskt tillägg för SaaS- plattformen AIMO som illustrerar SMS- och MMS- trafik samt omsättning*

Date 12.04.2014 Number of pages 41 Appendices 1

Abstract

The purpose of my thesis was to design and create a module for Arena Interactive SaaS-platform AIMO. The project could be classified as a data mining project where the new add-on would present the flow of messages and turnover in the system during a given time interval. The goal was to build an add-on that could display graphical statistics that illustrates Arena Interactive SMS- and MMS traffic. It would also show different trends and a graph of the turnover. The result is today a working add-on that the personnel at Arena Interactive use during weekly- and monthly reports.

Language: swedish Key words: data mining, statistics, information processing

---

OPINNÄYTETYÖ

Tekijä:	Alexander Sundqvist
Koulutusohjelma:	Informaatioteknologia, Vaasa
Ohjaajat:	Mikael Jakas

Nimike: *Data mining – Ett grafiskt tillägg för SaaS- plattformen AIMO som illustrerar SMS- och MMS- trafik samt omsättning*

Päivämäärä: 12.04.2014 Sivumäärä 41 Liitteet 1

Abstrakti

Tämän opinnäytetyön tarkoitus oli suunitella lisäosa Arena Interactiven SaaS- alustalle AIMO:lle. Projekti voidaan luokitella tiedonlouhinnaksi, missä uusi lisäosa esittää tiedon kulkua ja liikevaihtoa systeemissä annetun ajan sisällä. Tarkoitus oli rakentaa lisäosa, millä voitaisiin näyttää graafista statistiikkaa, joka kuvastaa Arena Interactiven SMS- ja MMS-liikennettä. Ohjelma näyttää myös eri suuntauksia sekä liikevaihdon vaihteluita. Tulos on toimiva lisäosa, jota Arena Interactiven henkilöstö käyttää viikottaisissa ja kuukausittaisissa raporteissaan.

Kieli: ruotsi Avainsanat: data mining, tiedonlouhintatilasto, tietojenkäsittely

---

Förord

Det har varit intressant och lärorikt att få göra detta projekt som mitt lärdomsprov. Lärdomsprovet har varit en större utmaning än vad jag trodde till en början men det har gett upphov till fördjupade kunskaper och erfarenheter. Jag vill speciellt tacka min fru och familj som har stöttat mig, samt mina goda vänner som har hjälpt mig på ett eller annat sätt under utförandet av lärdomsprovet.

Jag vill tacka min handledare Jan Westin och Joakim Foxell samt de övriga anställda vid Arena Interactive för den hjälp, stöd och feedback jag har fått under projektets gång.

Tack även till Arena Interactive som har gett mig möjligheten att få ta del av en programutvecklares vardag samt för att jag fick genomföra lärdomsprovet hos dem.

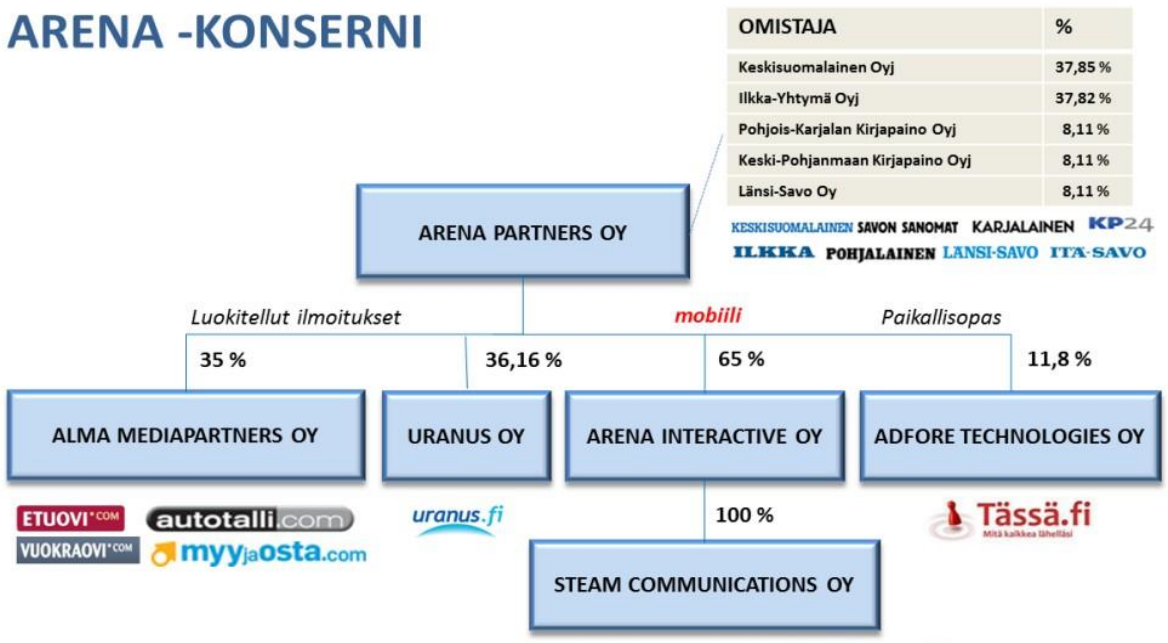
Alexander Sundqvist
10.04.2014

1 Uppdragsgivare

1.1 Arena Interactive

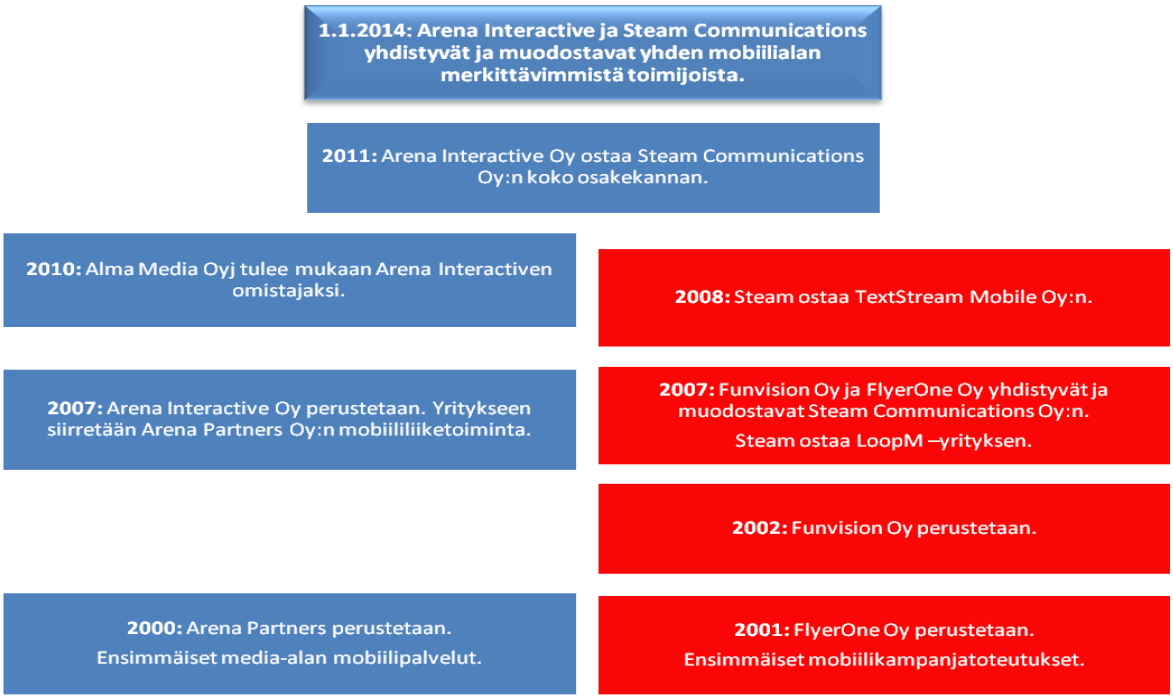
Arena Interactive är ett av de ledande företagen inom mobilkommunikationstjänster i Finland och Skandinavien. Företaget erbjuder sina kunder ett brett sortiment av skräddarsydda mobiltjänster. Arena Interactive grundades år 2007 och är ett dotterbolag till företaget Arena Partners. Arena Partners började år 2001 arbeta inom mobilbranschen och när dotterbolaget Arena Interactive grundades år 2007 flyttade man så gott som all mobilverksamhet dit. Arena Partners ägs av tidningskoncerner i Finland och dessa är Keskisuomalaninen, Savon Sanomat, Ilkka, Pohjalainen, Keskipohjanmaa, Itä-Savo, Länsi-Savo och Karjalainen. /9,6/

I mars 2010 köpte Alma Media en del av Arena Interactive. Arena Interactive köpte i sin tur upp Steam Communications år 2011 och från och med 01.01.2014 så förenades dessa två företag. Arena Interactive och Steam Communications fusionerar sina verksamheter och fortsätter under namnet Arena Interactive med en gemensam VD. /6/



Figur 1. Arena koncernens andelar. /6/

Efter fusionen mellan Arena Interactive och Steam Communications har företaget totalt 21 anställda i Vasa, Jyväskylä och Helsingfors. Programutvecklingen och en del av kundtjänsten sker i Vasa. Marknadsföring och kundtjänsten finns i Jyväskylä. I Helsingfors finns en mångsidig försäljning, informationshantering, marknadsföring och produktutveckling. Timo Wallius är den nya gemensamma Vd:n för Arena Interactive och Steam Communications som blev invald externt efter sammanslagningen. /7/



Figur 2. De större förändringarna vid Arena Interactive genom åren. /8/

Arena Interactive har idag många kunder, både nationellt och internationellt. Kundantalet är närmare 300 företag, varav majoriteten finns i Finland. Ett par kända kunder i Österbotten är till exempel Anvia och Pohjalainen. Arena Interactives kunder är främst inom mediabranschen och dagstidningar och därför har det blivit naturligt för Arena Interactive att fördjupa sig i tidningsbranschen och gjort skräddarsydda lösningar för dem. /8/

Efter sammanslagningen av Arena Interactive och Steam Communications så har kundkretsen utvidgats både geografiskt och produktmässigt. Steam Communications kundbas har koncentrerats sig kring huvudstadsregionen och deras tjänster sträcker sig också utöver de traditionella dagstidningarna.

1.2 AIMO

AIMO står för Arena Interactive Mobile Oppertunities och är Arena Interactives huvudsakliga webbtjänst. AIMO är en SaaS-plattform (Software as a Service) och med hjälp av den så kan Arena Interactives kunder hantera alla sina mobiltjänster dygnet runt, till exempel marknadsföring och betaltjänster. Nedan i figur 3 kan man se webbsidan till Arena Interactives flaggskepp AIMO. /10/

Figur 3. Första sidan på www.aimomedia.com. /8/

AIMO är en webbaserad plattform, en webbtjänst för Arena Interactives kunder. Webbtjänsten möjliggör för kunderna att underhålla sina tjänster hela tiden, oavsett var i världen man befinner sig. Här kan de hantera alla slutkunder, skicka SMS och MMS av olika slag. Med AIMO så behöver kundtjänsterna inte utföra arbetet manuellt eftersom den integreras med tidningarnas distributionssystem och sköter således det hela per automatik. Detta är en av de största fördelarna med AIMO att det mesta går att skötas med automatiserade processer för att minimera resursanvändningen. AIMO:s huvudsakliga

tjänster är följande; kund- och marknadsmeddelanden, mobila betaltjänster, elektroniska biljetter och kuponger samt specialanpassade integrerade tjänster. /10/

Som ett praktiskt exempel kunde man ta en tidsbunden prenumerant av dagstidningen Pohjalainen. När prenumerationen håller på att gå ut så kan Pohjalainen skicka ut ett SMS som frågar kunden om denna vill förlänga prenumerationen. Ifall kunden önskar förlänga prenumerationen så svarar denna på sms:et och prenumerationen av tidningen fortsätter.

2 Uppgift

Uppgiften som Arena Interactive gav mig var att utveckla en modul till deras SaaS-plattform AIMO. Målet med modulen var att bygga upp grafisk statistik som skulle illustrera Arena Interactives SMS- och MMS-trafik samt omsättning. Modulen skulle även presentera olika trender och mönster i form av hur många meddelanden som levereras och behandlas per dag.

Med hjälp av modulen skulle Arena Interactive få en bättre helhetsöversikt över flödet av meddelandena och omsättningen av pengar i systemet under ett givet tidsintervall. Andra önskemål var att man skall kunna se företagets trafik och omsättning som sin helhet men även gå in på närmare nivåer som exempelvis land, företag eller kortnummer. Modulen utvecklas i C#, .NET webbforms tillsammans med lämpliga grafiska illustrationer av resultatets data. Tillägget till AIMO kommer att användas av företagets ledning vid månads- och veckorapporter.

3 Struktur, verktyg och tekniker

3.1 Struktur

I detta kapitel presenteras generella tillvägagångsätt om hur man bygger upp en webbsida samt vilka av dessa mitt tillägg har verkställts genom. Här presenteras och jämförs front end, med back end eftersom jag vill ge en helhetsöverblick om hur de olika delarna hänger ihop och på vilket sätt de är beroende av varandra.

3.1.1 Front End

Front end är det som användaren ser när en applikation körs. Front end fungerar således som en GUI och den sköter all kommunikation mellan användaren och back end:en (3.1.2). GUI:n presenterar datan för användaren via det grafiska gränssnitt som finns tillgängligt. Ofta används till exempel HTML (Hyper Text Markup Language) som grund till texten, CSS (Cascading Style Sheet) gör det snyggt och JavaScript möjliggör dynamisk interaktion med webbsidan. Front end svarar för att skicka och ta emot en varierande mängd data och information. Front end:en pratar med back enden via någon slags service eller metod som mellanhand och den i sin tur sätter till eller begär data, från till exempelvis en server. /11/

Front end:en i mitt tillägg till Arena Interactives plattform är byggt i C# och ASP.NET webbforms och kompletterat till en stor del med JavaScript. Front end:en är designad med hjälp av CSS och JavaScript. Allt detta körs mot Arena Interactives Maria databas där datan finns lagrad, med hjälp av olika anrop både i kod och med AJAX- (Asynchronous JavaScript and XML) förfrågningar.

3.1.2 Back End

Motsatsen till front end är back end. Back end är där all kod och funktioner utförs när användaren klickar på olika knappar och funktioner i front end:en. Back end:en tar emot data från front end:en genom olika förfrågningar som bearbetas och formateras till ett lämpligt format. Back end är för övrigt ofta associerat med servar men det är inte hela sanningen utan oftast består back end:en av någon slags service, en standard. De vanligaste DTF (Data Transfer Format) är JSON (JavaScript Object Notation) och XML (eXtensive Markup Language). De vanligaste servicetyperna är i sin tur är REST (Representational State Transfer) eller SOAP (Simple Object Access Protocol).

I mitt projekt har jag använt mig av Webmethods och de fungerar som en webb service som exponerar publika metoder för HTTP- anrop. Webbmetoderna kommunicerar således mellan front end och back end som i sin tur hanterar datan i JSON- format. Webbmetoderna är en webbservice som är väldigt intelligent och förstår till exempel att själv serialisera och konvertera databasobjekt direkt till ett läsligt JSON-format. För att kunna kommunicera med databasen så behöver man en mekanism som stöder och hanterar datan till och från servern på ett stabilt sätt. Webbmetoderna som nämnts ovan använder sig indirekt av DTO för att kommunicera med databasen. DTO:n har inget med själva databasen och dess tabellnamn, kolumner eller fält att göra utan är ett sätt att hantera datan som skickas fram och tillbaks. Man kunde också påstå att DTOn används inte i dess sanna benämning utan byggs upp dynamiskt i anropet som anonyma objekt.

Webbmetoderna är asynkrona, det vill säga att när data hämtas till en webbsida så går det fortfarande att använda den övriga webbsidan utan att den fryser fast och användaren måste vänta tills datan är hämtad. I ett asynkront anrop begär klienten data för valda element, i detta fall ett JSON- objekt med data som grafen skall presentera. De övriga elementen på webbsidan går då fortsättningsvis att använda eftersom det är bara grafen som är låst. Motsvarigheten till asynkront är synkront. När man hämtar data synkront betyder det kort sagt att hela webbsidan fryser och laddas om helt och hållet. Utifall att jag skulle använda mig av en metod som hämtar data synkront skulle hela sidan laddas in på nytt, samt datan som grafen skall presentera.

Data transfer- formatet webbmetoderna använder sig av är JSON, som är ett textbaserat format för utbyte av data som är väldigt kompakt och effektivt. Generellt sätt så är JSON ett enkelt sätt att hantera data både för programmerare och för automatiserade data processer. JSON kan användas i de vanligaste datatyperna och i mitt fall med hjälp av objekt och Arrays. Ett enkelt exempel av hur ett JSON objekt av typen Array kan användas finns i kodexempel 1. /20/

```
{
"employees": [
{ "firstName":"John" , "lastName":"Doe" },
{ "firstName":"Anna" , "lastName":"Smith" },
{ "firstName":"Peter" , "lastName":"Jones" }
]
}
```

Kodexempel 1. Några namn sparade i ett JSON objekt av typen Array. /20/

Douglas Crockford utvecklade och använde JSON flera år innan han skapade webbsidan json.org. Efter detta fick Google och Yahoo nys om detta och började använda sig av JSON. Orsaken till att JSON har blivit alltmer populär är för dess lättillgänglighet och språkoberoende men även för att det är mycket samspelt med AJAX. Dessutom lär JSON vara effektivare och prestera bättre i jämförelse med XML (eXtensible Markup Language). XML är precis som JSON ett sätt att hantera, lagra och sända data men påminner mera om HTML. XML hanterar huvudsakligen text(även JPEG-bilder och listor) men däremot så klarar JSON av text, numer och Arrays. På grund av detta så är det naturligt att jag använder mig av JSON i mitt projekt eftersom JSON är objektorienterat och XML är dokumentorienterat. JSON har en klar och tydlig syntax för både Arrays och listor, det är upp till utvecklaren att förstå vad som är vad och hur man använder DTF:erna. Det finns för- och nackdelar med allt, XML är även som JSON mycket välanvänt idag men vilket sätt man väljer att använda beror helt på själva typen av datan som hanteras. /21/

3.2 ASP.NET och .NET Framework

I kapitel 3.2 presenteras alla former av Active Server Pages och master page samt hur de fungerar i sin korthet. Här beskrivs även de olika ramverkens tjänster, strukturer och implementationer samt jämförelser mellan de äldre varianterna och med den mest moderna MVC.

3.2.1 ASP

Active Server Pages (ASP) är en Microsoft server-side teknik som ger möjlighet till att skapa interaktiva, dynamiska och användarvänliga webbsidor. ASP kan köras i alla webbläsare eftersom IIS servern översätter ASP:en så webbläsaren kan tolka den i form av HTML. De största och vanligaste skriftspråken kan kombineras och inkluderas i en ASP-webbsida för att få den funktioner och visuella utseendet man önskar. Webbsidorna som skapas med ASP kan även nå samma tjänster som Windowsapplikationer med hjälp av IIS:en. Dessa är till exempel SMTP (Simple Mail Transfer Protocol) för epost utskick, COM(Component Object Model) struktur och ADO (ActiveX Data Objects) för att kommunicera med Windows- baserade databaser. En ASP- webbsida blir anropad via en IIS (Internet Information Services) server och är implementerade genom ett dynamiskt bibliotek, asp.dll. /23/

3.2.2 ASP.NET

ASP.NET fungerar på liknande sätt som klassiska ASP, det är också utvecklad för att bygga smidiga webbsidor och tjänster. Största skillnaden mellan klassisk ASP och ASP.NET är att i ASP.NET skapar man webbsidor med .NETs programmeringsspråk som till exempel C# och VB:NET. Prestandan är tillfredsställande och har varit bland de mest flexibla när det gäller att bygga och implementera nya funktioner till ASP.NET- baserade webbsidor. /23/

3.2.3 ASP.NET Web Form

ASP.NET web forms bygger på den ovannämnda ASP.NET men är en av den mest använda ASP- tekniken som finns idag. Den är kallad web forms för att dess funktioner och möjligheter är den samma som hos Windows forms, därav namnet web forms. Web forms använder sig av en server-side modell som körs på servern men renderas på klientsidan som HTML. Attributet ”runat=’server’” anger att web formen måste köras på servern. /23/

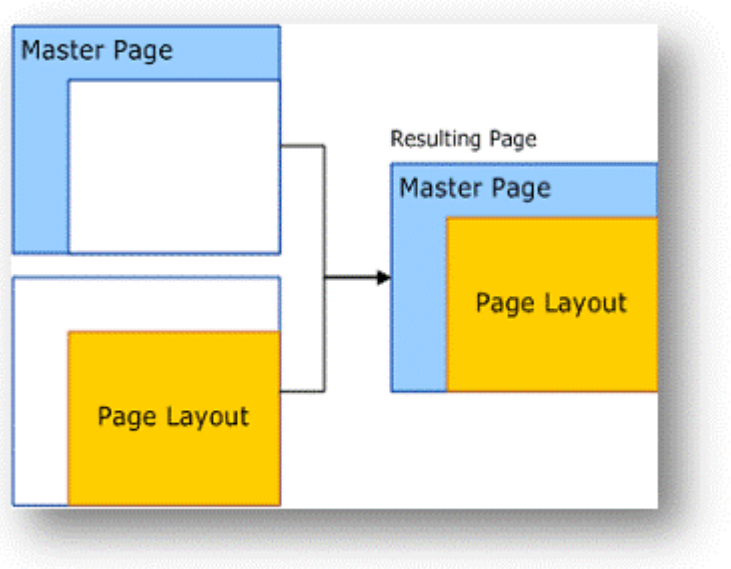
3.2.4 ASP.NET MVC

Ett modernare alternativ till webforms skulle vara MVC (Model View Controller). Allt fler företag och systemerare föredrar idag att använda sig av MVC istället för webbmetoder, eftersom det är betydligt snabbare och smidigare på många sätt. Orsaken till att man föredrar att utveckla i MVC idag är för att dess struktur möjliggör mer återvinning av programkoden, speciellt i större projekt. Från och med utvecklingsverktyget Visual Studio 2012 så möjliggör Microsoft parallell användning av den klassiska ASP:en med den moderna MVC. Detta betyder att man kan blanda båda teknikerna idag men man borde kanske sträva efter att vara konsekvent och använda den ena varianten.

Många anser att webforms är ett föråldrat sätt att arbeta på och det måhända stämma. Problemet för företag och projekt som är kodade enligt den aningen äldre metoden med hjälp av web forms, är att det är väldigt mycket arbete att förnya ett helt projekt från web forms till ett MVC- strukturerat projekt. Därav är orsaken till att web forms lever vidare ända fram tills ett projekt blir förnyat helt och hållet. MVC är ett annat sätt att hantera och skicka förfrågningar på. Det här förverkligas genom att man skickar förfrågningar till controllern istället för via webmetoden som sedan formaterar modellen, som i sin tur är kopplad till en service eller databas. MVC är på samma sätt som en ASP.NET webforms mellanhand som sköter om kommunikationen mellan GUI:n och databasen. Den sköter alltså om att hämta, uppdatera och ta bort data från en databas enligt inputen från GUI:n som användaren har givit. /25/

3.2.5 Master Pages

Många webbsidor har idag samma element och funktioner som används på ett flertal sidor. ASP.NET har en egen mall för sådana sidor som används ofta, master page. En master page kan användas som en mall för alla sidorna på till exempel en domän. Se figur 4 för illustration av master page. Gemensamma element är ofta sidhuvud med logobilder, sidfötter med datum och information om webbmastern, samt olika varianter av menyer. Genom att använda sig av en master page så behöver man inte upprepa samma kod för varje sida man skapar utan detta ramverk används för alla. På detta sätt är det enkelt att underhålla ett flertal sidor eftersom en ändring berör alla de webbsidorna som använder sig av master pagen. En master page kan precis som alla andra ASP.NET- webbsidor innehålla både CSS och JavaScript. /24/



Figur 4. Illustration av hur en ASP.NET master page fungerar. /24/

3.3 Webb- och skriptspråk

I detta kapitel presenteras kända webb- och skriptspråk som har använts i tillägget jag har skapat. Här beskrivs språken allmänt där kritiska och personliga åsikter framkommer. En del av dessa har varit förbestämt från arbetsgivaren eftersom de har använts tidigare i webbtjänsten AIMO.

3.3.1 CSS

CSS eller Cascading Style Sheet är ett stilschema vars uppgift är att designa och utforma webbsidors utseende. CSS kan inte jämföras med HTML eller PHP utan är enbart ett sätt att ange stilen för en webbsida. Det finns flera olika sätt att använda CSS på men det vanligaste sättet är nog att man skapar en skild CSS-fil som sedan inkluderas i det projekt, webbsida, som man vill designa. Andra sätt är att direkt i sin webbsida skapa en stilmall med CSS eller det sista, att ange varje enskilt elements attribut med ”style”. CSS är ett av de viktigaste verktygen en webbdesigner kan använda sig av för att kunna bygga en fräsch och modern webbsida. /27/

3.3.2 Google Charts

Google charts är ett kraftfullt grafiskt verktyg som är relativt enkelt att använda som utvecklas och uppdateras regelbundet där nya funktioner och graftyper tillkommer. Google Charts är kompatibelt med de flesta programmeringsspråk och är dessutom gratis att använda för alla, även företag. Google Charts möjliggör ett enkelt sätt att visualisera data grafiskt på en hemsida. Det finns ett 30-tal olika grafer tillgängliga och massvis med exempel på hur man kan använda dem beroende på hurudan data man behandlar och hurudant resultat man strävar till att uppnå. Google Charts baserar sig helt och hållet på JavaScript och går därför att implementera i de flesta projekt oberoende av programmeringsspråk. /17/

3.4 Övriga tekniker

I kapitel 3.4 presenteras de övriga teknikerna som jag har använt mig av i projektet. De är främst databassystem och jämförelse mellan det vanliga MySQL och den nya varianten MariaDB. Här presenteras även de största skillnaderna samt för- och nackdelar mellan dessa två databashanteringssystem.

3.4.1 MySQL

SQL står för ”Structured Query Language” och är den mest använda standarden inom databashantering. MySQL är en teknik som släpptes 1992 av ett svenskt företag TcX DataKonsult AB. De ursprungliga utvecklarna av MySQL var två svenskar, Allan Larson och David Axmark, samt en finlandssvensk, Michael Widenius. MySQL är lanserad under GNU (General Public License) version GPLv2. Idag är den nyaste officiella standarden av MySQL ”SQL:2003”.

MySQL är världens populäraste open source databashanteringssystem med över 100 miljoner nerladdningar. Det finns många olika databashanteringsverktyg idag, varav MySQL är en av de mest använda. Många stora företag som Facebook, Adobe, Nasa och Google förlitar och använder sig av MySQL. MySQL har öppen källkod men databassystemet utvecklas och underhålls av Oracle Corporation. Med öppen källkod menas att alla får ladda ner klient/serverprogrammet gratis och använda det efter behov. Om man vill titta på och ändra koden så får man göra det enligt önskemål. För att få distribuera en egen modifierad version behöver man först köpa en licens av Oracle Corporation. /26/

3.4.2 MariaDB

MariaDB är en förbättrad variant av MySQL eller en så kallad ”fork”. Både MySQL och MariaDB utvecklas vidare idag, största skillnaden är att Oracle äger MySQL och MariaDB är en open source produkt. MariaDB fungerar för övrigt likadant som MySQL och därför

behöver man inte konvertera eller omstrukturera en databas vid ett eventuellt byte. MySQL5 är utgångspunkten för MariaDB och all funktionalitet fungerar precis på samma sätt som hos MySQL. Det finns inga stora förändringar mellan de två olika varianterna men MariaDB har bättre prestanda samt ett antal förbättrade funktioner. MariaDB stöder till exempel binär loggning och tidsstämplar ända ner till mikrosekundnivå. Även joins och subquerys är under utveckling för att arbeta snabbare. Trots att det inte är några större förändringar så är MariaDB mer optimerad än MySQL och är därför ett solklart alternativ till den vanliga MySQL:en. Utvecklingsstegen är små men stabila. MariaDB har dessutom en utförlig dokumentation på deras hemsida, mariadb.com, ifall det är något man undrar över. /28/

3.5 Verktyg

I detta kapitel presenteras och diskuteras de verktyg jag har använt mig av under projektets gång. Här finns kort historik samt personliga och konstruktiva åsikter om verktygens användarvänlighet, prestanda och ändamål.

3.5.1 SQLyog

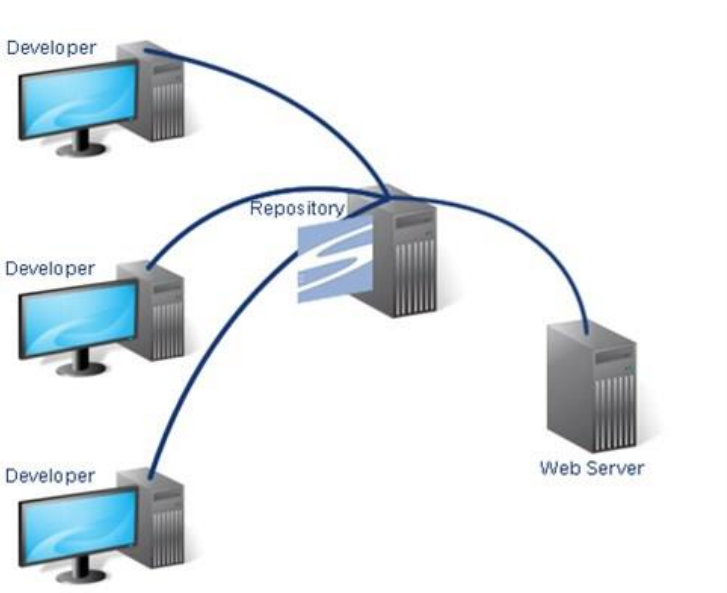
SQLyog är ett MySQL databashanteringsprogram. SQLyog är ett snabbt och smidigt verktyg för att hantera databasers strukturer och dess innehåll. Det finns alla behövliga funktioner som är lätt tillgängliga i det grafiska gränssnitt som finns till förfogande. Man kan lätt administrera användarrättigheterna för en databas och dess användare. Det går snabbt och enkelt att ta säkerhetskopior och att flytta över data från en databas till en annan. SQLyog finns i en enkel gratisversion men för att få använda alla funktionalitet behövs en avgiftsbelagd licens. SQLyog används idag av över 2 miljoner användare och 30 000 företag runt om i världen. /13/

3.5.2 Sublime

Sublime är ett textediteringsprogram som påminner mycket om det kända Notepad++. Sublime är mycket mer komplex än de övriga, simpla textediteringsprogrammen. I Sublime kan man ange i vilket programmeringsspråk man skriver, så ger Sublime den riktiga färgkodningen till din kod. Sublime är en intelligent texteditering och det finns oändligt med smarta funktioner till förfogande bara man sätter sig in i dem och har tiden att lära sig. Till exempel kan man ordna om ett kodblock snabbt så den får det rätta utseende genom att använda dess snabbtangenter och kommandon. Sublime är troligtvis ett bättre alternativ för en utvecklare än Notepad++. /14/

3.5.3 TortoiseSVN

TortoiseSVN är ett av de många tillgängliga versionhanteringssystem som finns och SVN baserar sig på Subversions hanteringssystem. TortoiseSVN är relativt enkelt att använda eftersom det integreras med både Windows Shell och Microsoft Visual Studio. Alla kommandon kan göras grafiskt direkt via Windows Explorer eller via önskat program som integrationen fungerar med. SVN har en enkel layout och är mycket användarvänligt i de flesta situationer. Se figur 5 för en illustration av hur ett versionhanteringsprogram fungerar. /31/



Figur 5. Illustration av hur man kan använda SVN. /31/

TortoiseSVN har tydliga loggar om vad man har gjort i vilka revisioner och kan återta gamla revisioner snabbt och enkelt ifall det behövs. TortoiseSVN har inte alltid fungerat felfritt. Man kan använda programmet på flera olika sätt men det vanligaste är att alla programutvecklare arbetar mot samma kopia av ett projekt, det vill säga samma repository och laddar turvis upp nya versioner. Ifall det är många klienter som använder samma filer så finns det en bra lösning till detta, genom att låsa filen. På detta sätt undviker man att flera klienter ändrar i samma fil samtidigt och då kan problem uppstå. Det här blir snabbt till ett problem med större företag där det finns många som använder samma filer dagligen men då kanske man skulle använda sig av ett annat version hanteringsprogram än SVN. Vid Arena Interactive har TortoiseSVN fungerat bra på grund av att det är endast ett halvt dussin programmerare som använder SVN, varav endast ett par använder sig av samma filer nån gång per vecka.

Ett annat stort och omtyckt versionshanterings program är Git. Git har öppen källkod och är gratis för alla användare. En stor skillnad mellan SVN och Git är också att Git används främst i en textbaserad terminal där man anger kommandona i text och således kanske är ett snabbare alternativ att använda för en erfaren systemerare. Det finns även grafiska varianter av GIT men via terminalen är kanske det vanligaste sättet att arbeta. Det finns för- och nackdelar med alla version hanteringsprogram och faktorer som tillgänglighet, stabilitet och säkerhet påverkar alltid slutanvändarens val av mjukvara. /16/

3.5.4 Sophos VPN client

VPN eller Virtual Private Network är ett vanligt sätt att arbeta exempelvis hemifrån. Det finns många sätt att ansluta till en VPN- tunnel, endera genom en inbyggd Windows-lösning eller genom någon annan tredjeparts applikation. Sophos är enkelt att installera, använda och dessutom har en utförlig dokumentation. Sophos har arbetat med och strävat till att få en hög säkerhet och kryptering av datan som skickas över tunneln till det anslutna nätverket. Sophos VPN client använder sig av enkrypteringen SSL. Andra standard enkrypteringar som finns är till exempel SSL, PPTP, L2TP, Ipsec och Cisco. En VPN tunnel i sin korthet på så sätt att när en användare tar med sig arbetsdatorn hem, så kan

användaren ansluta via sitt privata hemmanätverk till sitt arbetsplatsnätverk för att nå viktiga filer och databaser genom denna krypterade tunnel. /19/

I mitt fall fanns Sophos VPN client tillgängligt och dess programvara har jag använt mig av. Sophos har visat sig fungera stabilt utan några som helst problem, så det är nog ett program jag kan tänka mig att använda i framtiden och även rekommendera till andra som söker VPN-lösningar, både till privatpersoner och företag.

3.5.5 Google Chrome Tillägg

JSON editor online är en Google Chrome extension eller tillägg för som möjliggör felsökning av JSON- objekt. Även när det gäller felsökning och testning av JavaScript så fungerar detta tillägg. I min modul som innehåller en hel del JavaScript och JSON formaterad data så har tillägget varit till en stor hjälp både under testning och felsökning.

4 Data mining

Här presenteras data mining och dess betydande del i processen som utförs när man söker fram relevant information ur väldiga mängder data. Tillägget hanterar stora mängder data och den grafiska framställningen av datan består av en process som kan ses som data mining. Resultatet som man kan avläsa av graferna visar trender och mönster från Arena Interactives kunders användning.

4.1 Introduktion

I dagens högteknologiska samhälle sparas massvist med information dagligen om så gott som allt. Går något snett så finns det alltid digitala fingeravtryck och historik som man kan spåra sig tillbaka med. Om man inte sparar själv vad man gör så är det i de flesta fall någon annan som gör det för en, medvetet eller omedvetet. Till exempel bankerna har sina transaktioner som sparas många år framåt och stora it-företag vill gärna hålla reda på vilka internet sidor vi besöker regelbundet. Allt efter att tekniken utvecklas, datorer blir snabbare och större och sättet vi hanterar vår data så kommer våra liv alltid att finnas i digital historik vare sig man vill eller inte. Hur som helst så är dagens informationssamhälle ett sådant att vi samlar på oss oändliga mängder data som sedans sparas i olika former. Frågan är bara om vi kan lära oss av våra gärningar och försöka förutspå framtiden i form av trender och statistik.

> *"Most organizations have large databases that contain a wealth of potential accessible information. However, it is usually very difficult to access this information. The unbridled growth of data will inevitably lead to a situation in which it is increasingly difficult to access the desired information: it will always be like looking for a needle in a haystack, only the amount of hay will be growing all the time."*(Adriaans & Zantinge, 1996). /4/

4.2 Historia

Ordet data mining blev infört redan under IT-boomen på mitten av 1970-talet när man började spara information digitalt, men det var först i slutet av 1980-talet som man började använda termer och uttryck som KDD och data mining. Vartefter åren har gått så har teknologin utvecklats mycket och även metoderna för att analysera data har delvist ändrats. Själva data miningen består av flera olika teorier som härstammar från olika grunder. Det finns inte heller någon direkt översättning från engelskans data mining till svenska så det har därmed blivit ett låneord som blivit allt mer känt. /30/

Denna teknologi som har vuxit fram är inte ett ensamt tillvägagångssätt utan den består av en hel samling verktyg som används under processen i sin helhet. Teknikernas ursprung varierar men har alla en gemensam faktor, en stor mängd data. De ursprungliga teknikerna till data mining är statistik, databasteknologi, artificiell intelligens, maskininlärning och visualisering med flera. Data mining används oftast endast där ursprungsmängden data är väldigt stor, ifall det handlar om en mindre mängd data kan man med mer klassiska verktyg eller för hand plocka ut eller visualisera eventuella trender och mönster. Data mining har alltså utvecklats för att kunna upptäcka och hitta värdefull information och mönster från stora kvantiteter av data (Bertino, 2001).

4.3 Data mining i sin korthet

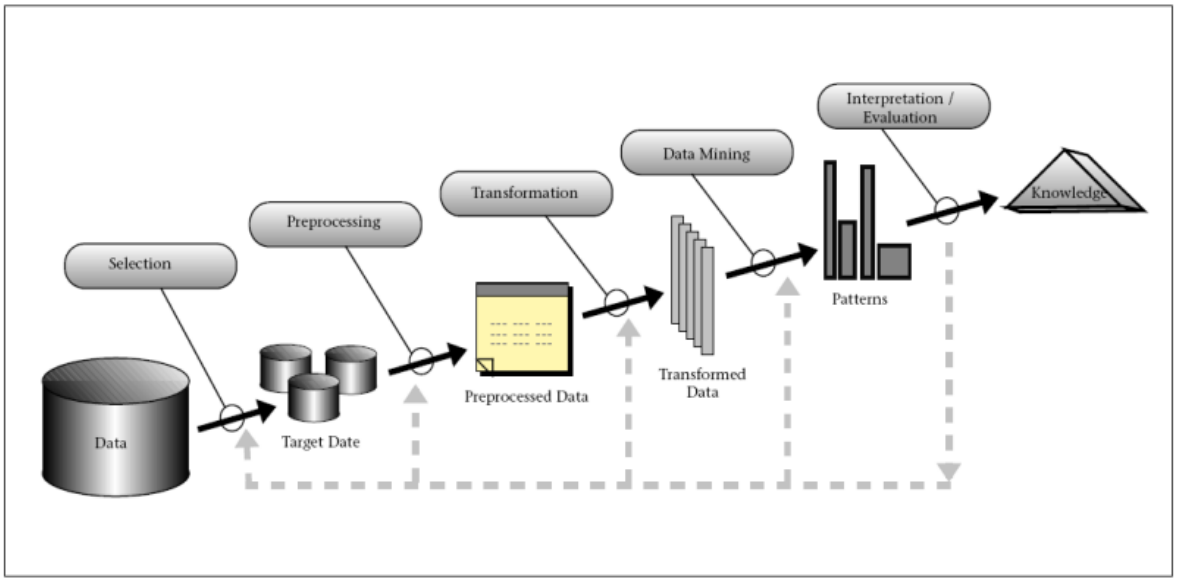
Det finns inget enkelt sätt att beskriva ”data mining” på. Det är också svårt att dra en gräns på vad som är data mining och vad som inte är. Data mining är i sin helhet ett väldigt stort och komplex område. Om jag skulle försöka beskriva vad data mining är i en mening så skulle den lyda ungefär såhär; ”Att med automatiska metoder få fram dold och eller relevant information samt trender ur väldigt stora mängder data”. Data mining kan var en hel- eller halvautomatisk process som analyserar stora mängder data för att hitta olika mönster och trender. Data mining är alltså en slags teknik som man använder sig av när man extraherar värdefull information, oftast från någon slags databas där datan kan vara upp till biljoner rader. KDD eller Knowledge Discovery in Databases används ofta som en

synonym till data mining trots att KDD egentligen beskriver hela processen där själva data miningen endast representerar en bit av utförandet.

> *”Data mining is the analysis of (often large) observational datasets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.”* (Hand & Mannila & Smyth, 2001) /3/.

4.4 KDD- processen och data mining

Det finns en mängd olika tillvägagångssätt och tekniker när det gäller att gräva fram relevant data, det beror helt på hurudan data man analyserar och vad man önskar uppnå för resultat. I detta arbete nämner jag ett sätt och kort tittar på hur en KDD- process kunde se ut, se figur 6. Figuren ger även en bättre uppfattning om var själva data miningen kommer in och att den egentligen bara är en liten del av en lång process, trots att den är viktig. Alla steg i hela processen är likvärda och varje stege är viktigt för att det följande skall ha rätt och tillförlitlig utgångsdata. Dessutom upprepas flera steg i processen för att få ett dugligt resultat, det vill säga att processen är iterativ. Nedan i figur 6 visar en variant av hela Knowledge Discovery in Databases- processen. /18/



Figur 6. Knowledge Discovery in Database- (KDD) processen. /18/

I figur 6 börjar man KDD processen ”selection” med att bestämma och identifiera vad man vill söka och vilka ändamål man behöver resultatet till. I det andra steget begränsar man data som skall bearbetas. Begränsningen kan ske med hjälp av datumintervaller och får på så sätt redan då en hel del mindre data att hålla reda på. Här är det även viktigt att beakta om datan lämnar oförändrad eller huruvida den kan förändras. Exempelvis om det tillkommer ny data till en databas och hur man kan beakta och begränsa denna.

Tredje steget ”preprocessing” innebär att man förbereder och kontrollerar datan som skall formateras om. Här är det viktigt att all data är konsekvent och ser likadan ut i ett och samma format, för att kunna jämföras med varandra. På så sätt eliminerar man eventuella tomma värden eller övriga avvikelser. Till exempel är det svårt för en bilförsäljare att jämföra bilar och dess inköpspriser mellan olika valutor. Så här skulle bilförsäljaren omvandla alla inköpspris till samma valuta för att enklare få en uppfattning om vad som är dyrt respektive förmånligt.

I det fjärde steget så blir all irrelevant och onödig data borttagen så man kan fortsätta att fokusera på datan med ett värdigt innehåll. Redan nu kan man utföra själva data miningen då man har all data i samma format och har gallrat ut onödig data. I det femte och sjätte steget kan man köra en data mining algoritm eller en metod för att hitta ett mönster i datan. Under det sjunde steget utförs själva data miningen som också är en del av det åttonde och sista steget.

I de två sista stegen söker man mönster som undersöks och tolkas. Här kommer man med högsta sannolikhet att hitta flera mönster och trender men man bör sträva till att få bort otydliga och svårtolkade resultat. När man väl har ett resultat man har strävat efter behöver man slutligen något slags hjälpmedel för att beskriva vad man har hittat. Ofta använder man sig av visuella grafer för att på ett enklare och mer överskådligt sätt följa upp det resultat man har åstadkommit. Avslutningsvis bör man dokumentera och ge de berörda partnerna möjlighet att ta del av resultatet så man kan förutspå trender eller undvika eventuella framtida problem. /18, 22/

4.5 Summering av data mining

KDD och data mining handlar med andra ord om att försöka bena ut all irrelevanta data för att sedan fokusera på det viktiga för att uppnå resultat. Slutligen är det även mycket viktigt att kunna beskriva och visa vad den nya utvunna informationen visar och antyder på. Till exempel i mitt lärdomsprov så visas slutresultatet grafiskt för att snabbt få en tydlig och klar uppfattning om Arena Interactives kunders trafik och omsättning. Där är all information lagrad i en databas som i de flesta fall idag. Datan begränsas först med hjälp av datumintervaller och sedan de val användaren gör via den tillgängliga GUI:n jag har designat och implementerat.

Det finns ett oändligt antal tillvägagångssätt för data mining, i kapitel 4, Data mining, har jag valt att presentera ett tillvägagångsätt med en metod av KDD och data mining. I ämnet data mining kan man fördjupa sig i nästan hur mycket som helst. Andra sätt kunde exempelvis vara genom avancerade genetiska algoritmer, minnesbaserade resonemang, kluster och olika beslutsträd. Dessutom så utvecklas data miningens tillvägagångssätt i snabbt takt för att få fram nya trender och mönster på, precis på samma sätt som all annan teknik utvecklas idag som blir både snabbare, säkrare och större kapaciteter på mindre ytor.

För att lyckas få ett trovärdigt och realistiskt resultat genom KDD och data mining så är det oerhört viktigt att datan behandlas på rätt sätt och att all data är i samma format. Annars kan det uppstå stora fel redan i ett tidigt skede i processen och det resulterar i att datan blir obegriplig och helt oanvändbar. En annan viktig sak är att man har tillräckliga mängder med data så att man kan utvinna mönster och trender som faktiskt är realistiska.

5 Utförande

I kapitel 5 beskriver jag hela projektets gång från planeringen till den färdiga applikationen. Här diskuteras bland annat tilläggets olika faser i programmeringen, strukturen och problem som har uppstått samt deras lösningar. Det finns också personliga åsikter om verktyg, metoder och tekniker som har både för- och nackdelar lite beroende på hur de har använts.

5.1 Planering

När man skall designa och planera en applikation eller ett tillägg för ett befintligt system finns det mycket man bör ta i beaktande. Först bör man forska och jämföra olika metoder och tekniker med beaktande på eventuella befintliga system. Det finns som med allt annat, för- och nackdelar beroende på hurudana tillvägagångssätt, metoder och verktyg man väljer att använda sig av.

För att jag skulle kunna planera tillägget för Arena Interactives plattform AIMO måste jag bilda mig en helhetsuppfattning om vad projektet innebär. Jag började med att bekanta mig med hur Arena Interactives flaggskepp AIMO har byggts upp. Det förberedande arbetet var väldigt tidskrävande eftersom man måste försöka ta sig an en djupdykning direkt ned i en obekant kod i en miljö som till stor del är helt nytt. Eftersom mitt tillägg innehåller en väldig mängd data, behövde jag även bekanta mig med MySQL databasens struktur för att jag skulle kunna hitta den data jag söker.

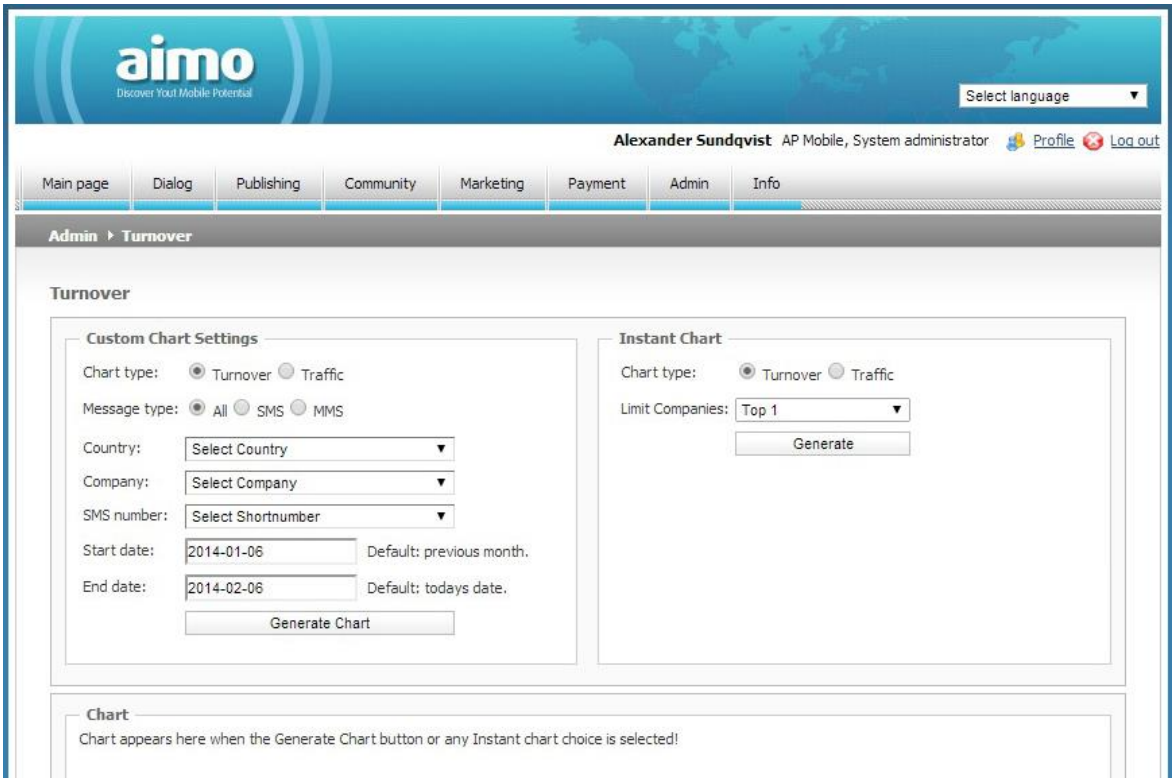
Verktyg och tekniker som jag har använt har delvis varit förbestämt till den standard som Arena Interactive använder i dagsläget. Programmeringsspråket har varit C# med MySQL som databas, och deras verktyg har varit Microsoft Visual Studio 2010 och SQLYog, varav samtliga är bekanta sedan tidigare för mig från min utbildning. Arena Interactive använder sig även av versionshanteringsprogrammet TortoiseSVN, vilket var obekant för mig och därmed fick jag möjligheten att lära mig ett nytt versionshanteringsprogram.

Förutom de förbestämda programmen och teknikerna, har jag haft väldigt fria händer att planera, testa och verkställa deras önskade grafiska tillägg. Jag fick börja med att gå direkt på hjärtat av mitt projekt, det vill säga själva grafen. Här fanns det många möjligheter men kriterier från min arbetsgivare var att den skulle vara gratis för kommersiellt bruk samt att det skulle gå att implementera i mitt projekt. Redan i ett tidigt skede av forskningen kring hurudan teknik jag skulle använda för grafen, kunde jag konstatera att JavaScript erbjuder många olika alternativ till lösningar. Efter några veckors tid med planering och forskning så hittade jag några lämpliga kandidater som jag skalade ner en efter en tills jag hade en trolig lämplig lösning. Jag tittade en tid på inbyggda .NET- grafer men de var svåra att konfigurera och hade begränsade funktioner. Dessutom fanns det förvånansvärt lite möjligheter att personalisera och utveckla .NET graferna så de föll bort ganska snabbt. Sedan hittade jag ett JavaScript-baserat graf verktyg som heter ”*HighCharts*” som såg väldigt lovande ut. Här fanns det goda möjligheter till att modifiera och konfigurera graferna efter eget tycke men tyvärr var detta bibliotek avgiftsbelagt för kommersiellt bruk. Slutligen stannade min forskning vid ”*GoogleCharts*”. Det är ett gratis JavaScript bibliotek för att skapa olika sorters grafer av önskad data som går att modifiera relativt enkelt beroende på hur man väljer att hämta in datan. GoogleCharts påminde överlag mycket om HighCharts även om det var nyare och mindre utvecklat än HighCharts.

Efter att det viktigaste verktyget och tillvägagångsättet var bestämt, grafen, så var det dags att fundera och planera de övriga komponenterna. De övriga komponenterna består främst av vanliga ASP.NET- element och- funktioner, så det var inget märkvärdigt med dem. Då alla komponenter var igenomtänkta och planerade så var det bara att sätta igång med att verkställa projektet.

5.2 Applikationen

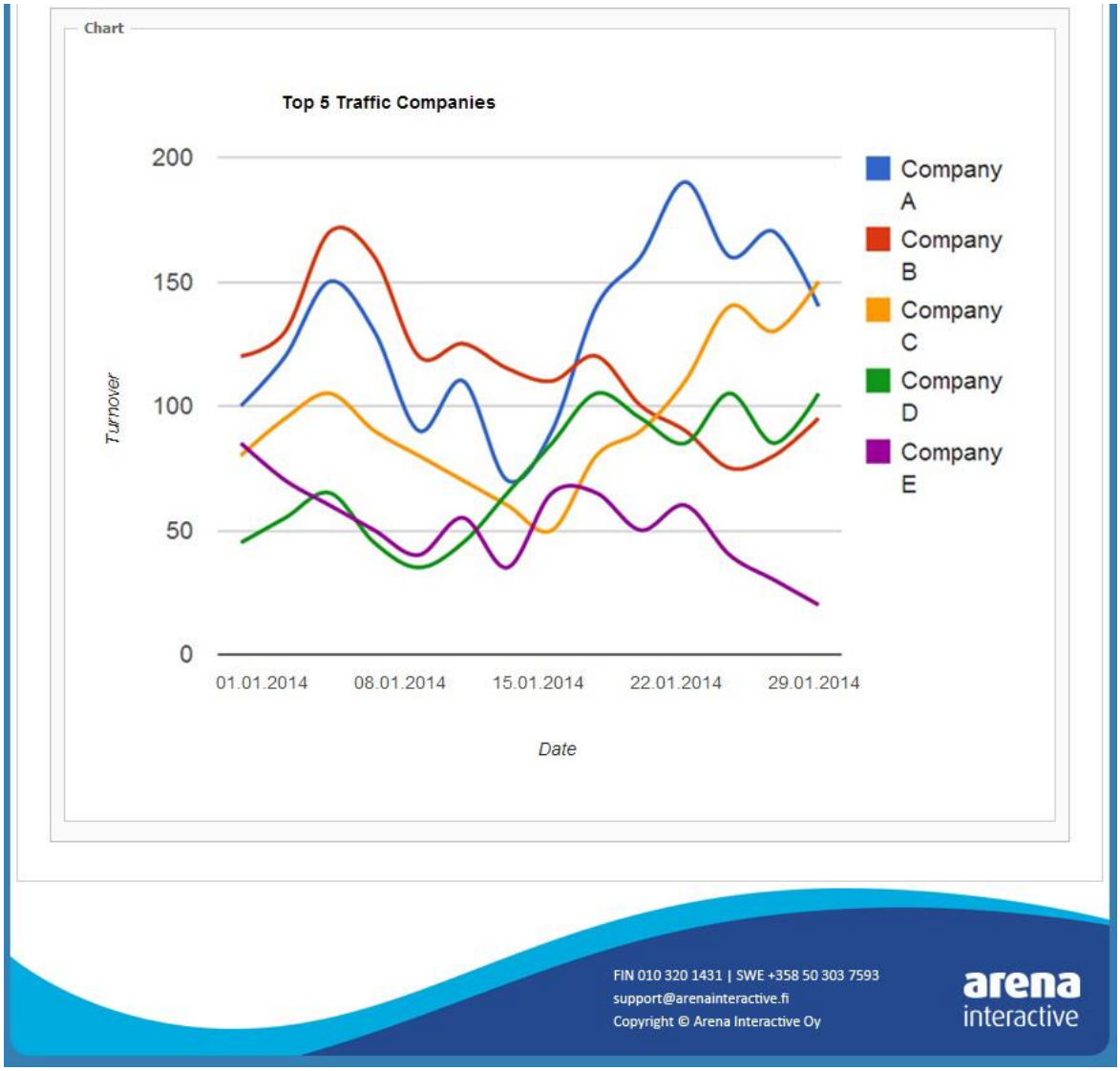
Som redan tidigare nämnts fick jag helt fria händer gällande tillvägagångssättet och illustrationen av datan samt designen av tillägget för AIMO. Jag har försökt tänka logiskt och ta med de valmöjligheter och element som behövs för att åstadkomma ett användarvänligt användargränssnitt som skall gå snabbt att använda under till exempel ett möte. Eftersom applikationen är ett tillägg till AIMO så har jag arbetat med innehållet för en webbsida och AIMO:s masterpage. Närmare information och illustration om hur master pages hittas i kapitel 3.2.5.



Figur 7. Skärmdump av AIMO och mitt tillägg för visningen av omsättningen.

I figur 7 kan man se de valmöjligheter som finns gällande datan för grafen. Med radiobuttons bör man först välja om det gäller mängden trafik eller omsättningen samt om man är intresserad av SMS- eller MMS- meddelanden. Följaktligen kan man som tillval, välja ur dropdownlistorna land, företag och ända ner till ett specifikt kortnummer. Datumintervallen kommer alltid med och standardvärde är den senaste hela kalendermånaden. Datumintervallen går naturligtvis att ändra efter önskemål beroende på vilken tid man är intresserad av följa upp grafiskt. Ifall man vill titta på en helhetsöversikt

av alla företagets kunders så väljer man inte att inte specificera land, företag eller SMS-nummer och således får man en helhetsöversikt. När man har fyllt i önskade inställningar för grafen så klickar man på generera- knappen och grafen skapas och visas för användaren. Se figur 8 nedan för exempelgraf med testdata av den slutliga produkten.



Figur 8. Skärmdump av AIMO och mitt tillägg för grafen med statistik.

I figur 7 kan man också se att på högra sidan finns en snabb-knapp som genererar vald graftyp direkt med en knapptryckning. Här har jag valt att göra färdigt bestämda grafer utgående från det företag som har genererat mest trafik och har högsta omsättningen för att man snabbt skall få fram en graf som visar relevant data, till exempel under ett möte. Här kan man även välja hur många kunder man vill visa på samma graf, ifall man vill jämföra dessa med varann.

5.3 Implementation

I detta kapitel presenteras implementationen av mitt projekt till AIMO. Här berättar jag om hur jag har strukturerat upp hela projektet och hur jag har gått tillväga. Det finns en hel del beskrivet om JavaScript eftersom det är en stor och betydande del i projektet.

5.3.1 GUI

När jag väl hade bekantat mig med min nya miljö och försökt förstå delar av AIMO så var det dags att börja med min modul. Eftersom AIMO är baserat på ASP.NET- webbsida så har den även en master page (se kapitel 3.2.4). Master pagen anger sidans yttre och inre mått gällande innehållssidorna. Master pagen innehåller sidhuvud- och fot samt menyn. Eftersom de yttre måtten för hela webbsidan var bestämt så måste jag ta hänsyn till detta och anpassa min modul innanför dessa ramar.

Jag valde att börja med den grafiska biten eftersom att jag ansåg att det är logiskt att börja med det man först ser när applikationen körs. Dessutom är det bra att få en överblick över hur koden i back enden skulle byggas upp. Vartefter GUI:n skapades så försökte jag även se vilka bitar av de tillgängliga CSS- filerna jag kunde återanvända för att undvika onödiga dubbletter i koden och för att hålla min modul så enhetlig och konsekvent som möjligt. ASP.NET- element som jag har använt är bland andra radiobuttons och dropdownlistor för att få de valmöjligheter som behövs för att framställa grafen. När jag ansåg mig ha en fullständig modul grafiskt sett så var det dags för nästa steg, att börja implementera själva funktionerna. Vartefter projektet har fortlöpt så har användargränssnittet ändrat lite hela tiden. Oftast eftersom jag har insett att något har varit otydligt eller för att GUI:n skulle vara mer logiskt och enkel för att bli så användarvänlig som möjligt.

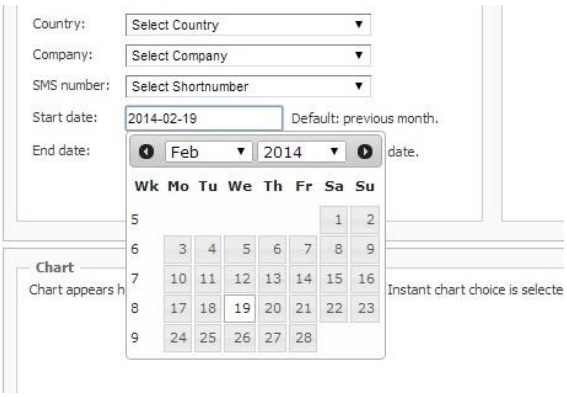
5.3.2 Datepicker

När jag var på god väg med projektet så var det dags för att ta in data om med vilket datumintervall man önskade hämta datan från databasen som skall presenteras i den slutliga grafen. Här började jag med att göra den i kod, validering, formatering och så vidare. Det visade sig vara knepigare än jag först trodde. Eftersom GoogleCharts baserar sig på JavaScript så hade jag tidigare stött på en lösning för just val av datum i .net. I kodexempel 2 ett exempel på hur koden för datepickern kunde se ut.

```
<script type="text/javascript">
    $(function () {
        $(".datepicker").datepicker({
            dateFormat: "yy-mm-dd",
            changeMonth: true,
            changeYear: true,
            showWeek: true,
            showAnim: "blind",
            firstDay: 1,
            maxDate: "0"
        })
    });
</script>
```

Kodexempel 2. Kodstycke från JavaScript delen av kalendern. Här deklarerar man i vilket datumformat som skall användas samt övriga designrelaterade inställningar.

JavaScript biblioteket för datepicker-kalendern var ett kraftfullt bibliotek med många möjligheter. Här var det enkelt att formatera datum och få fina pop-ups med en hel kalendermånad som man bara fick välja ett datum ur, istället för att manuellt fylla i hela datumet inklusive årtal i samma format som det är i databasen. I figur 9 ser vi den färdiga produkten av datepickerns kalender.



Figur 9. Skärmdump av hur datepickern ser ut i det grafiska interfacet.

5.3.3 Google Charts

Det vanligaste sättet att använda Google Charts är med ett enkelt JavaScript som man implementerar i webbsidan, se kodexempel 3 nedan. För att grafen skall ritas ut måste man även inkludera tre olika bibliotek till projektet: Google Vizualization, Google JSAPI och det tillhörande biblioteket för den önskade chart typen.

```
<script type="text/javascript">
    google.load('visualization', '1', { packages: ['corechart'] });
</script>
```

Kodexempel 3. GoogleCharts biblioteket inkluderas. /17/

Ett sätt på vilket man kan hämta datan från en server är genom webbmetoder. Webbmetoden gör ett AJAX anrop från klienten som hämtar ett JSON objekt med data. Webbmetoderna är väldigt intelligenta så man behöver inte serialisera JSON- objektet utan kan låta metoden direkt returnera listan med objekt. Se kodexempel 4 nedan som illustrerar webbmetoden. /15/

```
[WebMethod]
public static List<Data> GetData()
{
    List<Data> dataList = new List<Data>();

    dataList.Add(new Data("Column 1", 100));
    dataList.Add(new Data("Column 2", 200));
    dataList.Add(new Data("Column 3", 300));
    dataList.Add(new Data("Column 4", 400));

    return dataList;
}
```

Kodexempel 4. Webbmetoderna som hämtar data och sparar undan den som en lista med objekt. /15/

Alla grafer går att anpassa efter personliga behov och tycke, det finns många sätt att modifiera grafen till just så du vill ha den. Alla graftyperna är interaktiva och kan ändras så man uppnår det resultat man önskar. För att så många webbläsare som möjligt skall klara av att tolka och visa Google Charts så återges graferna med hjälp av HTML5/SVG- teknik.

HTML5 är alltså plattformsoberoende och möjliggör användning på både iOS X och Android som på så sätt kan alla använda Google Charts. Se Bilaga /1/ för en enkel presentation av en av Google Charts graftyper samt exempelkod. Scriptet som hämtar datan via en AJAX- metod på klientsidan från till exempel en server kunde se ut på följande vis:

```
$(document).ready(function() {
    $.ajax({
        type: 'POST',
        dataType: 'json',
        contentType: 'application/json',
        url: 'Default.aspx/GetData',
        data: '{}',
        success:
            function(response) {
                drawVisualization(response.d);
            }
        });
    })
})
```

Kodexempel 5. Java Scriptet som körs från klientsidan via ett AJAX- anrop. /15/

Slutligen funktionen som ritar själva grafen. DataTable blir initialiserat, kolumnerna blir tillsatta och datan matas in. Draw- metoden anropas och grafen ritas ut i applikationen där man har placerat den i GUI:n. I kodexempel 6 illustreras koden till en piechart. /17/

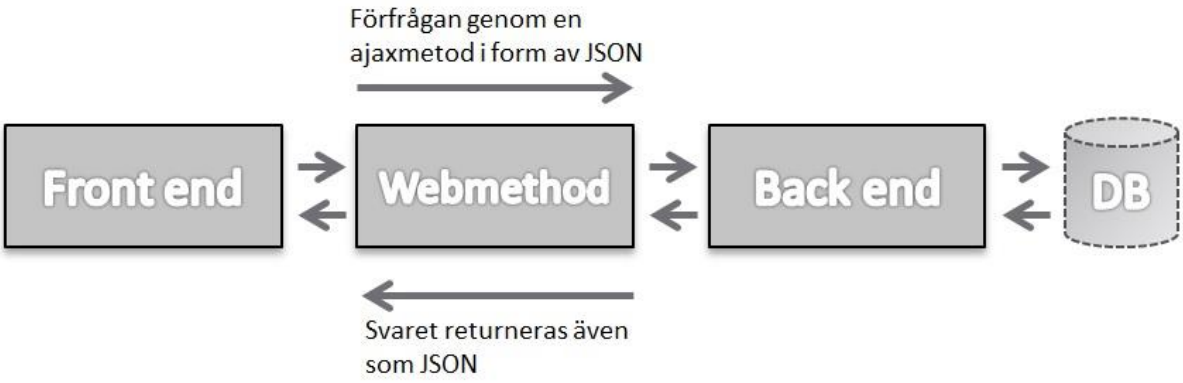
```
function drawVisualization(dataValues) {
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Column Name');
    data.addColumn('number', 'Column Value');

    for (var i = 0; i < dataValues.length; i++) {
        data.addRow([dataValues[i].ColumnName, dataValues[i].Value]);
    }

    new google.visualization.PieChart(document.getElementById('visualization')).
        draw(data, { title: "Google Charts Example" });
}
```

Kodexempel 6. Funktionen som ritar själva grafen. /15/

För att summera hela implementationen kring datan som används i grafen och för att försöka ge en bättre helhetsöverblick kan vi ta och titta på figur 10 som jag har illustrerat nedan.



Figur 10. Illustration om hur datan till grafen hämtas

Här ser vi hur datan hämtas till grafen via ajax methoder och som JSON- objekt. När användaren begär data från interfacet eller front enden så går hela processen från vänster till höger. När den önskade datan har hittats i databasen så skickas den tillbaka via back enden till frontenden, för att exponeras åt användaren i form av en tydlig och avläslig graf.

5.3.4 Databaskopplingar

Första utmaningen i back enden var att skapa databaskopplingar och Querys som hämtar den data jag behöver för att fylla mina dropdownlistor. Resultatet skulle bli egen funktion och metod för att hämta ut land, företag samt kortnummer ur databasen. Detta var tidsmässigt krävande, men även lärorikt ur ett annat perspektiv eftersom det handlade om att koppla ihop tabeller med till exempel inner join. Databas queryna byggdes och testades med hjälp av verktyget SQLYog. Efter en tid började det flyta på bättre efter att den första fungerande databasanropet var gjort. De databasanrop som har gått att återanvända så har jag använt mig av eftersom man alltid skall eftersträva att återanvända så mycket kod som möjligt och för att undvika dubbletter av liknande hämtningar. Flera av mina anrop hämtar data i form av listor med objekt som är fyllda med data.

5.3.5 Enum

För att på ett enklare sätt kunna välja vilken typ av graf man önskar generera så har jag använt mig av enums. Eftersom man med hjälp av radiobuttons väljer vilken typ av graf man önskar så måste man även sända med ett värde på dessa för att särskilja dem. Här använde jag mig av enums för att hålla reda på om det är trafik eller omsättning det är frågan om, istället för att komma ihåg och avskilja vilka värden de hade. Se kodexempel 7 för en kort illustration om hur ett enum ser ut och fungerar.

```
enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};
```

Kodexempel 7. Kodexempel på hur enums fungerar. /29/

Man kan tänka sig enum som en Array. Exempelkoden ovan betyder att Sat (lördag) får värdet 0, Sun (Söndag) får värdet 1 och så vidare. På så sätt kan man enklare hålla reda på vilken dag det är i kod med hjälp av korta definitioner som aldrig ändrar eller blir felstavat. En viktig sak att komma ihåg är att man skall hålla enums globala så att man alltid når dem oavsett var man jobbar i projektet. /29/ Se kodexempel 8 samt 9 om hur mina enums har implementerats i projektet.

```
public enum ChartTypes { Turnover = 0, Traffic = 1 };
```

Kodexempel 8. Kodexempel på hur mina enums fungerar

```
public ChartTypes ChartType { get; set; } //Turnover = 0, Traffic = 1
```

Kodexempel 9. Hur mina enums fungerar i klassen av typen ”ChartTypes”

5.4 Skript

För att hålla min mellanlagringstabell uppdaterad från de ordinarie tabellerna där all data finns så behövdes ett skript eller en applikation som sköter om det automatiskt. Det här problemet löstes genom att skapa ett program, en applikation, som körs varje natt på servern. Varför man kör den under natten är delvis för att det är då det är som minst trafik till servern och man undviker således eventuella fördröjningar för SMS-trafiken, men främst för att få ett logiskt byte till en ny beräkningsdag. Skriptet är uppbyggt så att den kontrollerar ändringar i ordinarie tabellerna och jämför den med min mellanlagringstabell. Ifall det har tillkommit ny data så kopierar den över de behövliga kolumnerna med tillhörande datan till mellanlagringstabellen.

Skriptet behandlar finansiell data och presenterar ett förkalkylerat resultat som kan data minas av den utvecklade modulen. Orsaken till att en helt egen applikation behövs för att fylla mellanlagringstabellen är enkel. Den kopierar inte bara över information, den förhands kalkylerar även omsättningen och tar i beaktande samtliga undantag. Det finns många faktorer som påverkar den slutliga omsättningen som mellanlagringstabellen innehåller. Eftersom det är många parter som vill ha sin andel av inkomsten så finns det mycket att ta i beaktande. Inte bara momsen och trafikkostnader för meddelandena utan även operatören, Arena Interactive och kunden som använder sig av AIMO skall få sin andel av intäkten som slutkunden ger upphov till när denna använder sig av till exempel en betaltjänst. Därför går det inte går att helt kallt kopiera över datan på ett enklare sätt direkt, exempelvis en schemalagd aktivitet på servern eller via SQL- kommandon. Ett annat exempel på ett problem kunde vara ifall ett SMS- meddelande överstiger 160 tecken, då blir det två meddelanden som skall beaktas som ett och med en avgift eller tariff.

5.5 Mellanlagringstabell

För att försöka undvika de långa databas queryna och för att kunna förhandskalkylera, blev det bestämt att jag skulle planera och skapa en ny databastabell för den data jag behandlar grafiskt. Jag fick fundera lite på vad den skulle innehålla innan tabellen blev färdig planerad och verkställd. Se kapitel 5.4 om applikationen som körs dagligen för att hålla min mellanlagringstabell uppdaterad med färsk data.

5.6 Säkerhet

Ett annat viktigt ämne som bör nämnas är säkerheten när man hanterar stora mängder känslig data som inte alla bör ta del av. I mitt projekt har det aldrig varit något problem eftersom modulen endast vänder sig till systemadministratörer vid Arena Interactive. Hela modulen finns implementerat i AIMO som i sin tur ligger bakom inloggning i form av personliga inloggningsuppgifter. Uppkopplingen till AIMO sker över en SSL- enkrypterad anslutning och användarkontona är dessutom skyddade med moderna hashningstekniker med individuella lösenords-salt.

Det är naturligtvis viktigt att ta säkerheten i beaktande så man inte utgör någon risk för att exponera företagsinformation och statistik för omvärlden. Det finns många sätt man kan skydda sig på idag med hjälp av antivirus, hård- och mjukvaror, brandväggar och personliga inloggningsuppgifter. Det finns alltid en risk när man är uppkopplad, frågan är bara hur väl förberedd man är och hur mycket man har satsat på att förebygga eventuella intrång.

5.7 Problem

Problem och uppförsbackar har förekommit med jämna mellanrum under projektets utveckling. De få gånger jag har strandat ordentligt har jag bett om hjälp och tips av mina handledare på Arena Interactive som har satt mig i rätt riktning igen och erbjudit lösningar kring de problem som har uppstått. Även mina vänner med insikt i programmering har fått

agerat bollplank när man har stött på mindre problem som har behövts ventileras. Eftersom JavaScript var relativt nytt för mig så har det varit ett återkommande problem att lära mig dess syntax till en början. JavaScript har även varit svårt att felsöka överlag, men där har webbläsartillägg varit till stor hjälp eftersom inte Visual Studio känner igen syntaxfel för JavaScripten. En annan sak man bör tänka på i samband med JavaScripten är att den är case-sensitive i de flesta fall, det vill säga att den inte känner igen till exempel variabelnamn som har skrivits med olika stora versaler, så det gäller att sträva till att vara så konsekvent som möjligt hela tiden när man arbetar med JavaScript. Ett annat bekymmer som har uppstått var att JavaScriptena laddades in till sidan i fel ordning och därmed fungerade inte alla funktioner korrekt. Här löstes problemet genom att använda ”defer” i script-taggen. Defer är en ypperlig funktion man borde använda oftare för att bestämma i vilken ordning JavaScriptena skall laddas in när sidan körs och således undvika eventuella problem, oavsett i vilken ordning man har inkluderat dem. Även formateringen av datum från input datan var problematiskt eftersom JSON hanterar datum på annat sätt, till exempel kunde ett vanligt datum se ut såhär i JSON: ”/Date(1224043200000)/”. Efter en del googlande så hittades en lämplig lösning. Eftersom formateringen av datum används på flera ställen så var det naturligt att skapa en egen funktion för att formatera datum från JSON till samma format som används i databasen. Se kodexempel 10 för en skärmdump om hur datumformateringen fungerar i kod. /20/

```
//converting json date to readable format
function formatDates(dates) {
    var i = 0;

    for (i = 0; i < dates.d.x.length; i++) {
        dates.d.x[i] = new Date(parseInt(dates.d.x[i].substr(6)));
    }
    return dates;
}
```

Kodexempel 10. Kodstycke vars funktion formaterar JSON- datum till ett läsligt format

SVN är en sak för sig. Version hanteringsprogrammet har inte alltid stått vid min sida men när man väl har använt det en tid så går det bra, speciellt när man börjar känna till hur det fungerar och vilka dess svagheter är. Alla mjukvaror har sina brister i funktionaliteten men en positiv sak med SVN är nog att den integreras i Windows Shell och de vanligaste programmen.

En kodrelaterad sak som dök upp vid felhantering var skillnaden mellan .any() och .count(). Det mindre bra alternativet i mitt fall var .count eftersom den räknar eller itererar igenom all data den hittar medans .any() stannar genast när den hittar någon form av data. Så det naturliga sättet för mig var att kontrollera om webbmetoderna har hämtat någon data och populerat listorna var .any() eftersom jag hanterar väldigt stora mängder data kan det ta en stund att iterera igenom alla resultat. På detta sätt sparar jag både tid och resurser, en liten optimering som kan ha stor betydelse i slutändan.

6 Resultat och Diskussion

Att få komma in i en programmerares värld utan någon större erfarenhet är minst sagt en utmaning. Det är inte en negativ sak utan mer en möjlighet att pröva och lära sig något nytt. Projektet har varit både roligt och lärorikt men ibland har det inte varit lätt att hitta lämpliga lösningar. Då har man fått stanna upp och fundera en gång till eller rentav ta ett steg tillbaka och försöka undvika roten till problemet från första början. Ofta har det löst sig rätt naturligt när man har insett att vissa funktioner eller metoder är väldigt intelligenta och förstår hur de skall hantera datan på rätt sätt, fastän koden kanske inte alltid är fläckfri.

Uppgiften var egentligen väldigt ”straight forward” och med mycket fria händer när det gällde designen och tillvägagångssättet. Designmässigt så har jag försökt få en enkel och klar struktur utan att krångla till det mer än nödvändigt. Det krävdes en del forskning innan jag hittade lämpliga lösningar som passade både den befintliga koden och mina kunskaper. Dessutom har jag hela tiden strävat till att vara så konsekvent som möjligt när det gäller sättet att koda på. Så det blir så likt som den befintliga koden som möjligt och på så sätt ger mitt tillägg en framtid om att förbättras och utvecklas. Jag har nu redan idéer för hur man kunde förbättra applikationen och förslag på hur Arena Interactive kan vidareutveckla statistiktillägget.

Eftersom det inte har varit frågan om någon slutkund i ändan av detta projekt så har jag nog inte behövt känna av någon direkt stress i samband med projektet. Men det har naturligtvis varit spännande eftersom det är erfarna programmerare och systemerare vid Arena Interactive som sedan kommer att använda detta tillägg, som jag har skapat efter deras önskemål. Om man vänder på kakan och ser det ur en annan synvinkel, så har jag haft lätt att förstå hur de önskar få detta projekt förverkligat eftersom vi pratar samma språk, till skillnad från om det varit en slutkund som inte själv är väldigt tekniskt insatt i själva programmeringen eller hur man bygger applikationer.

Vartefter projektet har fortlöpt så har mängden JavaScript ökat mer och mer. JavaScript är inget jag har varit speciellt bekant med från tidigare, utan det är något som har kommit till lite vartefter. Jag har fått lära mig att komplettera ASP.NET med JavaScript och har kanske fokuserat mera på den biten eftersom den har varit obekant sen tidigare. JavaScript är ett nyttigt skriptspråk att lära sig att använda eftersom den används på så många olika plattformar i dagens applikationer. Det har varit roligt att arbeta med JavaScript och Google Chart- delarna, eftersom det ofta går att bygga funktionerna på ett sådant sätt att de även går att återanvända till andra ändamål.

Personalen på kontoret i Vasa har alltid varit positivt inställd och har alltid gett en hjälpande hand då det har behövts. Även goda vänner har ställt upp som bollplank när man har grubblat över logik eller kodrelaterade farthinder. Personligen är jag väldigt nöjd med resultatet eftersom det blev ungefär som jag visualiserade det i projektstarten, om inte bättre. Detta har även varit det första arbetsrelaterade programmeringen som jag har gjort, så visst är det en milstolpe att utveckla och skapa något som är användbart i kod.

7 Källförteckning

1. Flanagan David (2011). *JavaScript the definite guide* ISBN: 978-0-596-80552-4

2. Sharp John (2010). *Visual C# 2010* ISBN: 978-0-7356-2670-6

3. Hand David, Mannila Heikki och Smyth Padhraic (2001). *The Principles of Data Mining* ISBN:0-262-08290-X

4. Adriaans Peter, Zantinge Dolf (1996). *Data Mining* ISBN:0-201-40380-3

5. Bertino Elisa, Catania Barbara, Piero Zarri Gian. *Intellegent Database Systems* ISBN:0-201-87736-8

6. *Arena Partners* (2014)
   http://www.arenapartners.fi (hämtad 05.01.2014)

7. *Arena Interactive, personalen* (2014)
   http://www.arenainteractive.fi/yritys/ihmiset/ (hämtad 05.01.2014)

8. *Arena Interactive, historia* (2014)
   http://www.arenainteractive.fi/yritys/historia/ (hämtad 05.01.2014)

9. *Arena Interactive, kunder och allmän information* (2014)
   http://www.arenainteractive.fi/ (hämtad 05.01.2014)

10. *Arena Interactives flaggskepp AIMO* (2014)
    www.aimomedia.com (hämtad 05.01.2014)

11. *Front end vs back end* (2012)
    http://skillcrush.com/2012/04/17/frontend-vs-backend-3/ (hämtad 05.01.2014)

12. *SaaS* (u.å)
http://www.webopedia.com/TERM/S/SaaS.html (hämtad 05.01.2014)

13. *SQLYog klient* (u.å)
https://www.webyog.com (hämtad 05.01.2014)

14. *Sublime2 textediteringsprogram* (u.å)
http://www.sublimetext.com/docs/2 (hämtad 05.01.2014)

15. *Google Charts ASP.NET blogg* (2013)
http://girlfromoutofthisworld.com/google-charts-asp-net-jquery-ajax-quick-easy/
(hämtad 05.01.2014)

16. *TortoiseSVN version hanteringsprogram* (u.å)
http://tortoisesvn.net/docs/release/TortoiseSVN_en/index.html
(hämtad 05.01.2014)

17. *Google Charts* (2013)
https://developers.google.com/chart (hämtad 02.02.2014)

18. *From Data mining to KDD* (1997)
http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf
(hämtad 02.02.2014)

19. *Sophos VPN klient* (u.å)
http://www.sophos.com/en-us/products/secure-vpn.aspx (hämtad 15.02.2014)

20. *JSON objekt exempel* (u.å)
http://www.w3schools.com/json/json_syntax.asp (hämtad 15.02.2014)

21. *JSON* (u.å)
http://json.org/ (hämtad 15.02.2014)

22. *Intro to data mining* (2013)
http://ecognomy.blogspot.fi/2013/12/intro-to-data-mining-for-marketers-part_13.html (hämtad 15.02.2014)

23. *ASP, ASP.NET* (u.å.)
http://www.indiabix.com/technical/dotnet/asp-dot-net (hämtad 18.02.2014)

24. *ASP.NET master pages* (u.å)
http://www.cleverworkarounds.com/2007/10/08/sharepoint-branding-how-css-works-with-master-pages-part-1/ (hämtad 18.02.2014)

25. *Web forms MVC* (u.å)
http://msdn.microsoft.com/en-us/library/ff649643.aspx (hämtad 23.02.2014)

26. *MySQL* (u.å)
http://www.mysql.com/about (hämtad 18.02.2014)

27. *CSS, Cascading Style Sheet* (u.å.)
http://webdesign.about.com/od/beginningcss/a/aa021607.htm (hämtad 18.02.2014)

28. *MySQL - MariaDB* (u.å)
https://mariadb.com/kb/en/about-mariadb/ (hämtad 20.02.2014)

29. *Enumerators, C#* (u.å)
http://msdn.microsoft.com/en-us/library/sbbt4032.aspx (hämtad 23.02.2014)

30. *Data mining och datahistoria* (2012)
http://whatsthebigdata.com/2012/04/26/a-very-short-history-of-data-science/
(hämtad 15.02.2014)

31. *TortoiseSVN användardiagram* (2014)
http://hosting.com/support/upload/working-with-an-svn-repository-using-tortoise-svn/ (hämtad 05.01.2014)

```
<html>
  <head>
    <!--Load the AJAX API-->
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">

      // Load the Visualization API and the piechart package.
      google.load('visualization', '1.0', {'packages':['corechart']});

      // Set a callback to run when the Google Visualization API is loaded.
      google.setOnLoadCallback(drawChart);

      // Callback that creates and populates a data table,
      // instantiates the pie chart, passes in the data and
      // draws it.
      function drawChart() {

        // Create the data table.
        var data = new google.visualization.DataTable();
        data.addColumn('string', 'Topping');
        data.addColumn('number', 'Slices');
        data.addRows([
          ['Mushrooms', 3],
          ['Onions', 1],
          ['Olives', 1],
          ['Zucchini', 1],
          ['Pepperoni', 2]
        ]);

        // Set chart options
        var options = {'title':'How Much Pizza I Ate Last Night',
                       'width':400,
                       'height':300};

        // Instantiate and draw our chart, passing in some options.
        var chart = new google.visualization.PieChart(document.getElementById('chart_div'));
        chart.draw(data, options);
      }
    </script>
  </head>

  <body>
    <!--Div that will hold the pie chart-->
    <div id="chart_div"></div>
  </body>
</html>
```