Ett Content Management-system med Web 2.0

Eveliina Rytkönen

Examensarbete för Tradenom (YH)-examen

Utbildningsprogrammet för Informationsbehandling

Raseborg 2011

EXAMENSARBETE

Författare: Eveliina Rytkönen

Utbildningsprogram och ort: Informationsbehandling, Raseborg

Handledare: Klaus Hansen

Titel: Ett Content Management-system med Web 2.0

Datum 1.2.2011 Sidantal 40 + 1 Bilagor 1

Sammanfattning

Detta examensarbete behandlar en produktutveckling – ett CMS eller innehållshanteringssystem och webbplats, åt firman Marryt. Målet har varit att skapa grunden till en enkel webbplats i enlighet med Web 2.0 standarder.

Fokuset för arbetet ligger på utvecklingen av enkla Web 2.0-funktioner i koppling till innehållshanteringssystemet. Grundprinciperna bakom de språk, gränssnitt och tekniker som används här - XML, DOM, JavaScript och Ajax, samt hur de relaterar till varandra förklaras kort. Element av HTML, CSS och PHP berörs.

Utvecklingen av webbplatsens externa design och dess relation till Web 2.0 beskrivs.

Språk: Svenska Nyckelord: innehållshanteringssystem, XML, Web 2.0, DOM, JavaScript, Ajax

---

OPINNÄYTETYÖ

Tekijä: Eveliina Rytkönen

Koulutusohjelma ja paikkakunta: Informationsbehandling, Raasepori

Ohjaaja: Klaus Hansen

Nimike: Ett Content Management–system med Web 2.0 /
Sisällönhallintajärjestelmä Web 2.0:lla

Päivämäärä 1.2.2011 Sivumäärä 40 + 1 Liitteet 1

Tiivistelmä

Tämä opinnäytetyö käsittelee Marryt –yrityksen sisällönhallintajärjestelmän ja verkkosivujen tuottamista tuotekehityksenä. Tavoitteena on ollut Web 2.0 -standardien mukaisten yksinkertaisten verkkosivujen perustan luominen.

Työn painopiste on yksinkertaisten Web 2.0 -funktioiden luominen sisällönhallintajärjestelmän yhteydessä. Työssä selitetään lyhyesti käytettävien kielten, rajapintojen ja tekniikoiden - joita ovat XML, DOM, JavaScript ja Ajax - periaatteet ja niiden liittyminen toisiinsa. Myös osia HTML:ää, CSS:ää sekä PHP:tä kosketetaan.

Verkkosivujen ulkoisen designin kehittäminen ja sen yhteys Web 2.0:aan kuvataan.

Kieli: Ruotsi Avainsanat: sisällönhallintajärjestelmä, XML, Web 2.0, DOM, JavaScript, Ajax

---

BACHELOR'S THESIS

Author: Eveliina Rytkönen
Degree Programme: Business Information Technology
Supervisor: Klaus Hansen

Title: A Content Management System using Web 2.0 / Ett Content Management–system med Web 2.0

Date 1 February 2011 Number of pages 40 + 1 Appendices 1

Abstract

This thesis deals with a product development - a CMS and website, for Marryt, a firm. The goal has been to create the foundations for a simple website in accordance with Web 2.0 standards.

The focus of the thesis is the development of simple Web 2.0 functions in connection with the CMS. The basic principles behind the languages, interfaces and techniques used - XML, DOM, JavaScript and Ajax, and how they relate to each other, are briefly explained. Elements of HTML, CSS and PHP are touched upon.

The development of the website's external design and its relation to Web 2.0 are described.

Language: Swedish Key words: CMS, XML, Web 2.0, DOM, JavaScript, Ajax

---

Innehållsförteckning

1 Inledning 1

1.1 Bakgrund 1

1.2 Målsättning och syfte 1

1.3 Verktyg 2

2 Kravspecifikation 3

2.1 Extern design 3

2.2 Intern design 4

2.3 Begränsningar 4

3 Projektets faser 4

3.1 Definition 4

3.2 Design 5

3.3 Konstruktion 5

3.4 Test 6

3.5 Dokumentation 6

4 Planer för verkställande 7

5 Extern design 8

5.1 Web 2.0 8

5.2 Indexsidan 9

5.3 CSS 9

5.4 Logotyp 10

5.5 Grafisk profil 11

5.6 Meny 12

5.7 Innehåll 13

5.8 Sidfot 13

6 Intern design 14

6.1 Säkerhet 15

6.1.1 Logga in/ut 15
6.1.2 Kalender 16
7 Innehållshanteringssystem 16
7.1 Bakom sidinnehållet 16
7.1.1 Varför XML? 17
7.1.2 XML-dokument 17
7.1.3 DOM 19
7.1.4 JavaScript 19
7.1.5 Ajax 20
7.2 Visa sidinnehåll 21
7.2.1 Kunddel 21
7.2.2 Administratörsdel 22
7.3 Editera sidinnehåll 24
7.4 Skicka sidinnehåll 25
7.5 Spara sidinnehåll 26
8 Resultat 28
9 Reflektion och kritisk granskning 29
9.1 Tidsplanen 29
9.2 Tekniken 30
9.3 Arbetet 31
9.4 Andra kommentarer 32
Källförteckning 33
Bilaga 1 Kravspecifikation

1 Inledning

Detta examensarbete är utfört vid Yrkeshögskolan Novias utbildningsprogram för informationsbehandling. Utbildningen leder till en tradenomexamen. Examensarbetet förverkligas i form av en produktutveckling – en webbplats beställd av och för firman Marryt.

Detta dokument beskriver examensarbetsprocessen. Inkluderat finns information om projektets bakgrund samt de olika projektfaserna. Produktutvecklingen beskrivs och väsentlig kod förklaras. I slutet reflekteras över arbetsprocessen, produkten och lärandet.

1.1 Bakgrund

Företaget Marryt grundades år 2004 av Martti Rytkönen. Firmans huvudverksamhet är utbildning med fokus på säkerhet. Marryt erbjuder följande utbildning;

- arbetssäkerhetskort: grundkurs och fortbildningskurs
- certifikat för heta arbeten
- tak- och vattenisoleringscertifikat för heta arbeten
- förstahandssläckningsutbildning

Firman är stationerad i Lappvik, Hangö men utbildningstillfällen hålls runtom i Finland. Marryt drivs vid sidan om annat arbete som en bisyssla.

Marryt har från tidigare ingen webbplats. Firman drivs från ett privathem och har därmed inget kontor som kunder kan besöka. Företaget skickar offerter åt potentiella kunder via brev eller e-post och uppmanar dem att kontakta firman inom samma medium. Klienter kan också ta kontakt via telefon. Det här innebär att potentiella kunder inte känner till firman, om den inte tagit kontakt med dem först och många potentiella kunder går förlorade.

1.2 Målsättning och syfte

Målet för examensarbetet är att utveckla grunden till en enkel webbplats enligt Web 2.0-standarder. Sidorna skall använda Web 2.0-basfunktioner och fungera som en grund att

bygga vidare på. Sidorna skall hållas till Web 2.0-principer både vid den externa och interna designen. Webbplatsen skall innehålla en administratörsdel där sidinnehållet kan editeras utan att sidan laddas om. Den uppdaterade informationen skall alltså skickas asynkront i bakgrunden med Ajax. Detta mål verkställs i form av detta projekt.

Uppdragsgivarens huvudmål med examensarbetet är att få en webbnärvaro för Marryt. Detta sker i form av webbplatsen. Webbplatsen skall vara en plats där kunderna kan komma åt information om företaget Marryt. Sidorna skall innehålla information om firman, allmän information om de erbjudna kurserna, en lista över kommande utbildningstillfällen, länkar samt kontaktinformation. Förutom kontaktinformationen fås de ovannämnda punkterna endast genom att kontakta Marryt. All denna information skall vara tillgänglig på webbplatsen.

Webbplatsen är egentligen ett CMS (Content Management System) eller innehållshanteringssystem. Detta innehållshanteringssystem skall göra det möjligt för uppdragsgivaren att editera informationen på webbplatsen utan någon kunskap om HTML. Användaren skall alltså inte behöva gå in i koden för att editera sidinnehållet.

Examensarbetets syfte är att utveckla en enkel Web 2.0-kod. Koden bearbetas inte vidare under detta projekt men kan lätt utvecklas senare. Syftet med webbplatsen för uppdragsgivaren är att informera kunder om kommande kurser samt varför de behövs, alltså vilken nytta kunden har av att delta i dem. Webbplatsen skall ge kunderna den information de behöver för att kunna kontakta Marryt för att anmäla sig till en kurs eller be om mera information.

1.3 Verktyg

Marryts webbplats verkställs med två program inom Adobe familjen; Adobe Photoshop CS3 och Adobe Dreamweaver CS3. I Photoshop utformas webbplatsens grafiska design. Då denna är klar används Dreamweaver och Notepad ++ för att bygga upp motsvarande sida med kod. Notepad ++ är en texteditor riktad mot programmerare medan Dreamweaver är en text och grafisk editor av webbsidor.

Mozilla Fiefox–webbläsarens Firebug-tillägg används för att underlätta kodningsprocessen samt för att utreda möjliga problem. Webbplatsen analyseras och testas med YSlow-tillägget. Wampserver 2.0 är en mjukvara som tillåter användaren lägga upp en virtuell

server på den lokala maskinen. Servern används för att testa sidorna under utvecklingsprocessen. Webbläsarna Google Chrome 8, Mozilla Firefox 3.6 och Internet Explorer 8 används för att kontrollera webbsidans design och funktionalitet under projektet.

2 Kravspecifikation

Kravspecifikationen är ett dokument som beskriver uppdragsgivarens krav på projektet. Kravspecifikationen för detta projekt finns i bilaga 1. I de underliggande styckena gås det in i mera detalj för de olika delarna; extern design, intern design och begränsningar. Kravspecifikationen utarbetades i samarbete med Marryt i juli 2010 under ett antal möten. Den består utöver de tidigare nämnda punkterna även av information om hurdana funktionella och icke-funktionella krav Marryt ställer. Nedan beskrivs de centrala kraven kort i punktformat;

- Enkel navigation
- Mörk text på ljus bakgrund
- Synlig text på finska (gäller inte koden)
- Innehållshanteringssystem

2.1 Extern design

Uppdragsgivaren har inga krav på den grafiska profilen. Firman har ingen logo och denna måste skapas först – webbplatsen utvecklas runt den. Webbplatsens utseende planeras preliminärt genom ett antal skisser. Av dessa väljer uppdragsgivaren ett par vilka vidareutvecklas till prototyper i grafiskt format. Uppdragsgivaren väljer en av dessa prototyper och den används som modell för webbplatsens externa design.

Uppdragsgivaren kräver att webbplatsen har enkel navigation. Enkel definieras här som att innefatta maximalt två nivåer i menysystemet. Den önskade informationen nås med högst två knapptryck. I menysystemet inkluderas inga expanderande menyer, alltså menyer som öppnas då man för musen över dem.

Texten på webbplatsen är lätt att läsa; den består av mörk text på ljus bakgrund. Webbplatsens officiella språk är finska. All synlig text, menyer och text på kund- och administratörssidan är på finska.

2.2 Intern design

Webbplatsen innefattar en administratörsdel med innehållshanteringssystem. Administratörsdelen kräver inloggning och har grundläggande säkerhet. Administratören har möjlighet att editera sidinnehållet. Med sidinnehåll avses här titel samt brödtext på webbplatsens sidor.

Webbplatsen är tillgänglig med olika webbläsare. Sidorna ser lika ut och funktionaliteten förblir oförändrad oberoende av vilken webbläsare som används. Användaren ser ingen skillnad mellan sidorna då de öppnas i t.ex. Mozilla Firefox och Google Chrome.

2.3 Begränsningar

Administratören kan inte lägga till nya användare, editera menyn, färgerna, bakgrunden eller något annat utöver sidtitlarna och brödtexten. Uppdragsgivaren har inga krav gällande de programmeringsspråk eller metoder som används. Examensarbetet innefattar inte någon manual, på administratörssidorna finns dock en kort guide till innehållshanteringssystemet.

3 Projektets faser

Detta examensarbete kan delas in i fem egentliga delar; definition, design, konstruktion, test samt dokumentation. Projektfaserna fortlöper i kronologisk ordning och tidvis parallellt med varandra.

3.1 Definition

Arbetet börjar i sin enklaste form i april 2010 då ett examensarbetskontrakt görs upp mellan Marryt och den studerande. En representant från Yrkeshögskolan Novia godkänner i detta skede arbetsplanerna.

Under den förberedande fasen i maj - juni 2010 repeteras webbprogrammeringens grunder. Böckerna *Kom igång med HTML* (Hayes, 2006) och *Kom igång med CSS* (Weakley, 2006) läses för detta ändamål. De innehåller många av de grundläggande principerna inom dessa ämnen och uppfriskar minnet inför examensarbetet. Den studerande läser också artikeln *30 HTML Best Practices for Beginners* (Way, 2009) som är mycket informativ. Alla punkter i artikeln är inte relevanta för projektet men artikeln - liksom böckerna - behandlar grunderna.

En tidsplan och projektplan utvecklas inför Idéseminariet den 9.9.2010. Kravspecifikationen utvecklas i samband med arbetes inledning i slutet av juli 2010. Den 13.9.2010 hålls ett handledningsmöte mellan den studerande och handledaren. Vid detta möte utreds projektets framgång och projektets utsträckning definieras ytterligare. Efter mötet utvecklas projektplanen vidare. Varje steg av projektet leder till en justering av de följande delarna av projektplanen. Vid mellanseminariet den 19.10.2010 görs de sista justeringarna till projektplanen baserade på kommentarer gjorda vid tillfället, samt återstående arbetstid.

3.2 Design

Efter diskussioner gällande krav och önskemål med Marryt väljer uppdragsgivaren mellan ett antal skisser på möjliga webbplatser. De valda skisserna bearbetas i Adobe Photoshop CS3 och uppdragsgivaren väljer bland dessa prototyper grunden till webbplatsens design. Den valda prototypen vidareutvecklas i Photoshop och webbplatsens externa design bestäms. Den interna designen - webbplatsens underliggande struktur, utvecklas parallellt med den externa designen. Då dessa är klara övergår arbetet till Dreamweaver och Notepad ++. I Dreamweaver realiseras den externa och interna designen med HTML, PHP, JavaScript, Ajax och CSS.

3.3 Konstruktion

Funktionaliteten utvecklas parallellt med designen efter att HTML-stommen lagts för webbplatsen. Den grundläggande funktionaliteten åstadkoms delvis i ett tidigt skede men måste omarbetas upprepade gånger innan en fungerande lösning hittas. Projektets fokus övergår helt till funktionaliteten i slutet av augusti 2010 då designen är klar, frånsett detaljer.

3.4 Test

Webbplatsen testas ingående under konstruktionsfasen och genomgår en slutlig testning efter att koden anses färdig. På basen av dessa test görs vissa åtgärder för att förbättra webbplatsens funktionalitet.

DTD (Document Type Definition) eller doctype läggs alltid till i början av HTML-dokument eftersom denna deklaration berättar vilka regler som skall användas vid tolkning av dokumentet. Detta i sin tur gör det möjligt för webbläsaren att visa innehållet rätt. DTD är viktigt eftersom koden kan vara korrekt, men ändå inte fungera, eller visas som fel vid testning, om den inte motsvarar den deklarerade doctypen. Webbplatsen i fråga använder DTD XHTML 1.0 Transitional för att hålla koden till XML-standard.

YSlow av Yahoo är ett Mozilla Firefox-tillägg som integreras med Firebug-tillägget för samma webbläsare. På basen av ett antal förutbestämda regler utvärderar YSlow webbplatsen. Användaren kan välja mellan tre alternativ med olika regler, baserat på vad som passar webbplatsen i fråga. YSlow ger vitsord baserade på fem olika delar; innehåll, css, bilder, javascript och server. Det finns information tillgänglig om de olika delarna samt hur möjliga problem eller låga vitsord kan korrigeras. I detta projekt används alternativet ”Small Site or Blog” vid val av regler. YSlow hittar då det körs två punkter som är underkända i serverdelen. YSlow rapporterar en iframe utan länk. Vid kontroll visar sig denna iframe ligga i den externa editor-koden. Det andra problemet gäller filer som behöver komprimeras. Detta problem härstammar från det att filerna inte i testskedet överförts till en extern server. Då detta görs komprimeras filerna.

Webbplatsens kod testas med Dreamweavers interna validerare under utvecklingen. För vidare kontroll används W3C Markup Validation Service och W3C CSS Validation Service. Förutom detta ingår ingen separat testning av webbplatsen i bruk i projektet. Denna del av testningen sker av uppdragsgivaren efter examensarbetes slutförande, då webbplatsen tagits i bruk. Möjlig vidareutveckling på basen av denna testning hör inte till projektet och avtalas separat.

3.5 Dokumentation

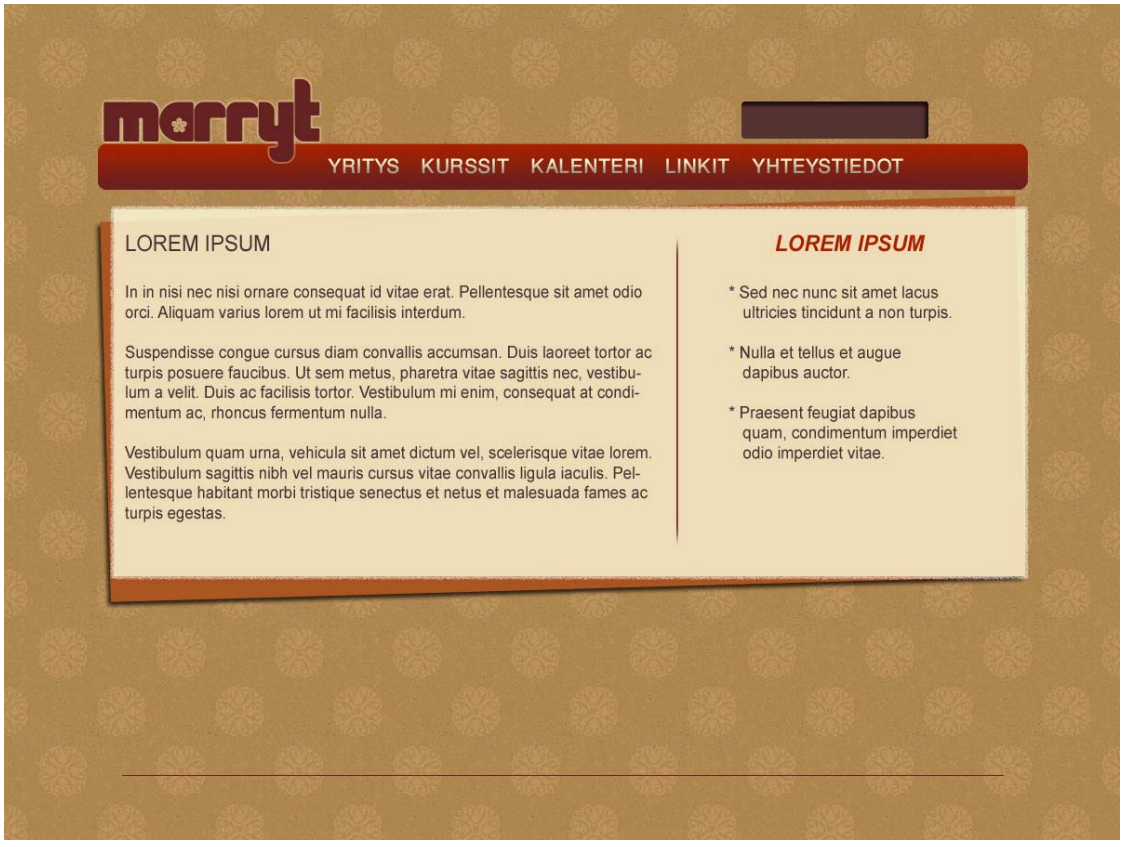
Dokumentationen fortlöper under projektets gång i form av anteckningar, sparade länkar samt andra varierade dokument. Projektplanen och tidsplanen skrivs i början av projektet.

Examensarbetets skriftliga del påbörjas under webbplatsens verkställande och avslutas efter att webbplatsen blivit klar. I detta skede används Theseus - en internetbank av publicerade examensarbeten, för att ge en bild av hur andra examensarbeten utformats.

4 Planer för verkställande

Verkställandet av webbplatsen är uppdelad i design och funktionalitet. Denna del av examensarbetet utreder först hur webbplatsens yttre - utseendet, utvecklas och fortsätter sedan till funktionaliteten. I verkligheten sker en stor del av utformnings- och funktionalitetsarbetet parallellt under projektets lopp.

Webbplatsens användargränssnitt - den HTML-kod som ligger till grund för hela webbplatsen, byggs upp innan funktionalitetens utveckling påbörjas, men efter det sker arbetet jämsides. Under arbetets lopp upptäcks problem med den originella formgivningen – den externa designen, och då funktionaliteten är klar ställs dessa problem till rätta och webbsidans design finslipas. Det första Photoshop-utkastet till den externa designen syns i figur 1.



Figur 1. Första utkastet till den grafiska designen.

5 Extern design

Figur 2 visar webbplatsens framsida med testinformation inlagd. Fokuset är på sidinnehållet - ju färre objekt det finns på sidan, desto mindre avleds kunden från den essentiella informationen. Ett av uppdragsgivarens krav på webbplatsen är att den skall vara enkel till utseendet; alltså relevant information skall framhävas medan allt sådant som kan tänkas distrahera besökaren minimeras. Dessa krav gör sig väl för en Web 2.0-utformning av webbplatsen. Med artikeln *Web 2.0 How-To Design Style Guide* som påminnelse och guide till designprinciperna för Web 2.0, byggs sidan upp i Adobe Photoshop CS3. Kund- och administratörsdelen är identiska till utformningen, skillnaden ligger i innehållet.

Figur 2. Webbplatsens slutliga indexsida med testinformation.

5.1 Web 2.0

Marryts klaraste krav från början har som tidigare nämnts varit att webbplatsen skall vara klar och enkel. Web 2.0 innebär egentligen ingenting mera än modern webbdesign. Man kan säga att Web 1.0-webbsidor är webbsidor från 90-talet – sidor som går ut på att inkludera så mycket information, bilder och annat lockande material som möjligt. Sidorna är fulla med objekt utan någon direkt koppling till sidinnehållet, och det kan vara svårt att hitta det som är relevant. Ett exempel på detta är söktjänster. Den moderna söktjänsten

består av en ensam sökruta centralt i det annars tomma webbläsarfönstret, medan söktjänster tidigare omringades av nyheter, väder, reklam, bilder och så vidare. Web 2.0 understryker enkelhet. I artikeln *Web 2.0 How-To Design Style Guide* definieras Web 2.0 enligt följande;”*2.0 design means focused, clean and simple*”.

5.2 Indexsidan

Alla de olika komponenterna på sidan ligger under en container div. Denna div centreras i webbläsarfönstret för ett modernt - Web 2.0 enligt, utseende. Sidhuvudet - den så kallade headern, innehåller logon och menyn. För att följa den moderna stilen är denna tydligt definierad som en separat del. Menybalken skiljer sig från resten av sidan med hjälp av sin distinkta färg. Textrutan och sidfoten är också klart separata delar av helheten.

Webbplatsen använder white space - tomrum, för att framhäva innehållet. I figur 2 kan white space definieras som den randiga bakgrunden, utrymmet mellan länkarna i menyn, utrymmet mellan textrutan och texten, utrymmet mellan texten och titeln samt utrymmet mellan textparagraferna. Detta utrymme mellan de olika delarna framhäver dem. Utan white space flyter de olika delarna ihop och sidan blir svårförstådd. Tomrum används ofta i traditionella media för att ge en känsla av lyx; många märken har använt sig av reklam i tidningar där annonsen består av en helt tom sida, med undantag av namnet på märket. Tomrum gör webbplatsen angenäm för användaren att se på - det leder ögat till innehållet.

Sidan har ett sluttande randigt bakgrundsmönster. Mönstret genereras med Stripe Generator 2.0 beta. Denna applikation skapar på basen av ett antal alternativ en bild i PNG (Portable Network Graphics) -format. Denna bild kan sedan upprepas över skärmen vilket skapar ett mönster, i detta fall ränder. En enfärgad bakgrund får sidan att se platt och livlös ut. Stora bilder däremot är opraktiska och avleder från innehållet. Det gråbrun- och rödrandiga bakgrundsmönstret ger sidan en intressant bakgrund utan att besökaren distraheras från innehållet.

5.3 CSS

För att minimera problem med olika webbläsare läggs en så kallad reset - eller ställ om-funktion, till i början av CSS-filen. Detta definierar utgångsvärdena för CSS-filen. Det finns ingen gemensam standard för CSS-utgångsvärden utan de varierar från webbläsare

till webbläsare. Webbsidorna ser alltså olika ut i olika webbläsare eftersom de har olika utgångspunkter. Då alla värden nollställs i början av CSS-filen visas elementen på sidan precis så som de definieras i koden oberoende av vilken webbläsare som används. En standard textparagraf i en div har med Mozilla Firefox automatiskt tomrum mellan divens övre och nedre kant samt texten. I Internet Explorer finns inget sådant tomrum. De olika webbläsarna tolkar alltså samma kod på olika sätt utan någon direkt orsak. Detta är viktigt då webbplatsen önskas se lika ut oberoende av vilken webbläsare kunden använder. Används inte någon form av reset kan man vara tvungen att göra upp en skild CSS-fil för varje webbläsare. Beroende på vilka värden man vill nollställa väljer man passlig reset-kod. I tabell 1 har marginalen och utfyllnaden för hela webbplatsen nollställts med en av de enklaste reset-koderna. Definitioner av marginal och utfyllning senare i CSS-filen skriver över denna nollställning. På detta sätt kan man skapa ett universalt utseende oberoende av vilken webbläsare som används.

Tabell 1. CSS reset.

```
1  *
2  {
3  margin:0;
4  padding:0;
5  }
```

5.4 Logotyp

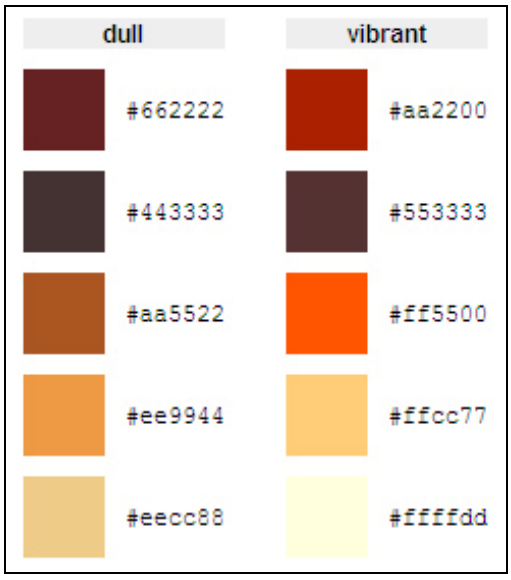
Marryt har från tidigare ingen webbplats, broschyr eller annat material som kunde användas som grund för webbplatsen. Därför inleds arbetet med att framställa en logotyp. Logotypen i sin tur fungerar som inspiration till resten av webbplatsen. Genom experiment i Photoshop tillverkas den slutliga logon för webbplatsen. Logotypen är helt uppbyggd av gemener med typsnittet chick och med en stroke-effekt. Färgen justeras efter valet av färgskala från en icke-specificerad röd till den röda färgen definierad med hexadecimal-koden #662222.

Figur 3. Logotypen på mörk bakgrund.

5.5 Grafisk profil

I samband med logotypen utreds även webbplatsens färgschema. I detta skede framställs essentiellt Marryts grafiska profil. Med tanke på att Marryts fokus ligger på kurser speciellt inom brandsäkerhet är färgerna rött, brunt och orange en naturlig utgångspunkt. Färgerna påminner om en solnedgång och en grann bild på detta hittas via Google. Color Palette Generator (DeGraeve) är en webbsida som på basen av en bild sammanställer ett färgschema. Genom att mata in URL-adressen på solnedgångsbilden på webbsidan sammanställs ett färgschema baserat på bilden.

Färgschemat består av tio färger: fem matta och fem granna. Dessa visas i figur 4 och utgör webbplatsens färgschema.



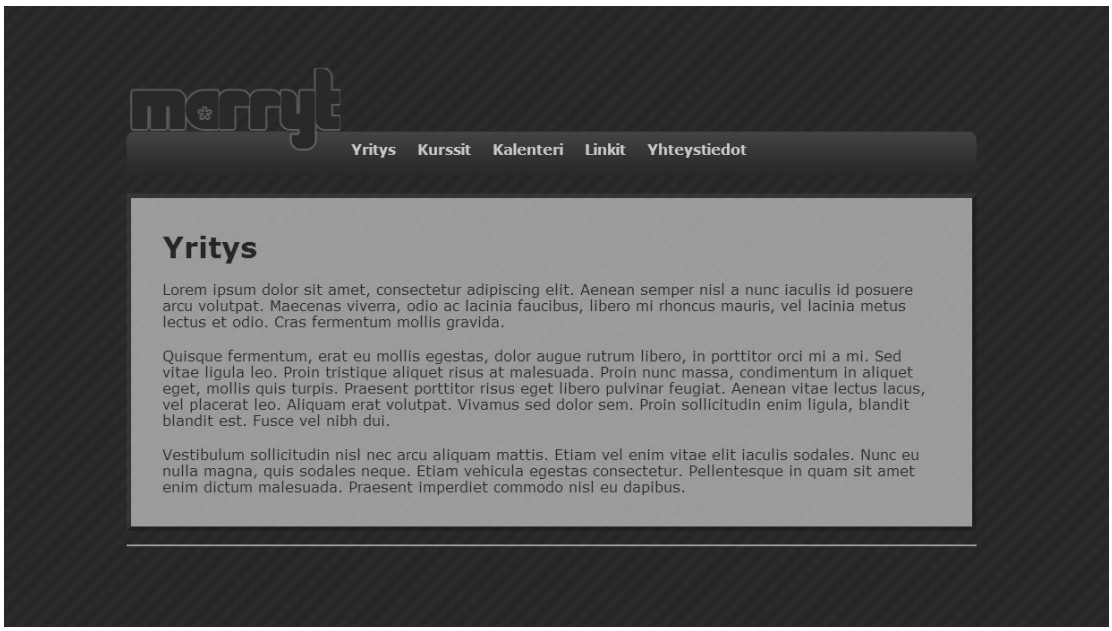
Figur 4. Skärmbild på färgschemat genererat av Color Palette Generator.

5.6 Meny

Det är viktigt att navigationen tydligt åtskiljs från resten av sidan. Tomrum används för detta men även färger kan vara nyttiga i detta sammanhang. Menybalken består av en linjär gradient i rött. Detta görs för att ge sidan djup – att göra den mindre platt i användarnas ögon. Gradienter är också nyttiga eftersom de hjälper till att mjuka upp intrycket. I figur 5 syns graderingen av menybalken.

Figur 5. En del av menybalken med gradering.

För navigationen lyder ”less is more”. En klar lista på de olika delarna av sidan fungerar bäst både ur design- och funktionalitetsvinkel. Menylänkarna måste vara tydliga, därför består de av text i en ljus färg som står ut mot den röda bakgrunden. Problem kan uppstå med användare som lider av någon grad av färgblindhet. I boken *Verkkografiikkaa* (Hatava, 2003, 68) nämns att det bästa sättet att kontrollera sidans funktionalitet för alla, är att se på webbplatsen i svartvitt format. I figur 6 syns webbplatsen i sin helhet i svartvitt. De olika delarna syns utan problem.



Figur 6. Webbplatsens indexsida med testinformation i svartvitt.

5.7 Innehåll

Sidinnehållet är - tillsammans med menyn - den viktigaste delen av webbplatsen. I *Verkkografiikkaa* (Hatava, 2003, 68) berättar Hatava att ljus text på mörk bakgrund kan anses vara bättre på grund av ett antal orsaker men att tidigare undersökningar inte har hittat någon betydande skillnad mellan ljus text på mörk bakgrund och mörk text på ljus bakgrund. Hon konstaterar också att konsistens är viktigare än vilken modell man väljer eftersom ögat vänjer sig vid en viss stil.

Webbplatsen för Marryt använder mörk text på en ljus bakgrund. Uppdragsgivaren kräver detta från början. Färgerna svart och vitt används dock inte, eftersom de ger ett hårt intryck och inte lockar ögat. Hatava skriver om detta i *Verkkografiikkaa* (2003, 16) så här; ”Suurin kontrasti ei kuitenkaan välttämättä tarkoita parasta mahdollista kontrastia. Silmät väsyvät jos on koko ajan ponnisteltava äärirajoilla”. Den största kontrasten är alltså inte nödvändigtvis den bästa eftersom ögonen blir trötta om de hela tiden anstränga till gränsen.

En mörk grå (#443333) ur färgschemat mot en lätt genomskinlig ljus (#ffffdd) bakgrund gör en stor skillnad även om det är något de flesta användare inte lägger märke till. Färgerna är inte lika extrema vilket ger ett mjukare intryck och gör texten trevligare att läsa.

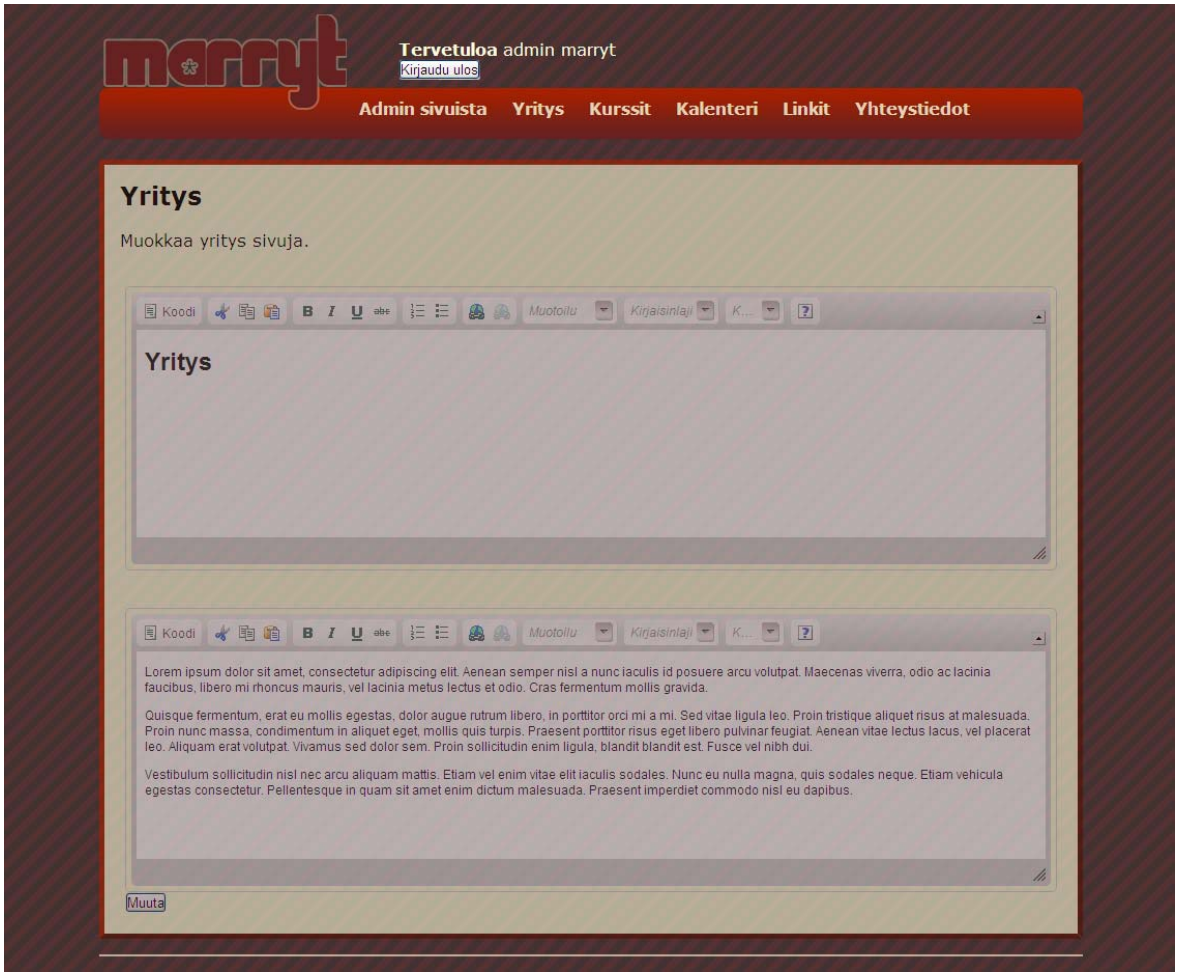
Så kallade rich surfaces eller rika ytor hjälper webbplatsen att se mera färdig och vidrörbar ut. Uttrycket syftar till 3D-effekter och effekter som låter bilden på skärmen påminna om någon verklig yta. Plast och metall är populära ytor att återskapa. Ett sätt att göra detta är genom reflektioner och speglingar eller skuggor. Den ram som omringar textrutan i figur 8 är ett exempel på detta. Ramen i rött (#aa2200) markerar gränserna för textrutan.

5.8 Sidfot

Sidfotens – eller footerns - uppgift är att klart meddela användaren att sidan tar slut, liksom sidhuvudets - eller headerns - uppgift är att meddela att sidan börjar. Användaren må redan veta detta men det ger en god struktur, något som förväntas av Web 2.0-design. Sidfoten består här av en ljus siddelare samt rum för kontaktinformation. Sidfoten justeras vid överlämning av produkten i enlighet med uppdragsgivarens önskemål.

6 Intern design

De administrativa sidorna uppgör grunden för innehållshanteringssystemet. Administratören tillbringar tid på dessa sidor vid uppdatering av innehållet. Sidorna har samma utformning som kundsidorna, den enda egentliga skillnaden är en extra länk i menybalken. Kund- och administratörssidorna är uppbyggda med samma stomme – samma div klasser. Detta innebär att de kan använda samma CSS-fil utan problem. Administratörssidorna är uppbyggda med PHP. PHP körs på servern medan HTML tolkas direkt av webbläsaren. För att kunna köra PHP-koden måste servern ha en skripttolk. Användningen av PHP är inte nödvändig – funktionaliteten hade kunnat åstadkommas med HTML, men PHP möjliggör enklare kod med mindre onödig upprepning.



Figur 7. Editeringsformulär på administratörssidorna.

6.1 Säkerhet

Eftersom webbplatsen inte innehåller någon känslig information såsom kundregister är säkerheten inte huvudsaken för detta projekt. En grundläggande säkerhet är dock viktig för alla webbsidor. Vid ett tidigare projekt i kursen Dynamisk Webbtillämpning vid Yrkeshögskolan Novia byggde den grupp jag tillhörde en fiktiv banks hemsida. Denna sida innehöll kod för in- och utloggning samt elementär säkerhet. Denna kod har editerats i enlighet med projektets behov och återanvänts.

Den administrativa delen består av en enda sida som visar innehållet av andra sidor i textrutan. URL-adressen till yritys-sidan på de administrativa sidorna syns i figur 8. Vid val av denna länk visas innehållet av yritys-sidan i textrutan på de administrativa sidorna.

Figur 8. Yritys-sidans URL-adress.

Har webbplatsen ingen säkerhet kan användare skriva in URL-adressen i tabell 2 och så förbigå inloggningsfältet på administratörssidan admin.php. För att undvika detta används koden i tabell 2 runt den kod som kallar på menybalken och sidinnehållet. Koden kontrollerar att användaren är inloggad. Om användaren redan är inloggad kommer han till den önskade sidan, annars omdirigeras han till inloggningssidan.

Tabell 2. PHP-kod för att kontrollera om användaren är inloggad.

```
62          if(isset($_SESSION['logged_in'])){
```

6.1.1 Logga in/ut

In- och utloggning sköts med PHP och en MySQL-databas. Användarnamn och krypterade lösenord sparas i databasen. Webbplatsen använder PHP-sessioner för att kontrollera användarens inloggningsstatus. Vid fel lösenord meddelas användaren om detta med en JavaScript alert.

6.1.2 Kalender

Kalendern skall från början vara en separat modul uppbyggd med PHP och Ajax. Vid undersökning och preliminär utveckling visar det sig dock att en sådan kalender i detta skede är mera komplicerad än behövligt. Tidsbrist för en sådan utveckling leder till en omjustering av kalenderfunktionen. Kalendern editeras nu med samma innehållshanteringssystem som allting annat. Kalendertillfällen sparas i textformat i en XML-fil och skiljer därmed inte från de andra sidorna. Den originella kalenderfunktionen är ett projekt för sig om den skall bli gjord korrekt.

7 Innehållshanteringssystem

CMS står för Content Management System eller innehållshanteringssystem. Marryt-webbplatsen är liten och kräver inte ett stort innehållshanteringssystem. Ett alternativ kunde vara att använda ett lätt system inriktat mot bloggar, t.ex. Wordpress. Från början står det dock klart att arbetet skall börja från noll och ett innehållshanteringssystem skall skräddarsys åt Marryt. Vid slutet av projektet integreras ändå en extern editor – CKEditor, till webbplatsen. Detta sker för att göra sidorna mera användarvänliga för kunden.

Beslutet att använda JavaScript och Ajax vid mån av möjlighet görs tidigt. Dessa leder till XML. Det finns inte några manualer som direkt adresserar vad som vill åstadkommas med projektet. Tom Myers artikel *Build an XML-Based Content Management System with PHP* (2003) berättar om hur XML kan användas i innehållshanteringssystem. Den är en bra grund, men använder PHP för hela processen medan projektet använder JavaSript och Ajax. Artikeln används som guide till XML och dess uppbyggnad men den inkluderade PHP-koden är inte av intresse.

7.1 Bakom sidinnehållet

Webbplatsen består av en kunddel och en administratörsdel. Kunddelen utgörs av fem sidor; yritys, kurssit, kalenteri, linkit och yhteystiedot. Sidinnehållet läses in på sidorna från externa XML-filer med JavaScript. Sidinnehållet består av text. Administratörsdelen läser in den editerbara texten från samma XML-filer och innefattar även en instruktionssida; admin sivuista. Grunden till kunddelen är uppbyggd med HTML och CSS. Administratörsdelen är PHP-baserad men innehåller samma HTML och CSS

användargränssnitt som kunddelen. Inloggningen sköts med PHP och en underliggande MySQL-databas med krypterade lösenord. Innehållshanteringssystemet är uppbyggt med PHP, XML, JavaScript och Ajax. CKEditor är integrerad.

7.1.1 Varför XML?

Eftersom innehållshanteringssystemet skräddarsys för att hålla det så lätt som möjligt används XML (Extensible Markup Language). I *XML steg för steg 2:a upplagan* (Young, 2002, xi) skriver Young att XML finns till för att: ”lagra och utbyta information på webben”. Vidare skriver Young ”med XML, i vilket man kan definiera egna element, attribut och dokumentets struktur, kan man beskriva praktiskt taget vilken information som helst”. Medan ett PHP- och MySQL-baserat innehållshanteringssystem är ett alternativ har här valts att använda XML i samband med JavaScript och Ajax. Projektet förblir så sina Web 2.0-rötter troget. Med XML undgås också problem med SQL-injektion.

7.1.2 XML-dokument

XML är ett format för att spara information. Ett enkelt sätt att beskriva XML är att det motsvarar HTML men man får själv bestämma taggarna eller elementen och strukturen. I HTML finns element såsom p, body och li. Dessa är förutbestämda och webbläsarna vet hur de skall hantera dem. I XML kunde motsvarande element vara text, content och link. Det finns inga förutbestämda taggar och därmed kan de fritt formuleras. För att läsa informationen i en XML-fil måste en parser användas. I HTML är det klart att paragraf- och länkelementen ligger i body-taggen. I XML finns det inga sådana restriktioner. XML kan lätt användas för olika databaser, t.ex. bokdatabasen i tabell 3. Rotelementet books berättar vad XML-dokumentet innehåller – böcker. Varje book-element har sedan underliggande title-, published-, author- och summary-element. På engelska definieras dessa som books som root, book som child och de resterande elementen som subchild element. Det byggs alltså upp en viss hierarki.

Tabell 3. Exempel på XML-fil.

```
<?xml version="1.0"?>
<books>
  <book>
    <title>Book One</title>
    <published>2010</published>
    <author>Author One</author>
    <summary>Description one.</summary>
  </book>
  <book>
    <title>Book Two</title>
    <published>2011</published>
    <author>Author Two</author>
    <summary>Description two.</summary>
  </book>
</books>
```

Sidinnehållet sparas i XML-dokument. Webbplatsen använder grundlägande XML-dokument med huvudelementet content och underliggande elementen maintitle och body. Då användaren vill ändra titeln Book Two till Book Three i exemplet i tabell 3 går han in på editeringssidan där XML-filens innehåll visas i ett editeringsfält. Vid editering skapar koden en ny XML-fil med de förutbestämda elementen och texten i editeringsfältet. Den nya XML-filen har samma namn som den gamla filen och sparas på samma ställe vilket leder till att den gamla filen skrivs över.

XML-filens innehåll - elementens värde, kan inte innehålla vissa märken. För att undvika detta kan man använda CDATA. Genom att placera ”<![CDATA[” före och ”]]>” efter innehållet mellan elementtaggarna försäkrar man att texten inte läses som XML. Då XML filen öppnas och visas på webbplatsen läser parsern XML-elementen men ignorerar innehållet. Webbplatsen visar innehållet och eftersom det är formaterat med HTML visas det upp i enlighet med koden. Det visar sig dock vara onödigt att använda CDATA eftersom W3C i sin artikel *Extensible Markup Language (XML) 1.0 (Fifth Edition)* (2008) påpekar att alla förbjudna tecken, t.ex. ”&” kan ersättas, i detta fall med ”&”. Denna markering missuppfattas inte som en del av XML-dokumentets struktur utan ignoreras av parsern och läses först av webbläsaren som HTML.

7.1.3 DOM

DOM står för Document Object Model. W3C definierar DOM som; ”a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents”. DOM används alltså för att dynamiskt uppdatera t.ex. XML-dokument. I DOM representeras ett XML-dokument som ett nodträd. I exemplet med bokdatabasen i figur 11 har books-noden ett antal underliggande barnnoder (childnode). Varje book-barnnod har sedan underliggande title-, published-, author- och summary-noder. Dessa barnnoder kopplas till underliggande textnoder som innehåller bokens titel, publiceringsår, författarens namn och en sammanfattning. DOM använder noder för att beskriva uppställningen medan XML använder element.

De XML-dokument som används av webbplatsen är uppbyggda enligt DOM-modellen. JavaScript- och Ajax-koden kan tack vare detta läsa XML-dokumenten och PHP kan användas för att skriva nya XML-dokument. DOM-modellen gör det möjligt för koden att identifiera XML-elementen och –värdena och därmed var de skall placeras.

7.1.4 JavaScript

JavaScript är ett klientsidans skriptspråk som används för att göra webbsidor mera dynamiska. I motsats till PHP som körs på servern, körs JavaScript i användarens webbläsare. I tabell 4 visas ett exempel på JavaScript-kod. Koden initieras med en script-tag som identifierar språket och typen som JavaScript. Alla versioner av webbläsare förstår inte JavaScript och för att hindra dem från att tolka koden som HTML kommenteras allt mellan script-taggarna bort. Detta har ingen effekt då webbläsaren förstår JavaScript men ifall den inte gör det hindrar detta webbläsaren från att skriva ut koden på rutan. I detta exempel består koden av en alert – en poppuppruta med texten ”This is JavaScript”, som ber användaren bekräfta detta. Vanligen kopplas JavaScript-funktioner till något användaren gör; ett knapptryck eller att föra musen över ett objekt.

Tabell 4. JavaScript-exempel.

```
3  <script language="javascript" type="text/javascript">
4  <!--
5  alert("This is JavaScript");
6  //-->
7  </script>
```

JavaScript kan placeras och köras där det kallas i ett HTML-dokument men detta rekommenderas inte. Koden blir tungkörd då webbläsaren måste gå igenom all JavaScript vid laddningen. Det är bättre att placera JavaScript-funktionerna i headern och kalla på dem i själva HTML-koden. Detta kan dock leda till massiva mängder kod i början av dokumentet och är därmed inte idealt. Bäst är att placera JavaScript-koden i en extern fil. I *JavaScript – Tehokas hallinta* (Smith & Negrino, 2007, 26) beskrivs hur en extern JavaScript-fil kan användas. Externa JavaScript-filer rekommenderas eftersom de möjliggör återvinning av koden. Om JavaScript-koden lämnas som intern kod i ett dokument kan den inte kallas från ett annat dokument och måste skrivas om.

I projektet använder kunddelen intern JavaScript-kod medan administratörsdelen har JavaScript-koden samlad i en extern fil. Dokumenten kopplade till CKEditor följer inte denna princip eftersom det är en extern modul.

7.1.5 Ajax

Ajax står för Asynchronous JavaScript and XML. I *JavaScript – Tehokas hallinta* (Smith & Negrino, 2007, 9) definieras Ajax som en kombination av XHTML, CSS, DOM, XML och XMLHttpRequest. I praktiken går Ajax ut på att endast skicka den information som behöver skickas. Detta för att få en snabb och smidig applikation. En traditionell webbsida laddar om hela sidan, men en sida som använder Ajax laddar endast om den del av sidan som ändras. Ajax nytta syns klart vid ifyllnad av formulär; varje fält kan valideras separat för sig medan användaren fyller i formuläret. Informationen användaren skriver jämförs med de givna parametrarna och om de inte motsvarar dessa meddelas detta genast på sidan. Ajax har gjort det möjligt att kontrollera att svaren är korrekt ifyllda utan att skicka formuläret. När formuläret sedan skickas vet användaren att alla fält är godkända; han kommer inte att tvingas fylla i hela formuläret på nytt eftersom något fält blivit fel ifyllt. Google Maps är ett av de mest kända exemplen på användning av Ajax. Jämför Google Maps mot en annan kartservice; hela sidan måste laddas om då man vill granska en ny bit karta. Ajax gör upplevelsen snabbare och smidigare och innebär att användaren inte tvingas vänta medan delar av sidan som inte ändrats laddas om.

XMLHttpRequest är grunden för alla Ajax-funktioner. I *Ajax – Tehokas hallinta* (Asleson & Schutta, 2007, 25-26) konstateras det att det inte finns något standardiserat sätt att bygga upp XMLHttpRequest-objektet. Koden i tabell 5 visar hur objektet kan byggas i olika

webbläsare. Om koden returnerar ett värde för den första if-satsen använder användaren Internet Explorer version 7 eller senare, Mozilla Firefox, Google Chrome, Opera eller Safari. På rad 31 ges variabeln xmlhttp då värdet ”new XMLHttpRequest”. Ifall användaren använder Internet Explorer version 5 eller 6 används den andra if-satsen. Äldre versioner av Internet Explorer har inte XMLHttpRequest integrerat eftersom det inte var standardteknik då dessa kom ut. På rad 35 ses dock hur xmlhttp-variabeln ändå kan definieras med hjälp av ActiveX. Variabeln xmlhttp får samma värde i alla fall, men de olika webbläsarna kräver skild kod för att åstadkomma det.

Tabell 5. XMLHttpRequest-objektet definieras.

```
29  if (window.XMLHttpRequest)
30    {// code for IE7+, Firefox, Chrome, Opera, Safari
31    xmlhttp=new XMLHttpRequest();
32    }
33  else if (window.ActiveXObject)
34    {// code for IE6, IE5
35    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
36    }
```

7.2 Visa sidinnehåll

Administratörs- och kunddelen använder olika metoder för att visa sidinnehållet. Kunddelen använder HTML och visar sidinnehållet i en div i textformat. Administratörsdelen använder PHP och sidinnehållet visas i textområden ytterligare omvandlade med en extern editor. Denna skillnad finns, eftersom administratörs- och kunddelen används för skilda ändamål.

7.2.1 Kunddel

Koden i tabell 6 används för att visa sidinnehållet på kunddelen av webbplatsen. På rad 38 definieras den div-tag i vilken innehållet visas. På rad 40-47 sköts behandlingen av XMLHttpRequest. Koden på rad 48 öppnar kopplingen till servern. Det bestäms att ”GET”-metoden skall användas för att öppna ”yritys_text.xml”-filen. Det sker synkront vilket indikeras av ”false”-texten. Detta betyder att denna händelse körs då sidan laddas – det är alltså JavaScript. Hade det stått ”true” skulle koden köras asynkront – vid mustryck eller motsvarande utan att sidan laddas om, och det skulle handla om Ajax. På rad 49 skickas begäran till servern. På nästa rad definieras variabeln xmlDoc. Det bestäms att den

skall innehålla serverns svar på förfrågan i XML-format. Nu går svaret att behandla i enlighet med DOM-modellen.

Tabell 6. Index.html. JavaScript-kod för att visa sidinnehållet på kundsidan.

```
38  <div id="text">
39  <script type="text/javascript">
40  if (window.XMLHttpRequest)
41    {// code for IE7+, Firefox, Chrome, Opera, Safari
42    xmlhttp=new XMLHttpRequest();
43    }
44  else
45    {// code for IE6, IE5
46    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
47    }
48  xmlhttp.open("GET","yritys_text.xml",false);
49  xmlhttp.send();
50  xmlDoc=xmlhttp.responseXML;
51
52  var x=xmlDoc.getElementsByTagName("content");
53
54    document.write(x[0].getElementsByTagName("maintitle")[0].childNodes[0].nodeValue);
55    document.write(x[0].getElementsByTagName("body")[0].childNodes[0].nodeValue);
56
57  </script>
58  </div>
```

Variabel ”x” på rad 52 innehåller alla element med namnet content i form av en nodlista. Det finns i detta fall endast ett content-element. ”document.write” används för att skriva ut innehållet mellan parenteserna på rad 54 och 55. Koden ”x[0]” refererar till variabeln x och content-elementet. Noderna i XML-dokumentet gås alltså igenom. Koden skriver ut värdet av textnoden under maintitle-elementnoden. På följande rad skrivs värdet av textnoden under body-elementnoden ut. Resultatet av denna kod syns på indexsidan.

7.2.2 Administratörsdel

Admin.php uppgör stommen för administratörssidan samt behandlar koden för inloggning. I tabell 7 kontrollerar koden att användaren är inloggad. Är han det, visas menybalken med länkarna till de olika editeringssidorna. Länkarnas URL-adress innehåller ett get-värde. Med PHP if- och elseif-satser kontrolleras vid knapptryck vilken länk som valts genom att kontrollera om länkens get-värde motsvarar if-satsens get-värde. På rad 47 i tabell 7 är get-värdet ”admin”. På rad 65 kontrolleras get-värdet. Då användaren valt länken med get-värdet ”admin” används ”echo” för att visa texten på rad 66-68.

Tabell 7. Del av PHP-kod som vid val av länk visar önskad sida.

```
44              <?php
45              if(isset($_SESSION['logged_in'])){
46              echo    "<ul id=\"nav\">
47              <li><a id=\"admin\" href=\"admin.php?get=admin\">Admin sivuista</a></li>
48              <li><a id=\"yritys\" href=\"admin.php?get=yritys\">Yritys</a></li>
49              <li><a id=\"kurssit\" href=\"admin.php?get=kurssit\">Kurssit</a></li>
50              <li><a id=\"kalenteri\" href=\"admin.php?get=kalenteri\">Kalenteri</a></li>
51              <li><a id=\"linkit\" href=\"admin.php?get=linkit\">Linkit</a></li>
52              <li><a id=\"yhteystiedot\" href=\"admin.php?get=yhteystiedot\">Yhteystiedot</a></li>
53              </ul>";
54              }
55              ?>
62          if(isset($_SESSION['logged_in'])){ //Display page contents if logged in
63              if(isset($_GET['get'])){
64                  //admin page content
65                  if($_GET['get']=="admin"){
66                  echo "<h2>Admin sivu</h2>";
67                  echo "<p>Tähän tietoa admin sivuista.</p>";
68                  echo "<p>Sivujen muokkaukseen käytetään CKEditoria.</p>";
69                  }
```

I tabell 8 visas den kod som körs då get-värdet är ”yritys”. Rad 72 och 73 använder ”echo” för att visa text som tidigare. På rad 74 definieras en div med id-text. I denna div skapas ett formulär. I detta formulär används JavaScript för att på rad 77 definiera variabeln add. I detta fall får variabeln värdet ”yritys1”. Denna variabel deklareras för att senare kunna skilja på de olika sidorna. På rad 78 kallas JavaScript-funktionen ”showmy()”. På rad 80 definieras en knapp som vid knapptryck kallar på JavaScript-funktionen ”yritystxt()”. Tabell 8 slutar med de stängande taggarna för formuläret och diven.

Tabell 8. Admin.php. Visa yritys sidan.

```
71                  elseif($_GET['get']=="yritys"){
72                  echo "<h2>Yritys</h2>";
73                  echo "<p>Muokkaa yritys sivuja.</p>";
74                  echo "<div id=\"text\">";
75                  echo "<form action=\"#\">";
76                  echo "<script type=\"text/javascript\">";
77                  echo "var add = 'yritys1';";
78                  echo "showmy();";
79                  echo "</script>";
80                  echo "<input type=\"button\" value=\"Muuta\" onclick=\"yritystxt()\"/>";
81                  echo "</form>";
82                  echo "</div>";
83                  }
```

Funktionen ”showmy()” i tabell 10 börjar på rad 37 med att kalla på funktionen ”choose()”. Denna funktion används för att bestämma variabeln url och därmed URL-adressen på det XML-dokument som skall öppnas. I tabell 9 syns funktionen ”choose()”. På rad 57 och 60 kontrollerar koden om variabeln add motsvarar värdet givet i tabell 8, rad 77. Då den gör det ges variabeln url sitt värde.

Tabell 9. Del av JavaScript-funktionen choose().

```
56  function choose(){
57  if (add == 'yritys1'){
58      url = "../yritys_text.xml";
59      }
60  else if (add == 'kurssit2'){
61      url = "../kurssit_text.xml";
62      }
```

Sidinnehållet på de administrativa sidorna visas långt på samma sätt som på kundsidorna. Skillnaden är att sidinnehållet här placeras i textområden. Detta demonstreras på rad 44 och 48 i tabell 10. Textområdena ersätts på rad 51 och 52 med CKEditorns rich text editor.

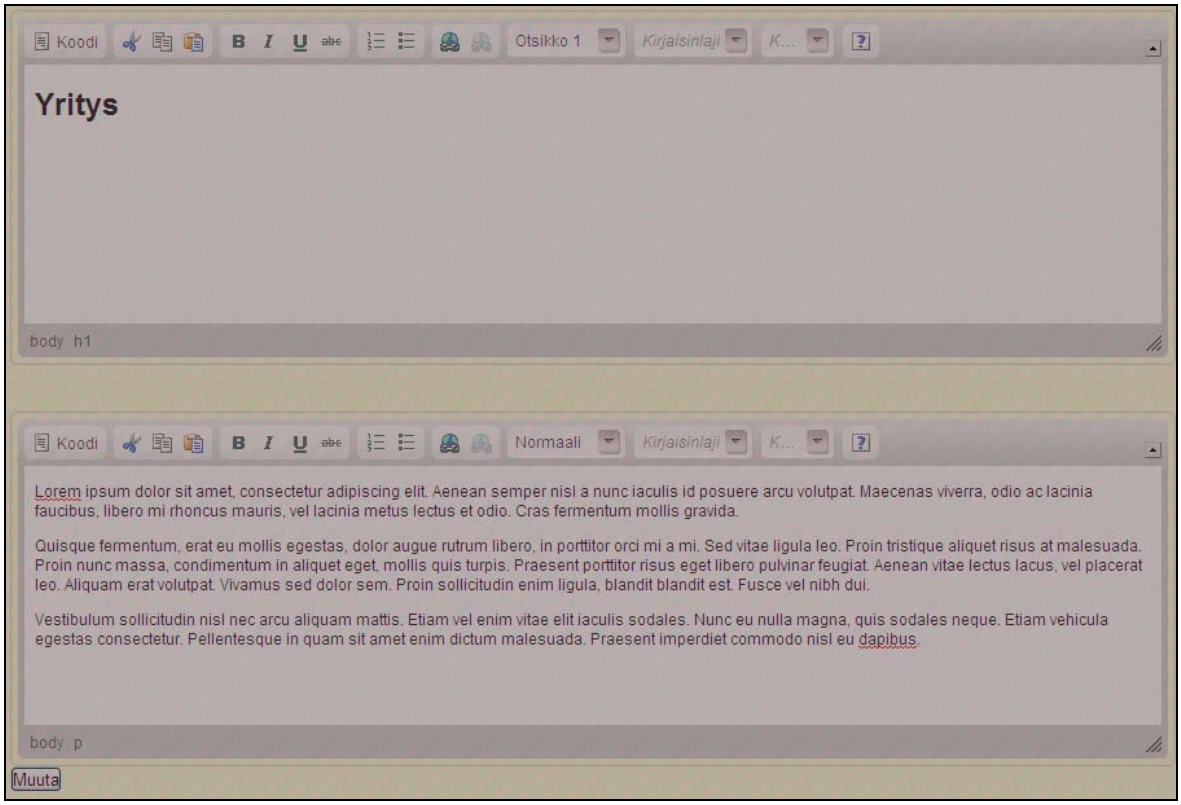
Tabell 10. JavaScript-funktionen showmy().

```
27  function showmy(){
28  if (window.XMLHttpRequest)
29    {// code for IE7+, Firefox, Chrome, Opera, Safari
30    xmlhttp=new XMLHttpRequest();
31    }
32  else if (window.ActiveXObject)
33    {// code for IE6, IE5
34    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
35    }
36
37  choose();
38  xmlhttp.open("GET",url,false);
39  xmlhttp.send();
40  xmlDoc=xmlhttp.responseXML;
41
42  var x=xmlDoc.getElementsByTagName("content");
43
44  document.write("<textarea id='maintitle' rows='1' cols='110'>");
45  document.write(x[0].getElementsByTagName("maintitle")[0].childNodes[0].nodeValue);
46  document.write("</textarea>");
47  document.write ("<br></br>");
48  document.write("<textarea id='txt' rows='15' cols='110'>");
49  document.write(x[0].getElementsByTagName("body")[0].childNodes[0].nodeValue);
50  document.write("</textarea>");
51  CKEDITOR.replace( 'maintitle' );
52  CKEDITOR.replace( 'txt' );
53  }
```

7.3 Editera sidinnehåll

I detta skede är sidinnehållet synligt och kan editeras. Uppdragsgivaren är från tidigare bekant med HTML och har inget direkt behov av en s.k. rich text editor. En lösning där uppdragsgivaren sköter editeringen i en HTML-editor är dock inte en speciellt elegant lösning och den motsvarar inte heller Web 2.0-principerna. På grund av detta används CKEditor. CKEditor är en open source WYSIWYG (What You See Is What You Get) JavaScript baserad editor. Detta betyder att innehållet i textområdena motsvarar den

slutliga texten till utseendet. Man ser alltså genast vid editeringen hur slutresultatet kommer att se ut. I figur 9 ses editeringsfälten för yritys-sidan. Orsaken till de två fälten är kompabilitet med XML-filerna.



Figur 9. CKEditorns editeringsfält för yritys-sidan.

7.4 Skicka sidinnehåll

Brett McLaughlin berättar i artikeln *Mastering Ajax, Part 7: Using XML in request and responses* (2006) hur flera XML-värden kan sammanslås i en string och sparas som en variabel. Variabeln skickas sedan till den PHP-fil som sköter om att spara XML-data. I den filen körs processen omvänt och XML-värdena plockas fram. Detta innebär dock att omvandla varje värde till en variabel innan dessa variabler kan slås samman till en enda variabel som sedan skickas. Denna process är onödigt komplicerad för de två elementen (maintitle och txt) och de bakomliggande värdena samt URL-adressen som skall skickas.

Då editeringen är klar trycker användaren på ”Muuta” vilket kallar på JavaScript-funktionen ”yritystxt()” i tabell 11. På rad 14 definieras variabeln maintitle som värdet av CKEditor-instanser av maintitle. Raden under körs samma kod med variablen och instanser av txt.

Tabell 11. Del av JavaScript-funktionen yritystxt().

```
14  var maintitle = CKEDITOR.instances.maintitle.getData();
15  var txt = CKEDITOR.instances.txt.getData();
16  choose();
17  xmlhttp.open("POST","edit.php",true);
18  xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
19  xmlhttp.onreadystatechange=function() {
20    if (xmlhttp.readyState==4) {
21          alert("Sivu on päivitetty.")
22      }
23   }
24  xmlhttp.send("maintitle=" + maintitle + "&txt=" + txt + "&url=" + url);
25  }
```

En programmerare önskar alltid hålla koden ren och enkel. Därför skrivs funktioner med globala variabler. När man kallar på funktionerna skickas variablernas värden med. Detta fungerar mycket bättre än att upprepa samma text om och om igen med olika värden. Därför har funktionen ”choose()” kallats här på rad 16 i tabell 11. Genom att definiera url-variabeln i en separat funktion undviks behovet att skriva denna funktion om för varje sida.

På rad 17 öppnas kopplingen till servern. Informationen skickas med POST till edit.php i Ajax-format. Detta innebär att sidan inte laddar om då innehållet skickas. Då informationen skickats meddelas användaren om detta genom en poppuppruta.

Istället för att skicka titeln, brödtexten och URL-adressen i en array skickas alla data med Ajax i en lång textstring på rad 24. I artikeln *Ajax Tutorial (Asynchronous Javascript + XML) Creating client-side dynamic Web pages* definieras en string med flera taggar och dem tillhörande värden genom att lägga in ”&” innanför citattecknet, före alla taggar förutom den första. Detta löser problemet med att inte kunna skicka flera värden samtidigt och ett antal taggar kan nu anges värden och skickas med Ajax.

Vid editering av innehållet är titeln obligatorisk. Om man lämnar titelfältet tomt visar sidan inget innehåll, eftersom koden stannar där och inte kommer vidare till själva texten.

7.5 Spara sidinnehåll

Sidinnehållet sparas med PHP-kod i filen edit.php. Koden börjar med att i tabell 12 definiera variablerna $txt, $maintitle och $url. Sidinnehållet skickas till edit.php med POST och kan därmed fångas upp med $_POST. POST och GET används båda för att föra information till servern. I detta skede används POST eftersom den i motsats till GET inte har någon begränsning på mängden information som kan förmedlas.

Tabell 12. Edit.php-variabler.

```
12  $txt = $_POST["txt"];
13  $maintitle = $_POST["maintitle"];
14  $url = $_POST["url"];
```

Koden i tabell 12 och 13 används för att bygga upp ett nytt XML-dokument med huvudelementet content och de två underliggande elementen maintitle och body. Koden här är baserad på kod från artikeln *Reading and writing the XML DOM with PHP* (2005).

Koden i tabell 13 på rad 22 och 23 kallar på ett nytt formaterat DOM-dokument. Variabeln $r definieras som ett nytt DOM-dokument i vilket elementet content skapas. Koden ”appendChild” betyder att en ny nod läggs till efter den sista barnnoden (childnode) för den givna elementnoden. Barnnoden maintitle skapas. En textnod skapas till maintitle-noden. Textnodens innehåll läses in från $content arrayn på rad 31. Textnoden kopplas till maintitle-noden. Samma process gås igenom då body-noden och tillhörande textnod skapas. Slutligen kopplas noderna maintitle och txt till content-noden på rad 36 och 37.

Det nya XML-dokumentet sparas till $url. Denna variabel definieras på editorsidan och skickas med titeln och brödtexten. På rad 14 i tabell 12 ges $url sitt värde. Det nya XML-dokumentet sparas med samma namn, på samma plats som det gamla dokumentet och därmed skrivs den gamla filen över.

Tabell 13. Edit.php. Skapa ny XML-fil.

```
17  $contents [] = array(
18  'maintitle' => $maintitle,
19  'body' => $txt
20  );
21
22  $doc = new DOMDocument();
23  $doc->formatOutput = true;
24
25  $r = $doc->createElement( "content" );
26  $doc->appendChild( $r );
27
28  foreach( $contents as $content )
29  {
30  $b = $doc->createElement( "maintitle" );
31  $b->appendChild($doc->createTextNode( $content['maintitle'] ) );
32
33  $t = $doc->createElement( "body" );
34  $t->appendChild($doc->createTextNode( $content['body'] ) );
35
36  $r->appendChild( $b );
37  $r->appendChild( $t );
38  }
39
40  echo $doc->saveXML();
41  $doc->save("$url")
```

Administratörssidan där editeringen görs förändras inte eftersom justeringarna skickas med Ajax. Sidan laddas alltså inte om utan den editerade informationen skickas asynkront - i bakgrunden. Det här är det underliggande målet med projektet.

8 Resultat

Webbplatsen för Marryt motsvarar inte de originella planerna. Detta kan åskådliggöras med figur 1 och 2. Webbplatsen fyller dock fortfarande de krav som uttryckts i kravspecifikationen.

Den slutliga webbplatsen och innehållshanteringssystemet är grundläggande Web 2.0 produkter med rum för utveckling. Web 2.0-standard har följts både för extern och intern design och essentiell Web 2.0-funktionalitet har implementerats i mån av möjlighet. Detta syns främst i användningen av Ajax, JavaScript och XML. Webbplatsen utgör en god grund att bygga vidare på i framtiden.

9 Reflektion och kritisk granskning

I denna del reflekterar jag över projektet efter dess slutförande. Detta stycke har delats in i fyra delar; i tidsplanen, tekniken, arbetet och andra kommentarer. Jag reflekterar över vad som hände och vad jag lärt mig av det, samt gör en kritisk granskning av projektet.

9.1 Tidsplanen

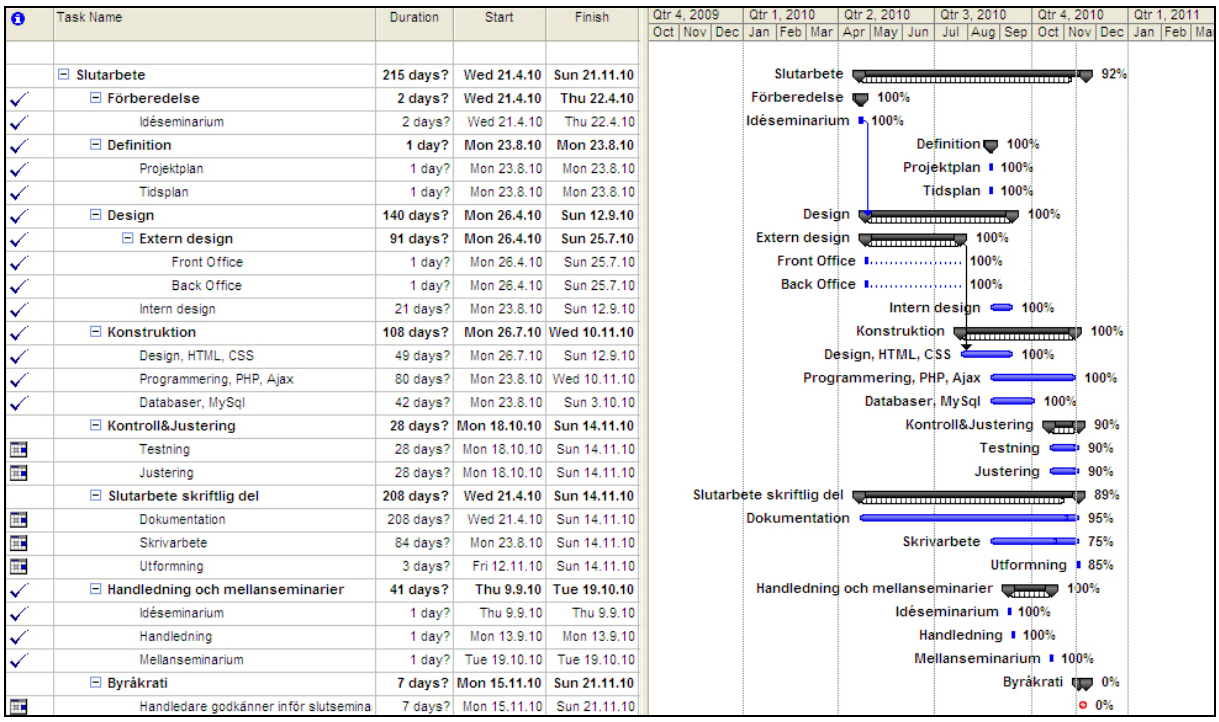
Tidsplanen färdigställdes den 23.8.2010 med hjälp av Microsoft Project. Den är indelad i åtta huvudrubriker samt ett antal underliggande uppgifter. Rubrikerna är följande;

- Förberedelse
- Definition
- Design
- Konstruktion
- Kontroll & Justering
- Slutarbete skriftlig del
- Handledning och mellanseminarier
- Byråkrati

Tidsplanen börjar den 21.4.2010 och slutar med den 12.12.2010. I september, under konstruktionsfasen stötte projektet på oväntade problem. På grund av privata skäl fördröjs arbetet, vilket placerar projektet närmare en månad efter tidsplanen då arbetet återupptas. Problem vid själva konstruktionen placerar projektet ytterligare ett par dagar bakom tidtabellen. Detta är förväntat, men på grund av de tidigare problemen finns det inte längre någon extra tid tillgänglig. I detta skede faller projektet en hel månad efter sin tidsplan.

Tidsplanen i figur 10 är en utveckling av tidigare tidsplaner. Den nya tidsplanen reflekterar situationen den 12.11.2010. Examensarbetet är färdigt till 92 % med kontroll & justering under arbete, liksom skrivprocessen.

Den ursprungliga tidsplanen inkluderar datum för slutseminarium, bedömning etc. för att försäkra att examensarbetet blir klart i tid för att anhålla om examensbetyg i december. Dessa datum är inte relevanta för den nya tidsplanen och har lämnats bort.



	Task Name	Duration	Start	Finish
	Slutarbete	215 days?	Wed 21.4.10	Sun 21.11.10
✓	Förberedelse	2 days?	Wed 21.4.10	Thu 22.4.10
✓	Idéseminarium	2 days?	Wed 21.4.10	Thu 22.4.10
✓	Definition	1 day?	Mon 23.8.10	Mon 23.8.10
✓	Projektplan	1 day?	Mon 23.8.10	Mon 23.8.10
✓	Tidsplan	1 day?	Mon 23.8.10	Mon 23.8.10
✓	Design	140 days?	Mon 26.4.10	Sun 12.9.10
✓	Extern design	91 days?	Mon 26.4.10	Sun 25.7.10
✓	Front Office	1 day?	Mon 26.4.10	Sun 25.7.10
✓	Back Office	1 day?	Mon 26.4.10	Sun 25.7.10
✓	Intern design	21 days?	Mon 23.8.10	Sun 12.9.10
✓	Konstruktion	108 days?	Mon 26.7.10	Wed 10.11.10
✓	Design, HTML, CSS	49 days?	Mon 26.7.10	Sun 12.9.10
✓	Programmering, PHP, Ajax	80 days?	Mon 23.8.10	Wed 10.11.10
✓	Databaser, MySql	42 days?	Mon 23.8.10	Sun 3.10.10
	Kontroll&Justering	28 days?	Mon 18.10.10	Sun 14.11.10
	Testning	28 days?	Mon 18.10.10	Sun 14.11.10
	Justering	28 days?	Mon 18.10.10	Sun 14.11.10
	Slutarbete skriftlig del	208 days?	Wed 21.4.10	Sun 14.11.10
	Dokumentation	208 days?	Wed 21.4.10	Sun 14.11.10
	Skrivarbete	84 days?	Mon 23.8.10	Sun 14.11.10
	Utformning	3 days?	Fri 12.11.10	Sun 14.11.10
✓	Handledning och mellanseminarier	41 days?	Thu 9.9.10	Tue 19.10.10
✓	Idéseminarium	1 day?	Thu 9.9.10	Thu 9.9.10
✓	Handledning	1 day?	Mon 13.9.10	Mon 13.9.10
✓	Mellanseminarium	1 day?	Tue 19.10.10	Tue 19.10.10
	Byråkrati	7 days?	Mon 15.11.10	Sun 21.11.10
	Handledare godkänner inför slutsemina	7 days?	Mon 15.11.10	Sun 21.11.10

Figur 10. Tidsplan 2 – vidare utvecklad tidsplan.

Projektet har tagit mycket längre tid än beräknat. Arbetet har dragit ut på tiden av varierande orsaker vilket har lett till ett projekt som spridit sig över tio månader med sporadiskt arbete under denna tid.

9.2 Tekniken

Under projektets gång har jag lärt mig otroligt mycket nytt. Jag var inte mer än ytligt bekant med JavaScript, Ajax och XML innan projektets början. Min kunskap om ämnena har fördubblats och nu kan jag anse mig ha en grundläggande uppfattning om alla tre.

Projektet har haft problem med användningen av externa JavaScript-filer. Koden har fungerat väl i den fil som kallar den, men det har inte gått att kalla på kod från en extern fil. På grund av detta problem har all JavaScript-kod förblivit i PHP-filerna under utvecklingen istället för att samlas i en extern JavaScript-fil. Problemet påverkar inte funktionaliteten - endast filstrukturen. Problemet löstes vid en standardtestning då det visade sig att en script-tag finns i JavaScript-filen. Medan JavaScript-kod i HTML-filer skall deklareras med script-taggar skall det inte finnas några sådana taggar i en JavaScript-fil. Lösningen av

detta problem tillät centraliseringen av JavaScript-koden i en extern fil vilket förenklade filhanteringen.

Vid genomgång av projektet i efterhand finns det sådant som kunde förbättras när det gäller koden. Sidinformationen ligger i separata XML-filer vilka kunde kombineras i en gemensam fil. Koden på kunddelen kunde också samlas i en HTML- eller PHP-fil för att minska mängden repeterad kod.

9.3 Arbetet

Det bör nämnas att både tidsplanen, designen och funktionaliteten måste omformuleras upprepade gånger i början av projektet - inte på grund av bristfällig planering, utan på grund av bristfällig information. Vid kommande projekt är det viktigt att söka upp mera information före planeringsfasen. I detta fall var det JavaScript, Ajax och XML som måste studeras djupare innan jag förstod ämnet tillräckligt bra för att kunna göra planer som verkligen kunde genomföras.

Vid kommande projekt är det viktigt att minnas att hålla fast vid planerna. Eftersom jag arbetat ensam har det varit enkelt att ändra på planerna även i ett sent skede. Detta är dock inte det bästa sättet att arbeta på eftersom man hela tiden måste justera någonting. Ur ett projekts synvinkel är det bättre att ta sig mera tid att planera ordentligt i början, än att senare tvingas justera planerna.

Under detta projekt har jag också lärt mig att inte ta på mig för mycket arbete. Eftersom vi studerande genomgående arbetat i projektgrupper inom informationsbehandlingsutbildningen, har vi lärt oss att arbeta med projekt så som detta. Jag var dock inte förberedd på hur stor skillnad det är att arbeta med ett projekt ensam. Det tar mycket längre tid att gå igenom och assimilera information när man måste veta allting själv och arbetet inte går att dela upp.

På grund av långa perioder av inaktivitet under projektets lopp har det vid varje nystart krävts en viss tid för att återbekanta sig med projektet. Med mera intensiv arbetstakt hade arbetet varit mera effektivt. Pauserna i arbetet har dock inte varit en helt negativ upplevelse. Tiden har gett chansen för tankar att mogna och gett nytt perspektiv.

9.4 Andra kommentarer

Slutligen vill jag konstatera att detta projekt har varit en unik upplevelse. Jag ser fram emot att vidareutveckla de kunskaper jag skaffat under projektet, samt själva koden bakom webbplatsen.

Källförteckning

Asleson, R & Schutta, N. T.. (2007). *Ajax – Tehokas hallinta.* Jyväskylä: Gummerus.

CKSource. (u.å.). http://cksource.com/ckeditor (hämtat:11.1.2011).

DeGraeve. (u.å.). *Color Palette Generator.* http://www.degraeve.com/color-palette/index.php?q=http://science.nayland.school.nz/HamishM/images/sunrise.jpg,5f2a213c302fb05525e99a4bf0cb87,a31b03563635ff5b00ffc872fff9e1 (hämtat: 10.11.2010).

Gilmore, W. J.. (2005). *PHP & MySQL - Tehokas hallinta.* Jyväskylä: Gummerus.

Hatava, A.. (2003). *Verkkografiikkaa.* Helsinki: Edita Prima Oy.

Hayes, D.. (2006). *Kom igång med HTML.* Borgå: WS Bookwell.

Herrington, J.. (2005). *Reading and writing the XML DOM with PHP.* http://www.ibm.com/developerworks/library/os-xmldomphp/ (hämtat:14.11.2010).

McLaughlin Brett. (2006). IBM. *Mastering Ajax, Part 7: Using XML in request and responses.* http://www.ibm.com/developerworks/web/library/wa-ajaxintro7.html (hämtat: 15.10.2010).

Myer Tom. (2003). sitepoint. *Build an XML-Based Content Management System with PHP*. http://articles.sitepoint.com/article/management-system-php (hämtat: 10.11.2010).

Smith, T & Negrino, T. (2007). *JavaScript – Tehokas hallinta.* Jyväskylä: Gummerus.

Stripe Generator 2.0 beta. (u.å). http://www.stripegenerator.com/ (hämtat: 11.11.2010).

Theseus. (u.å.). www.theseus.fi (hämtat: 11.11.2010).

W3C. (2005). *Document Object Model (DOM).* http://www.w3.org/DOM/ (hämtat: 12.11.2010).

W3C. (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition).* http://www.w3.org/TR/REC-xml/#syntax (hämtat: 12.11.2010).

W3C. (u.å.). *W3C CSS Validation Service.* http://jigsaw.w3.org/css-validator/ (hämtat: 10.1.2011).

W3C. (u.å.). *W3C Markup Validation Service*. http://validator.w3.org/ (hämtat: 10.1.2011).

Way, J.. (2009). *30 HTML Best Practices for Beginners.* http://net.tutsplus.com/tutorials/html-css-techniques/30-html-best-practices-for-beginners/ (hämtad: 11.11.2010).

Weakley, R.. (2006). *Kom igång med CSS*. Mölnlycke: Elanders Infologistics.

Web design from scratch. (u.å.). *Web 2.0 How-To Design Style Guide.* http://www.webdesignfromscratch.com/web-design/web-2.0-design-style-guide/ (hämtat: 10.11.2010).

Wordpress. (http://wordpress.org/) (hämtat: 14.11.2010).

XUL Ajax. (u.å.) *Ajax Tutorial (Asynchronous Javascript + XML) Creating client-side dynamic Web pages.* www.xul.fr/en-xml-ajax.html#ajax-post (hämtat: 11.11.2010).

Yahoo! Developer Network. (u.å.) *Yahoo! YSlow*. http://developer.yahoo.com/yslow/ (hämtat: 10.1.2011).

Young, M. J.. (2002). *XML steg för steg 2:a upplagan.* Göteborg: Elanders Graphic Systems AB.

Bilaga 1

Kravspecifikation

Projektets namn	Marryts webbplats
Beställare	Företag: Marryt Kontaktperson: Martti Rytkönen
Bakgrund - Beskriv den aktuella situationen, problem och behov	Marryt har ingen webbplats. Firman drivs från ett privathem och har därmed inget kontor som kunder kan besöka. Företaget skickar offerter åt potentiella kunder via brev eller e-post och uppmanar dem att kontakta firman inom samma medium. Klienter kan också ta kontakt via telefon. Det här innebär att potentiella kunder inte känner till firman, om den inte tagit kontakt med dem först och många potentiella kunder går förlorade.
Effektmål - Hur skall projektet /produkten ändra situationen	Marryt skall ha en webbnärvaro. De skall finnas en plats online där kunderna kan komma åt information om företaget.
Projektmål - Resultatet av projektet översiktligt	• Webbplats • Innehållshanteringssystem
Funktionella produktkrav - Det som produkten skall göra för användaren	• Enkel navigation • Tillgänglig med olika webbläsare
Gränssnitt och design - Speciella krav till Utseendet	• Mörk text på ljus bakgrund • Text på finska