Kontorama.fi

Skapandet av en webbplats

Christoffer Svartström, Jan Grönqvist

Examensarbete för Tradenomexamen

Utbildningsprogrammet för Informationsbehandling

Raseborg 2014

EXAMENSARBETE

Författare: Christoffer Svartström, Jan Grönqvist
Utbildningsprogram och ort: Informationsbehandling, Raseborg
Handledare: Klaus Hansen
Titel: Kontorama.fi – Skapandet av en webbplats

Datum: 17.04.2014 Sidantal: 54 Bilagor: 1

Abstrakt

Detta arbete beskriver arbetsprocessen för att skapa en webbplats åt företaget Kontorama Ab. Webbplatsen byggs upp från grunden med CMS-systemet WolfCMS, och ersätter företagets gamla webbplats sommaren 2014.

Syftet med den nya webbplatsen är att företaget skall få bättre synlighet på nätet, och därmed locka potentiella kunder.

Kraven på webbplatsen var att den skulle vara användarvänlig och att den skulle ladda snabbt. Andra krav var att det skulle gå att mata in några reklambilder på första sidan, gärna animerade. Sidan skulle vara fullständigt tvåspråkig, och en användarmanual skulle skapas.

Som ett resultat av detta arbete skapades webbplatsen utifrån de önskemål som kunden hade. Största delen av kundens krav och önskemål kunde uppfyllas, och resultatet blev lyckat. Arbetsprocessen beskrivs närmare i detta examensarbete.

Språk: Svenska
Nyckelord: Webbplats, WolfCMS, Kontorama, webbdesign

BACHELOR'S THESIS

Author: Christoffer Svartström, Jan Grönqvist
Degree programme: Business Information Technology, Raseborg
Supervisor: Klaus Hansen
Title: Creating a website for Kontorama.fi

Date 17 April 2014 Number of pages: 54 Appendices: 1

Abstract

This thesis describes the work process for improving the stationery company Kontorama Ab's website. The new website is created using WolfCMS as a CMS-system, and it replaces the old website during the summer 2014.

The goal with the new website is to add visibility for the company online, and thereby attract potential new customers.

The requirements for the website were that it should be user-friendly and have fast loading times. Other requirements were that it should be possible to add advertisements for products on the first page, preferably animated. The website must be multilingual, and a user manual should be made.

As a result, the website was created based on the requirements that the customer presented. Most of the requirements were met and the outcome was successful. The working process will be described more closely in this thesis.

Language: Swedish
Keywords: Website, WolfCMS, Kontorama, webdesign

Innehåll

Bilagor

1 Inledning

Examensarbetet går ut på att göra en ny webbplats åt företaget Kontorama, beläget i Ekenäs, Raseborg. Den nya webbplatsen skall ersätta den befintliga webbplatsen som i dagsläget är föråldrad både innehållsmässigt och utseendemässigt, och har ett svårarbetat administratorgränssnitt vilket gör att det är tidskrävande och inte så intressant för personalen att hålla sidan uppdaterad. Uppdraget att skapa en ny webbplats fick vi när vi frågade beställaren, Björn Lindqvist som är verkställande direktör på Kontorama, om det skulle finnas behov att förnya webbplatsen. Vi fick grönt ljus att skapa en ny, representativ webbplats åt företaget, och arbetsprocessen kommer att beskrivas i detta examensarbete.

1.1 Företagets bakgrund

Kontorama är grundat 1973 av Bernhard Lundström, och då var namnet Ekenäs Kontorsmaskiner. Företaget sålde kontorsmaterial, skrivmaskiner och hade kopieringstjänst som omfattade A4 kopior och större ritningskopior. Med tiden så började Ekenäs Kontorsmaskiner även sälja datorer och laserkopior.

På 2000-talet skedde ett ägarbyte då Björn Lindqvist köpte företaget av Bernhard Lundström. I och med ägarbytet skedde inte några större förändringar.

I dagens läge har Kontorama 3 anställda som sköter butiken. Företagets huvudsyssla är försäljning av kontorsmaterial, men Kontorama erbjuder också kopieringstjänst som omfattar allt från svartvita A4-kopior till färgplanscher upp till A0-storlek. Därtill har Kontorama tjänster såsom efterbehandling av A4-dokument och utskrift av ritnings- och byggnadskopior. Företaget säljer också datorer, program och kringutrustning och har serviceutrymmen där datorer, printrar och kringutrustning repareras och problem åtgärdas.

1.2 Kravspecifikation

För att ta reda på hurdan webbsida som Kontoramas personal önskade togs ärendet upp på ett möte, där personalen fritt fick säga sina önskemål om sidans funktionalitet, utseende och innehåll. Till önskemål som var litet diffusa så ställdes det djupgående frågor för att få fram vad personalen egentligen menade med sina önskemål och krav. På ungefär en halv timme kom vi fram till följande punkter som önskades:

- Enkel, lättnavigerad förstasida, skall ladda snabbt
- Erbjudanden/ett urval produkter på första sidan
- Lätt att lägga till erbjudanden/produkter
- Erbjudanden/produkterna skall synas en stund, sen kommer följande
- På kontaktsidan skall det finnas bilder på personalen, uppgifter om samarbetspartners
- Fullständigt tvåspråkig (svenska och finska)
- Bilder från butiken, panoramabild
- Färgerna skall vara blått och vitt
- Användarmanual skall skapas

Av dessa punkter så var en del rent innehållsmässiga saker, såsom bilderna och uppgifterna om samarbetspartners, men även dessa skall vi arbeta på eftersom det inte finns befintligt material att använda för detta. Andra punkter var mer konkreta gällande sidans funktionalitet och utseende, som vi skall sträva efter under uppbyggandet av sidan, såsom snabba laddningstider, och att användarvänligheten skall vara bra för alla parter, både kund och upprätthållare.

Användarmanualen för sidan kom som ett förslag av oss, och personalen tyckte det var en väldigt bra idé som den inte hade funderat på som en del av sidan, så även denna punkt lades med i kravspecifikationen.

1.3 Syftet med arbetet

I detta arbete kommer vi att skapa en webbplats åt ett företag utgående från de kunskaper vi lärt oss under vår utbildning, och bygga vidare på dem. Under arbetsprocessen räknar vi med att bekanta oss närmare med olika programmeringsspråk och scripting, och lära oss mera om dem. De huvudsakliga språken som används är HTML, CSS, PHP och JavaScript. Även olika webbutvecklingsverktyg utnyttjas, så dessa kommer vi även att bli bekanta med.

Vi använder oss av CMS-system för att inte behöva bygga upp allting från grunden, så har vi en färdig grund att bygga vidare på.

Syftet är att skapa en webbplats åt företaget Kontorama. Den nya webbplatsen skall ersätta den befintliga som varit i bruk sedan beställaren köpte företaget. Webbplatsen skall byggas upp på det sättet att den laddar snabbt, är logiskt uppbyggd vilket leder till att den är lättanvänd både ur Kontoramas personals synvinkel, och även ur Kontoramas kunders synvinkel. Utseendet skall vara visuellt tillfredsställande. Det skall finnas möjlighet att lägga till erbjudanden och produkter med beskrivning i ett slags bildspel.

Webbplatsen skall optimeras så bra som möjligt för sökmotorer för att få mera träffar och besökare, och kraven som finns uppräknade i kravspecifikationen skall uppfyllas.

1.4 Syftet med webbplatsen, kundens synvinkel

Syftet med den nya webbplatsen är att företaget stolt kan hänvisa kunder till sin webbplats, eftersom webbplatsen är som en digital fasad för företaget. Webbplatsen måste se professionell ut, och användaren skall enkelt kunna navigera och hitta information han eller hon är ute efter. Växling mellan olika områden på sidan skall gå snabbt, så vi strävar efter att söktiderna skall vara snabba.

Personer som söker efter kontorsmaterial på nätet skall lätt hitta fram till Kontorama, och sidan skall ge besökaren en bra bild av företaget, och erbjuda information om vad för tjänster och produkter företaget erbjuder. Även

erbjudanden och aktuella händelser skall vara lätta att få tag på. Detta kommer att förverkligas genom att optimera sidan så bra vi kan för att den skall få en bra ranking i sökmotorers lista över träffar.

Den gamla webbplatsen är på flera olika sätt inte lämplig för företaget. Första sidan och de övriga sidorna håller inte samma stil, eftersom navigeringsbalken förflyttas vilket kan vara förvirrande för användaren. Även själva layouten är gammalmodig och innehållet är i behov av uppdatering. Dessa ändringar kunde genomföras genom att använda befintlig sida som botten, men ett av kraven är att vem som helst i personalen enkelt skall kunna lägga till innehåll, som t.ex. aktuella händelser, produktbilder och text, men detta uppfyller inte den befintliga webbplatsens administratorgränssnitt som är gammalmodigt och stelt att använda. Dessutom var det ett önskemål att enkelt kunna rekommendera en artikel på webbplatsen, så att den syns på den persons Facebooksida som klickade rekommendera.

2 CMS

För byggandet av den nya webbplatsen använder vi oss av ett CMS-system. Ordet CMS kommer från engelskans Content Management System, vilket på svenska blir publiceringsverktyg, eller innehållhanteringssystem. Ett CMS-system hjälper användaren att enkelt upprätthålla sin webbplats, och sköta sådana uppgifter automatiskt som annars skulle kräva en massa manuellt arbete. (Wordpresskurser, u.å.)

Utbudet av CMS-system är väldigt stort, och vi skall studera några av de mest allmänna systemen och bestämma vilket som lämpar oss bäst. De CMS-system vi i detta kapitel tar en närmare titt på är Drupal, Joomla och Wordpress. Vi tar även en närmare titt på systemet WolfCMS i detta kapitel, vilket är systemet som vi använde oss av för att skapa den slutliga webbplatsen.

2.1 Drupal

Drupal är ett open-source CMS-system som upprätthålls och utvecklas av ett samfund som omfattar över 630 000 personer. Drupal går under GPL-licensiering (GNU General Public License) och det betyder att vem som helst får ladda ner programmet och distribuera det vidare.

Drupal används både av mindre webbplatser och storföretags webbplatser, och erbjuder ett stort utbud av moduler och themes, och det finns ett omfattande forum var man hittar svar på de vanligaste problemen och frågorna, och även Drupal.org erbjuder väldigt mycket information för både nybörjare och experter.

Det som krävs för att komma igång med Drupal är en webbserver med Apache eller Microsoft IIS, PHP 5.2 eller högre, MYSQL 5.0 eller högre, eller annan valfri databasserver. Dessa krav gäller för Drupal 7 vilket kommer att användas om vi besluter oss för att använda Drupal, även om ny version, Drupal 8 är på kommande, men det känns säkrare att använda något som använts en tid och inte är i betaskedet.

Drupal.org kom ut på nätet år 2001, och det hela startade med att Dries Buytaert skapade en webbplats som han och sina vänner kunde hålla kontakt via och dela nyheter i deras liv. Efter att sidan publicerades, som då hette drop.org, så började den vidareutvecklas och allt fler blev intresserade av den. Detta ledde till att sidan explosionsartat började utvecklas efter att Dries släppte programmet, som nu fått namnet Drupal, för allmän spridning, och det blev från en liten sida att dela nyheter på, till ett världskänt CMS-system. (Drupal, u.å.)

2.2 Joomla!

Joomla är också ett open-source CMS-system, precis som Drupal. Joomla är också välkänt och används flitigt globalt, och år 2007 var det 30 miljoner nedladdningar av Joomlas CMS-system. (docs.joomla, 2013)

Även som Drupal, så har Joomla ett stort samfund på nätet och den vägen får man hjälp till de vanligaste problemen och även tips av andra användare. Moduler och themes finns det också till Joomla i samma utsträckning som till Drupal.

Sidorna som är skapade med Joomla varierar från små, personliga sidor som sällan uppdateras till stora organisationers hemsidor, föreningars sidor och skolors hemsidor.

Kraven för att komma igång med Joomla är liknande som kraven för Drupal, men Joomla stöder också Nginx som webbserver. (Joomla)

2.3 WordPress

WordPress är i grunden utvecklat för att användas vid skapandet av bloggar, men har senare vidareutvecklats så långt att det går att skapa webbsidor som med vilket CMS-system som helst. Projektet startade 2003 och då var det ett tiotal personer inblandade, och i dagens läge är 17 % av alla webbsidor på nätet skapade med hjälp av WordPress. Som de andra CMS-systemen är även WordPress open-source, och har som Joomla och Drupal ett stort samfund bakom sig, med supportforum och dokumentation.

WordPress används för alla typers webbplatser i dagens läge, allt från små, privatpersoners webbplatser till stora företags webbplatser.

Även för WordPress gäller ganska långt samma krav att komma igång som för Joomla och Drupal, vilket betyder PHP 5.2.4 eller högre, MySQL 5.0 eller högre, och Apache eller Nginx webbserver. (WordPress, u.å,)

2.4 WolfCMS

WolfCMS är ett lite mindre CMS-system, som är baserat på ett äldre system som heter frogCMS. Systemet erbjuder ett enkelt användargränssnitt, bra verktyg för filhantering och hantering av användare och rättigheter är gjort så enkelt som möjligt. Valet blev WolfCMS eftersom det inte verkade så tungarbetat som de större systemen såsom Drupal som vi började med. Andra avgörande orsaker var att WolfCMS har stöd för snippets, vilket vi kommer att förklara närmare, samt att skapandet av ett tema var mera i vår smak, eftersom det är möjligt att skapa ett tema relativt lätt från grunden, och sedan implementera koden i WolfCMS.

2.4.1 Snippets

I WolfCMS kan så kallade snippets användas för att placera innehåll på sidan. Det är fråga om en PHP-kodsträng som hämtar den information från databasen som man specificerar i kodsträngen.
För ett exempel på hur en snippet kan se ut, se kodexempel 1 nedan.

Kodexempel 1: Exempel på snippet

```
<?php $this->includeSnippet('exempel-snippet'); ?>
```

Kodsträngen ovan placeras in i layouten, mellan de div-taggar som man vill att informationen skall synas i. För att få innehållet att synas, måste den också läggas till via administratörpanelen i WolfCMS. Då bör det anges en rubrik till kodsträngen så att WolfCMS vet vilken information den skall hämta. I exemplet ovan skall rubriken då heta exempel-snippet för att fungera. På samma sida finns det ett till textfält nedanför rubriken och det är där som informationen man vill att skall synas i snippeten matas in. Det kan vara fråga om vanlig text, som t.ex. kontaktinformation, html-skript eller PHP-kod. Bilder går också bra att använda i snippets.
Med hjälp av snippets kan man skapa en mer flexibel sida och ha en bättre kontroll över hur information placeras i layouten.

3 Förberedelse av arbetet

När vi studerat alternativen vi funderat på som CMS-system så föll valet på Drupal, till stor del eftersom det var bekant från undervisningen vi fått i Novia, och en annan fördel är att Drupal har en välfungerande community. Fördelen med en välfungerande community är att ifall det uppstår problem så finns lösningen med stor sannolikhet på Drupals forum, eftersom någon annan antagligen har haft problem med något liknande. En annan fördel, som Joomla och Wordpress också hade, var att det finns ett stort urval moduler tillgängliga, så man hittar lätt en modul som uppfyller kraven man ställer. Joomla var också ett starkt alternativ som vi kunde ha tänkt oss, eftersom vi också använt oss av det systemet i tidigare

utbildning, och vi hade positiva erfarenheter. Wordpress strök vi ganska snabbt eftersom vi inte har någon erfarenhet av det, och så ansåg vi båda att det är lite mera anpassat för bloggar. Denna antagelse var troligtvis obefogad, eftersom WordPress har utvecklas en hel del, och det var mest på grund av att programmet var obekant som vi inte valde det som underlag för skapandet av webbplatsen.

I början funderade vi hur vi skulle arbeta med webbplatsen, eftersom vi bor en bit ifrån varandra. Ett webbhotell skulle ha varit den bästa lösningen, men vi hade inget ledigt hemsideutrymme till förfogande vid den tidpunkten, så vi fick fundera på en annan lösning.

Nästa tanke var att köra en lokal webbplattform innehållande alla de komponenter som behövs för att kunna starta upp Drupal från en hemmadator, eftersom det var denna metod vi använt i undervisningen. Denna idé vidareutvecklade vi, och då föll valet på programmet Uwamp som installerades i en delad mapp i programmet Dropbox. Vi var till en början osäkra om detta skulle fungera som vi tänkt oss, men inga problem uppstod. Hurdana program dessa är tas upp i kategorin utvecklingsverktyg som vi använt oss av.

När dessa program var installerade så kunde vi börja med installation av CMS-plattformen vi skulle bygga upp sidan med, som i början var Drupal, men ändrades senare till WolfCMS.

Vi träffades ofta och arbetade för det mesta tillsammans med webbplatsen, både med den praktiska delen och med den skriftliga delen. De gånger vi arbetade på skilda håll så föll kundkontakten och skrivandet mera på Christoffer, och Jan skötte det praktiska med webbplatsen.

3.1 Verktyg som användes

För skapandet av webbplatsen användes en mängd olika verktyg för att underlätta arbetsprocessen, och här beskriver vi de olika programmen närmare samt berättar deras funktionsprincip.

3.1.1 UwAmp

Uwamp är en webbplattform som man kan köra från sin egen dator, och den emulerar en webbserver. UwAmp innehåller alla nödvändiga program, såsom Apache, MySQL och PHP. Detta program möjliggör att man kan köra igång ett CMS-system på sin hemmadator och via localhost få tillgång till webbsidan lokalt, och funktionen är densamma som på ett riktigt webhotell, förutom att alla filerna är lokalt på användarens dator.

3.1.2 Dropbox

Dropbox är en gratis tjänst som låter sina användare dela och synkronisera filer, foton och filmer enkelt. Det var två studerande som var trötta på att skicka filer till varandra över e-post som grundade Dropbox år 2007, och idag används det av över 100 miljoner användare runtom världen. (Dropbox, u.å.)

När Dropbox är installerat på datorn syns det som vilken mapp som helst, och i denna kan man skapa undermappar som man kan hålla för sig själv, eller dela med olika personer. Programmet är väldigt användarvänligt och lämpar sig väl för delning av filer som används under projektets gång som man vill ha tillgång till från olika datorer och dela med sig, och i början användes även Dropbox som utrymme för själva sidan som utvecklades.

3.1.3 Arkku

Arkku är en Internet-tjänsteleverantör som har funnits sedan 2003. Tjänsterna de erbjuder är bl.a. webbhotell, e-postutrymme och olika IRC-tjänster. De flesta tjänster de erbjuder är gratis, men om man har större krav och gratistjänsterna inte räcker till har de även tjänster som kostar.(Arkku, u.å.)

Webbsidan vi utvecklar ställer inte så stora krav på webbhotellets funktioner eller bandbredd, så gratisversionen av webbhotellet räckte till för oss.

För att få tillgång till webbsideutrymmet så skickar man in en skriftlig ansökan där man motiverar varför man behöver webbsideutrymmet. I denna ansökan skrev vi att vi skulle använda Arkku för att utveckla en webbsida till ett företag, och att detta är ett skolarbete som vi inte får betalt för, så ansökan godkändes.

3.1.4 Notepad++

Notepad ++ är ett litet men kraftfullt kodhanteringsprogram baserat på Scintilla, som stöder de flesta programmeringsspråken. Det är ett open-source program och är gratis att använda och ladda ner. (Notepad, u.å.)

Vi använder programmet för att editera koden i de olika filerna som webbplatsen är uppbyggd på. Fördelar med Notepad++ jämfört med traditionella notepad i Windows är igenkänning av programmeringsspråk, vilket hjälper till vid programmeringen och undviker skrivfel och felaktiga kommandon. En annan klar fördel är att man kan ha flera kodfiler öppna under olika flikar i programmet. Notepad++ har även en inbyggd FTP-klient, som användes vid editering av sidans CSS-filer, men är såpass begränsad att vi valde att ladda upp övrig information, såsom bilder och plugins, med ett annat program som heter Filezilla.

3.1.5Filezilla

Filezilla är såsom WinSCP ett FTP-program, också open-source. Programmet används för att flytta över filer mellan den lokala datorn som man har i användning, och en dator på nätet.

Även andra program som motsvarar Filezilla användes, som t.ex. Microsofts motsvarighet WinSCP och den inbyggda FTP-klienten i Notepad++.

3.1.6 Adobe Photoshop CS3, CS5

För att editera bilder och skapa olika grafiska element som används till webbplatsen använde vi oss av Photoshop som hör till Adobes produktfamilj. Vi använde oss av två olika versioner, CS3 och CS5, och orsaken till detta var helt enkelt att vi hade olika versioner av programmen på de datorer vi använde oss av under projektet.

Adobe Photoshop är ett bildredigeringsprogram som är mycket kraftfullt och erbjuder många möjligheter att manipulera och förbättra bilder, samt skapa helt ny grafik. Photoshop används av fotografer och grafiska designers världen runt.

3.1.7 Mozilla Firefox

För att testa sidans utseende och funktionalitet har vi främst använt oss av Firefox, men även andra webbläsare. Orsaken att vi tar upp Firefox skilt är att det har varit till stor hjälp speciellt eftersom det finns inbyggt ett inspektörverktyg, vilket möjliggör att direkt via webbläsaren se olika element, och även koden bakom dem. För att snabbt kontrollera värden man angett så har inspektören varit till hjälp, och vid tillfällen då olika element har ställt till med problem så har det varit lätt att lokalisera felet, ofta med hjälp av Firefox.

3.2 Byte av CMS-program

Efter att vi satt igång med Drupal och börjat modifiera ett starttema, så märkte vi ganska snabbt att projektet var på väg åt fel håll. Kunden önskade en lätt, snabb sida som var enkel, men dessa önskemål kände vi inte igen i Drupal. För det första så kändes administrationsgränssnittet en aning tungarbetat, och själva utvecklingen av systemet krävde att man verkligen var insatt i systemet för att förstå hur det egentligen fungerar. Detta märkte vi bra vid modifieringen av bastemat vi tänkte använda oss av. Det var en massa steg och inställningar som skulle göras för att komma igång, och olika inställningar som skulle göras var fördelade i flera olika kodfiler, vilket blev förvirrande. Den mest avgörande orsaken var ändå att Drupal är ett massivt, kraftigt CMS-system som vi inte tyckte passade in i det vi skulle göra.

Vi valde att söka ett CMS-system som inte var riktigt lika omfattande som Drupal, och som skulle passa vårt projekt bättre. Vi prövade testversioner av olika CMS-system och läste dokumentation, och slutligen så föll valet på WolfCMS.

Fördelarna med WolfCMS jämfört med Drupal är att det helt enkelt är ett mindre, enklare system som inte innehåller lika mycket funktioner. Detta passade oss bra, eftersom vi inte ändå behövde nära på all funktionalitet som Drupal erbjöd, och WolfCMS innehöll endast det nödvändigaste vilket räckte till i vårt fall. Även administrationsgränssnittet var tydligt, så det var lätt att komma igång med det. Dessa fördelar innebär snabbare funktionalitet och bättre överskådlighet, och troligtvis är även programmet lättare för slutanvändaren att behärska.

Nackdelarna med WolfCMS är lägre antal användare, så vid problem kan det vara svårare att få hjälp via deras forum. Detta leder också till att utvecklingen är långsammare. Det att WolfCMS är ett litet system, vilket vi räknade som en fördel, är även en nackdel med tanke på vidareutveckling av sidan. Detta hade liten tyngdpunkt vid val av CMS-system, eftersom beställaren inte hade några planer att bygga ut webbsidans funktionalitet i en nära framtid.

Vi valde också att inte använda Dropbox mera för uwamp-servern, eftersom vi ansåg att det inte var tillräckligt pålitligt ur funktionssynvinkel när webbsidan modifieras av två användare samtidigt, och genom att inte använda dropbox och skapa sidan i virtuell miljö så uppstår möjligtvis mindre överraskningar när hela projektet flyttas över till sin slutliga server. I detta skede ansökte vi om utrymme för webbplatsen via arkku.net. Ansökan godkändes, och webbplatsen flyttades till arkku.

Det fanns att välja en basversion och fullständig version av WolfCMS, och vi valde fullständiga versionen eftersom största delen av funktionerna i fullständiga versionen kommer att användas. På detta sätt blir det mindre moduler som behöver installeras i efterhand, och de moduler som finns med i fullständiga paketet som inte behövs går enkelt att inaktiveras.

Vi skapade en databas för projektet, och en användare som har rätt till databasen. Efter det började vi ladda upp filerna för WolfCMS. Vi skapade en mapp direkt under root på webbhotellet som vi döpte till projectx.

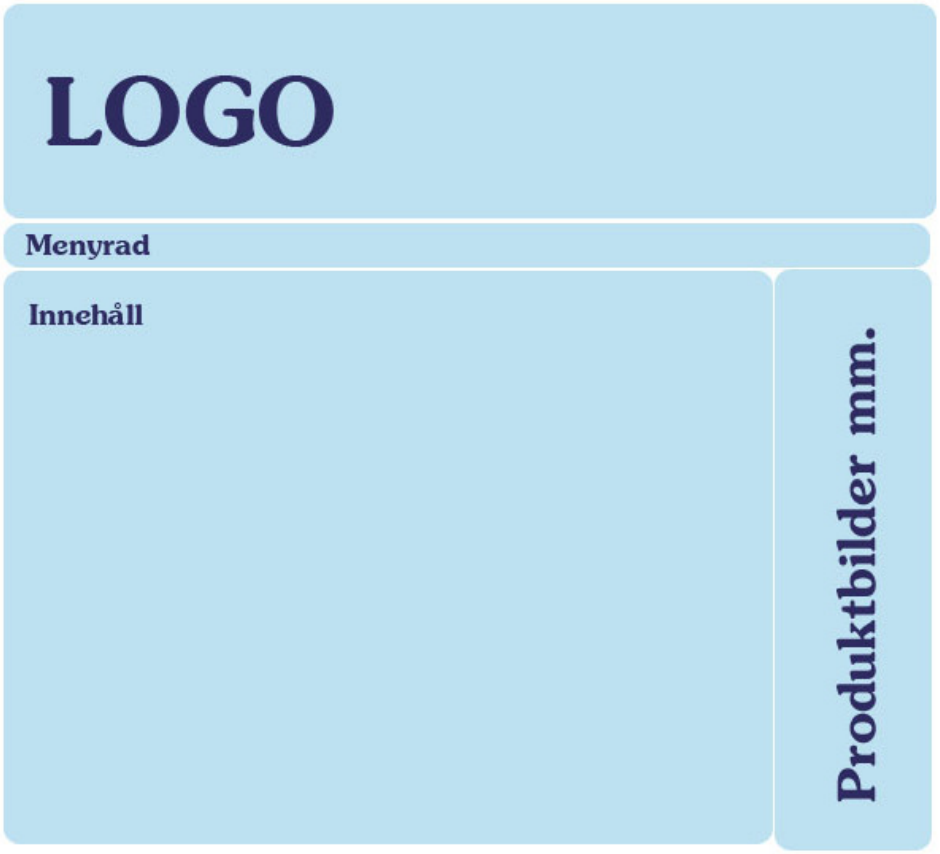
Efter detta konfigurerade vi filen config.php, och skrev in användarnamn och lösenord för databasen så kopplingen mellan sidan och databasen skulle fungera. När detta var gjort så loggade vi in som administrator på sidan för att kontrollera vilka funktioner det finns.

Efter detta började vi bekanta oss med ”how-to” informationen på WolfCMS sidor för att ha lite mera förkunskap om användandet av programmet, vilket sparade tid i det skedet när vi började arbeta närmare med systemet.

4 Design

För att få ett botten att börja arbeta på, så började vi fundera på webbplatsens layout. Vi ritade till en början en skiss på papper för att få en uppfattning av hur det kommer att se ut, och sedan ritade vi om den i Photoshop, se figur 1 nedan. Detta gjordes för att få bilden lättolkad och i digitalt format.

Vår plan var en lättnavigerad sida med en logisk layout. Logon skulle komma överst på sidan till vänster, med passande bakgrundsbild till höger om logon. Under logon skulle menybalken komma, och under den själva informationen. På första sidan skulle det även till höger om innehållsrutan vara en skild box med bilder och pris på utvalda produkter.



Figur 1: Första skiss ritad i Photoshop

När vi hade en bild av hur sidan skulle se ut började vi bygga upp layouten.

Eftersom WolfCMS har ett bra stöd för snippets så valde vi att istället för att ändra på ett befintligt tema så att det passar Kontoramas behov, skapa en ny layout från grunden med hjälp av HTML- och CSS-kod.

4.1 Basen för layout

Till WolfCMS finns det flera färdiga layouts som andra användare har skapat för att ibruktagningen av en sida skall vara en mindre invecklad process. Vi bestämde oss ändå för att göra en layout från grunden för att ha en bättre uppfattning om hur layouten är uppbyggd och för att kunna utveckla den så bra som möjligt för den slutliga produkten. Sidan får då också ett mer personligt utseende eftersom vi själv skapar designen från början till slut. Layouten är en stor del av projektet eftersom den interna designen, dvs. funktionaliteten bakom sidan skall vara relativt enkel.

Vid utveckling av layouten bestämde vi oss att följa de senaste standarder inom webbdesign, dvs. elementen på sidan är i form av div-taggar istället för tabeller och stilen på layouten skapas i form av CSS (Cascading Style Sheets) i en skild CSS-fil. All design i layouten är gjord med tanke på flexibilitet, överskådlighet och logiskt tänkande.

Hela layouten baserar sig på cirka sju div-element, beroende på om sidans innehåll är i en eller två kolumner, som det är t.ex. på kontakt -sidan. Den första div-elementen har vi döpt till logocontainer som har en bredd på 100 %. Inom elementen logocontainer har vi placerat ett till div-element som vi har döpt till logo. Denna lösning kom vi fram till för att få logon, som kommer att vara inom detta div-element, att följa samma vänstra marginal som övriga innehållet har på sidan.
Efter elementet logocontainer placerades ett div-element som vi döpte till nav, där huvudmenyn kommer att vara. Så som elementet logocontainer, kommer också detta element ha en bredd på 100 %. För att menyn inte skall placeras i vänstra kant, måste UL-taggarna vara specificerade så att UL-elementet kommer att vara centrerat på samma sätt som elementen logo.
Efter nav kommer det mer väsentliga innehållet på sidan och allt innehåll kommer att vara inom ett och samma element för att få bättre kontroll över dess placering.

Den viktigaste funktionen för detta element är att centrera allting inom det, så att vi inte behöver specificera det skilt för de övriga element som kommer att placeras inom body.

Inom body har vi skapat tre div-element; banner, container, footer. Banner, som vi också kallar för skyltfönster, har en bredd på 100 %. Det kommer dock inte att vara 100 % av webbläsarens bredd, utan kommer att följa den bredd som dess parent-div body har.

Som följande element skapades container. Här kommer allt väsentligt innehåll att placeras. Dess innehåll är också det enda som ändras när man navigerar på sidan, alla övriga element kommer i stort sätt att visa samma innehåll, oberoende var man är på sidan. Kontaktsidan kommer dock inte att ha innehållet inom detta element, eftersom innehållet kommer att placeras i två kolumner för att besökaren skall få en bättre överblick. På Kontaktsidan generas då två andra element, leftblock och rightblock och de kommer att innehålla informationen istället. De båda elementen har en bredd på 50 %, så att den totala bredden är den som är specificerad i elementen body. Rightblock har CSS-funktionen float:right, så att den placeras på samma höjd som leftblock.

Det sista elementet heter footer och är placerat direkt nedanför container. Här är det meningen att t.ex. kontaktuppgifter skall finnas.

Bilden på nästa sida, figur 2, visar de olika elementen med förklaringar som användes till den slutliga webbplatsen, samt var de är placerade för att enklare få en uppfattning om hur sidans layout är uppbyggd.

Figur 2: 1. Top 2. Meny 3. Skyltfönster 4. Reklamboxar 5. Innehåll 6. Footer

4.2 Header

Headern har 100 % bredd, med en ljusgrå gradient som bakgrund, vars färger går från gråa #EAEAEA till vitt, #FFF. Som innehåll finns Kontoramas logo och flaggorna som styr språkvalet.

Logon är placerad så att den följer sidinnehållets vänstra marginal, och storleken på logon är 300 pixlar bred och 52 pixlar hög. Bilden är en transparent PNG, för att bakgrunden skall synas bakom bokstäverna i logon.

För att logon inte skall bli vald vid kopiering av sidinnehåll, så valdes det att lägga logon som bakgrundsbild istället för att ha den som en vanlig bild.

Till höger i header lades språkvalet som en rund flagga. Flaggans storlek angavs till 35 pixlar hög och 35 pixlar bred. Även som med logon så är bildformatet för

flaggan en transparent PNG. Till flaggan angavs en marginal till höger på 30 pixlar, och uppåt 13 pixlar.

I headern placeras även en snippet mitt emellan logon och flaggorna, som kommer att användas för information som endast är aktuellt en kort tid, så som till exempel avvikande öppethållningstider och motsvarande. Storleken på denna angav vi till 40 pixlar hög och 350 pixlar bred, och rutan för innehållet skall inte synas när det inte finns något att visa. Detta löstes med hjälp av en if-sats i PHP, på samma sätt som med kolumnerna för innehållet.

4.3 Skyltfönster

Under menyn skapade vi en box som innehåller en bild som föreställer ett tangentbord och ett jordklot, och hela bilden är i blå nyans. Det är oklart om denna bild kommer att finnas på den slutliga sidan, men den placerades på plats för att få en överblick hur sidan kommer att se ut.

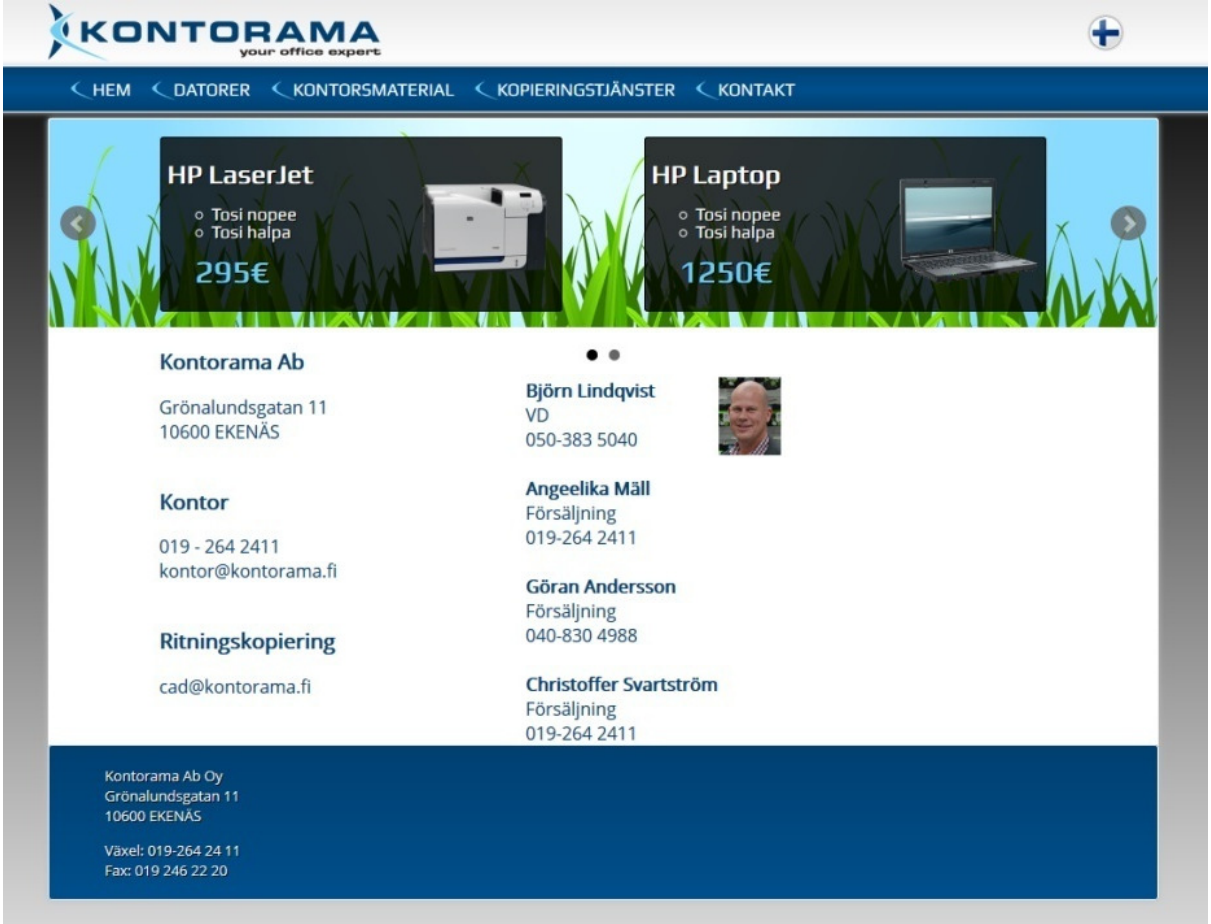
Bredden på bildboxen bestäms av bredden på containern som innehåller alla webbplatsens element. Höjden angavs till 187 pixlar. Hörnen har en radius på 3 pixlar för att skapa runda kanter. Runt skyltfönstret är det en border med en bredd på 1 pixel, vit till färgen.

I detta element placerades även två stycken reklamboxar, som kommer att beskrivas närmare i ett senare skede.

4.4 Innehåll

Innehållet placerades under elementet vi kallar för ”skyltfönstret”. I innehållet visas all aktuell information för sidan man navigerar till. Bredden på innehållet angavs till 80 % av sidans bredd genom att lägga 10 % marginaler till vänster och höger. Bredden på innehållet är alltid 80 % av ”parent-div” för att underlätta användandet av sidan med mobila enheter. Minimihöjden på innehållet angavs till 200 pixlar, och höjden är responsiv.

Det skapades även ett alternativ att dela upp innehållet i två kolumner på sidor, som t.ex. kontaktsidan. Denna lösning gjorde vi för att på sidor där innehållet är i format som t.ex. kontaktuppgifter, att det är kort informationstext, undvika att det blir mycket innehåll på höjden, och istället fylla ut mera på bredden, se figur 3.



Figur 3: Adressuppgifter i vänster kolumn och personalen till höger.

Denna lösning är även optimal i detta fall eftersom man kan se all information som finns på sidan utan att behöva navigera uppåt och neråt.

Detta löstes med hjälp av en PHP if-sats. Koden kontrollerar om det finns innehåll i rightbox, och om det finns så delas innehållet upp i två kolumner med hjälp av div-taggar. Exempel på koden som kontrollerar rightbox, se kodexempel 2 nedan. Motsvarande kodsträng finns också för leftbox.

Kodexempel 2: Kod som kontrollerar om det finns innehåll i rightbox

```
<?php if ($this->hasContent('rightbox')) printf ('<div id="rightbox">%s</div>', $this->content('rightbox')); ?>
```

Vid skapande av innehåll som delas upp i två kolumner så öppnar man den sidan man vill editera. Sedan skapar man två nya tabs, utöver bodyn som automatiskt genereras när man skapar en ny sida. Dessa två tabs döps sedan till leftbox och rightbox, så att if-satserna kan hämta innehållet och skapa kolumnerna. Viktigt är i detta skede att man inte har något innehåll i body-fliken när man tar i bruk kolumnerna.

4.5 Vänster balk, höger balk

På båda sidorna om innehållet var det till en början meningen att vi skulle ha balkar med olika information.

I den vänstra balken var det meningen att kontaktuppgifter skulle synas hela tiden i kort format, och den högra balken var menad för reklamboxarna.

Vid utveckling av sidan beslöt vi oss ändå för att flytta reklamboxarna upp till ”skyltfönstret” som är placerad under menyn. Detta tyckte vi gav ett mer modernt utseende, och gav mera utrymme för själva innehållet.

Denna åtgärd ledde till det att höger balk blev obehövlig, och då tyckte vi att det inte heller layoutmässigt var bra att ha en balk till vänster om innehållet, så båda balkarna slopades i slutskedet av projektet.

4.6 Sidfot

Sidfoten är placerad direkt under innehållet. Bredden angavs med CSS-funktionen calc till 100 % - 102 pixel, och höjden till minimum 50 pixlar, vilket sedan justeras automatiskt efter mängden innehåll. Det lades till 5 pixlar padding både till vänster och höger. Som bakgrund lades en gradient blå färg, som går från en blå färg, #004071, till en ljusare blå, #004F8C. Hörnen på sidfoten gjordes runda med hjälp av CSS-funktionen border-radius 3 pixlar.

Fonten är Open Sans och har en skuggning som är 1 pixel stor. Fontens färg angavs till vit, #FFF.

Innehållsmässigt skall sidfoten ha rubriker, som t.ex. kontorsmaterial, IT-tjänster och kopiering, och under dessa listas produkter och/eller tjänster som hör till rubriken ovanför. Längst till vänster kommer kontaktinformationen i sin korthet, så att det lätt går att hitta basinformation som telefonnummer och adress till företaget. För att få innehållet snyggt uppdelat skapades det 4 div-element som innehåller informationen som skall synas.

Längst till höger i sidfoten kommer det att placeras en Facebook ”like-box”, vilken är klickbar och länkar direkt till Kontoramas Facebook-sida.

4.6.1 Facebook like-box

För att få Facebook like-boxen till sidfoten så börjar man med att gå till adressen https://developers.facebook.com/docs/plugins/like-box-for-pages/ , där man matar in URL till den Facebook-sida man vill skapa en like-box för. Efter detta genererar sidan en box, och koden som skall användas för att implementera denna till webbplatsen. Det går ytterligare att göra några små justeringar för utseendet på boxen direkt från det grafiska gränssnittet, som t.ex. höjden som specificerades till 180 pixlar, och border togs bort.

För att få Facebook like-boxen att passa utseendemässigt in på sidan så behövdes små justeringar göras med hjälp av CSS. Bakgrunden angavs till vit #FFF, för att texten inte var läsbar mot footerns blåa bakgrund. Även hörnen rundades till med hjälp av funktionen border-radius som angavs till 3 pixlar. Senare upptäckte vi att det gick att göra färgschemat för innehållet mörkare, så vi gjorde detta och avlägsnade den vita bakgrunden, eftersom texten nu var läsbar mot blå bakgrund. Utan bakgrund passade boxen även bättre ihop med sidans utseende. Placeringen ändrades från höger hörn till vänstra, eftersom boxens layout passade bättre in till vänster i footer.

Till en början hade vi problem att få boxen placerad så som vi ville ha den i footer. Detta löstes när vi tog Responsive Grid System i bruk, och använde oss av egenskapen display, med värdet inline-block.

4.7 Meny

När vi hade en fungerande layout så började vi med att få menyn på rätt plats. Vi tog en färdig kodsträng från WolfCMS forumet som hämtar innehållet skapat under fliken ”sidor” i administrationsgränssnittet, se kodexempel 3 nedan som visar kodsträngen. Menyn placerades in under header, och ovanför skyltfönstret. Som standard var inte menyn visuellt tilltalande, så detta åtgärdades med att ändra utseendet med hjälp av CSS.

Kodexempel 3: Kod som skapar menyn.

```
<div id="nav">
 <ul>
   <li<?php echo url_match('/') ? ' class="current"': ''; ?>><a href="<?php echo URL_PUBLIC; ?>">Home</a></li>
    <?php foreach($this->find('/')->children() as $menu): ?>
      <li <?php echo in_array($menu->slug, explode('/', $this->url)) ? ' class="current"': null; ?>>
     <?php echo $menu->link($menu->title); ?></li>
     <?php endforeach; ?>
    </ul>
  </div> <!-- end #navigation -->
```

Menybalkens övre och nedre kant har en border som är 1 pixel bred och är vit till färgen, bredden angavs till 100 % och höjden till 40 pixlar. Menybalken har också en skuggning som är grå till färgen, #CoCoCo, och 5 pixlar bred. Detta gjordes för att skapa en mjukare övergång till bakgrunden. Bakgrunden är gradient som går från en ljusare blå färg till mörkare mot övre och nedre kanten. För att undvika användandet av bilder så valde vi att göra detta genom att endast använda oss av CSS-funktionen linear-gradient, och färgerna som skulle användas.

Fonten till menyn hämtas från tjänsten Google fonts, och den heter Play. Valet av font gjordes på basen av utseendet, som i vårt tycke var modernt och visuellt tilltalande. Vi valde senare att vi skulle ladda ner fonten och ha den på samma webbhotell som själva webbplatsen är på, för att försäkra oss om att fonten även finns kvar i fortsättningen, eller byter plats och därmed URL.

Menytexten skrivs ut med versaler, och är vit när den inte är vald (a:link). När man för muspekaren över ett menyval (a:hover) så blir texten blå till färgen, #6ECFF6, och det blir en skuggning bakom texten som är 1 pixel stor.

För att ytterligare göra menyn mer visuell så placerades Kontoramas ”båge” framför varje menyknapp.

När menyn var utseendemässigt bra så skulle den ännu programmeras för att fungera ihop med sidans tvåspråkighet, vilket vi kommer att berätta mera om i ett senare skede.

4.7.1 Undermeny

Till en början så var det planerat att det skulle finnas en undermeny som skulle placeras under huvudmenyn, eftersom det finns så många underkategorier för t.ex. kontorsmaterial och datorer. Idén med undermenyer slopades ändå vid slutskedet av arbetet, eftersom vi konstaterade att det inte finns tillräckligt med innehåll för varje enskild tjänst, så vi valde att inte använda oss av undermenyer.

Istället bestämde vi oss för att skriva med större rubriker direkt under huvudrubriken det som skall finnas under den kategorin, så allting under en viss meny blir synligt på en gång, utan underrubriker.

Vi lämnade ändå möjligheten att använda undermenyer kvar ifall det i ett senare skede skulle behövas. Undermenyn har en funktion vilken gör att den inte syns när det inte finns innehåll att visa. För att skapa en underrubrik till en sida så väljer man i administrationsgränssnittet att skapa en child-page till den sidan man behöver en underrubrik till. När detta är gjort så genereras undermenyn automatiskt.

4.8 Bakgrund

Som bakgrund valde vi vit färg, med ett rutmönster. Detta val gjordes för att få lite djup i sidan, och rutmönstret gjordes väldigt diskret för att inte sticka ut för mycket.

När vi hade layouten så långt klar att det påminde om en webbsida så matade vi in koden i WolfCMS som en ny layout, och döpte den nya layouten till ”Kontorama”. När vår kod var sparad som en layout så valde vi vår nyskapade layout som standardtema. Temat fungerade och såg ut som det skulle, så nu var layouten på WolfCMS frontend precis sådan som vi ville ha den.

4.9 Favikon

Som favikon till sidan tänkte vi först använda Kontoramas båge, men beslöt oss för att använda hela logobilden utan text istället, eftersom vi ansåg att detta fyller ut favikonens utrymme bättre, och ser visuellt bättre ut.

För att få favikonen i rätt format, det vill säga 16x16 pixlar och filformatet .ico, så modifierades den i Adobe Photoshop. Utöver storleken och filformatet valde vi även att göra bakgrunden transparent.

Vi visste från tidigare att det inte går att spara bilden i .ico-format från Photoshop, så vi valde att spara bilden som .png, och därefter ladda upp den till en tjänst på nätet. Denna tjänst omvandlar sedan bilden till .ico, och efter det är den färdig att laddas upp och användas på webbplatsen. Tjänsten vi använde oss av hittas på adressen http://favicon-generator.org/.

För att webbläsaren skall hitta favikonen så skall den laddas upp direkt i root-mappen på webbhotellet och döpas till favicon.ico. Några andra åtgärder behövs inte göras, eftersom webbläsare är färdigt programmerade att söka efter favikon på detta ställe.

4.10 Reklamboxen

Till en början var reklamboxens placering till höger om innehållet, men eftersom höger och vänster balk avlägsnades så placerades reklamboxarna uppe i skyltfönstret var det fanns ledigt utrymme istället.

Till en början var det meningen att ha tre till fyra reklamboxar synliga åt gången, men det blev till slut två lite större reklamboxar för att få rum till all behövlig information som skall visas i dem.

För detta ändamål skapades sex produktboxar. Meningen var att två boxar visas samtidigt, och de skall automatiskt gå vidare till de två nästa boxarna i bildspelet, och när sista boxarna har visats så börjar bildspelet om från början.

När alla containers var skapade så började vi med att lägga till snippets-informationen i koden, och via WolfCMS administratörgränsnittet skapade vi sex

stycken snippets, en för varje produktbox, med samma namn som vi specificerat i snippet-informationen. I detta fall blev namngivningen på boxarna product1 till product6. Med denna namngivning så är det lätt och överskådligt att se i vilken ordningsföljd boxarna visas på sidan. Senare ändrades namnen till produkt 1-6 för de svenska produktboxarna, och tuote 1-6 för de finska för att enklare skilja på dem.

När detta var klart så prövade vi funktionen genom att lägga till information i produktboxarna, och det fungerade som det var tänkt.

4.10.1 Reklamboxen animering

Eftersom ett av beställarens krav var att produktbilderna i reklamboxarna skulle ändras automatiskt, så började vi se på olika möjligheter för att förverkliga detta krav. Alternativen som var aktuella var att ha en gallerimodul var man kunde ladda upp en bild, och sedan fylla i en bildtext som skulle synas i reklamboxen tillsammans med bilden. Eftersom utbudet av färdiga moduler inte är så stort till WolfCMS, så hittade vi inte någon färdig modul gjord för WolfCMS som uppfyllde kraven vi ställde.

Efter det började vi söka efter program som inte direkt var moduler, men som var möjliga att programmera in för att fungera ihop med sidan. Många program vi stötte på var enbart menade för bilder, men sedan hittade vi ett program vid namnet Bxslider som verkade lovande, eftersom det enligt specifikationerna skulle stöda bilder, text samt HTML-innehåll.

4.10.2 BxSlider

BxSlider är ett slideshow-program som stöder bilder, video och HTML-innehåll. Det är responsivt, så det fungerar bra på både dator, tablett och telefoners webbläsare. Även stöd för olika browsers är bra. Vidare så har det stöd för touch-scrolling, har väldigt mycket inställningsmöjligheter och programmet tar endast lite rum på webbservern. (BxSlider, u.å.)

4.10.3 Implementering

När vi sedan tog ner koden för programmet och de färdigkonfigurerade programfilerna så började vi implementera programmet. Vi började med att ladda upp mappen med filerna som behövs till webbhotellet, och mappen placerades direkt i roten.

Därefter tog vi upp koden för layouten vi skapat via WolfCMS kontrollpanel. Nästa steg blev att kalla på Jquery, vilket vi till en början söker från Googles servrar. Detta kommer möjligtvis att ändras i ett senare skede, för att inte vara beroende av program som finns lagrat på en utomstående server.

Nästa steg är att definiera var JavaScript- och CSS-filen för BxSlider finns, och i detta fall var sökvägen i mappen BxSlider direkt i rotkatalogen. Exempel på hur koden för dessa funktioner ser ut syns på kodexempel 4 nedan.

Kodexempel 4: Kod som kallar på Jquery, samt sökväg för BxSlider:s filer

```
<!-- jQuery library (served from Google) -->

<script
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>

<!-- bxSlider Javascript file -->

<script src="/bxslider/jquery.bxslider.min.js"></script>

<!-- bxSlider CSS file -->

<link href="/bxslider/jquery.bxslider.css" rel="stylesheet" />
```

När man har länkat till Jquery-biblioteket, samt till de filer som BxSlider använder, så blir nästa steg att kalla på funktionen BxSlider med hjälp av Javascript, se kodexempel 5 på nästa sida. Sedan skall det väljas var på sidan man vill placera in funktionen. Till en början placerade vi slidern direkt i skyltfönstret för att kontrollera funktionen. Vi prövade att lägga till några bilder och text som en snippet till slidern, eftersom vi till en början var skeptiska till om det skulle fungera. Kodexempel 6 på nästa sida tydliggör hur innehåll länkas till BxSlider. När allting var inmatat så prövade vi funktionen som fungerade perfekt med olika typers innehåll, även vår snippet bestående av en bild och text visades så som det skulle. Det var en lättnad att allting fungerade som det var tänkt direkt, och vi gick

vidare och började anpassa BxSlider så att den funktionsmässigt och utseendemässigt uppfyller våra och kundens önskemål.

Kodexempel 5: Koden som kallar på funktionen BxSlider

```
<script>
$(document).ready(function(){
  $('.bxslider').bxSlider();
});
</script>
```

Kodexempel 6: Hur innehåll (snippet) länkas till BxSlider

```
<ul class="bxslider">
<li><?php $this->includeSnippet('produkt1') ?></li>
</ul>
```

4.10.4 Modifiering av BxSlider för att passa vårt behov

För att få BxSlider att passa sidans utseende så behövdes vissa ingrepp göras för att den skulle vara funktions- och utseendemässigt anpassad för att passa in på sidan.

I första versionen av BxSlider så passade inte navigeringsikonerna ihop med våra reklamboxar, eftersom de var för stora. Vi modifierade de befintliga genom att ta bort den runda bilden runt själva pilen, så endast pilen blev kvar. Dessa gjorde vi även mer transparenta så de inte påverkade läsbarheten fast de blev över innehållet. Efter att reklamboxarna flyttades upp till skyltfönstret beslöt vi ändå att använda oss av de ikoner som var standard, eftersom placeringen nu var annorlunda.

För att vi ville ha en mer levande sida, så valde vi att låta bildspelet automatiskt gå framåt, och efter sista produkten/reklamen så börjar det om från början igen. Detta gick att göra genom att ändra värdet på kommandot auto från false till true.

Som standard så blir det paus på bildspelet när man väljer att manuellt gå framåt eller bakåt i bildspelet. Denna funktion gillade vi inte, utan vi tyckte det var bättre att bildspelet efter en stund fortsätter som vanligt, även om användaren har tryckt

på framåt eller bakåt. Det fanns inte något färdigt kommando för att utföra denna funktion, så vi fick lägga till lite kod för att gå förbi denna egenskap.

Det vi gjorde var att vi sökte upp inställningen i koden där det står att bildspelet skall pausas när användaren navigerar framåt eller bakåt i bildspelet, och på dessa platser lade vi in en kodsträng som startar bildspelet igen efter att användaren navigerat.

Koden som användes gör att bildspelet återgår till den funktionsprincip som det har från början, fortsätter spela bilder som vanligt. Det pausas, men med hjälp av koden så återaktiveras det. Kodexempel 7 nedan visar funktionen för retur i bildspelet, samma princip användes för att vid manuell navigering framåt i bildspelet återuppta automatiska framåtspolningen.

Kodexempel 7: Koden som användes för att undvika paus av bildspelet

```
var clickPrevBind = function(e){

// if auto show is running, stop it

if (slider.settings.auto) el.stopAuto();

el.goToPrevSlide();

e.preventDefault();
```

För att slumpmässigt välja vilken bild som visas när sidan laddas, så ändrade vi värdet på kommandot randomStart från false till true. Denna funktion ville vi ha så att användaren inte alltid skall se bildspelet från första produkten/reklamen varje gång man klickar på en länk för att navigera på webbsidan.

Som standard har BxSlider ett navigeringsverktyg med vilket man ser hur många bilder det sammanlagt är i bildspelet, och man kan hoppa till en viss bild genom att trycka på motsvarande punkt i navigeringsverktyget. Detta behövde vi inte, eftersom det inte var intressant för oss att ha den funktionen, och så ville vi ha så lite störande element som möjligt vid reklamboxarna, så denna navigeringsmeny av ”bullettyp” kopplade vi ur funktion genom att ändra kommandot pager från true till false.

Det fanns också en funktion som gav användaren möjlighet att stoppa och starta bildspelet, men detta behövdes inte heller, så den funktionen stängdes genom att

ändra värdet för autocontrols till false. Nu började reklamboxarna ha den funktionsprincipen vi var ute efter, och kvar var endast lite finjustering, som att ställa in tiden för hur länge varje bild visas till aningen långsammare. Detta ville vi göra eftersom det som standard byter bild ganska ofta, och med det innehåll som reklamboxarna kommer att ha så var det aningen för snabbt, speciellt när det är två reklamrutor synliga samtidigt.

Tiden hur länge bilderna visas fanns under pause i BxSlider. Värdet var som standard 4000, vilket vi ändrade till 7000. Detta värde stod för millisekunder enligt våra mätningar, så det blev 3 sekunder längre tid som boxarna visades före de ändrades till nästa. Även hur snabbt överflyttningen till nästa boxar sker modifierade vi under speed, vilket ändrades från 500 till 1500 för att skapa en mjukare övergång.

Efter dessa ändringar så var reklamboxarnas utseende och funktionalitet i sådant skick att vi var nöjda, och vi gick vidare med arbetet.

4.10.5 Problemlösning

I BxSlider stötte vi på två problem. Det första problemet var att när en bild och text var inmatade i en reklambox, så blev bilden för lågt ner, utanför själva boxen. Efter att vi prövat lägga till innehåll på olika sätt så upptäckte vi att felet uppstod om man först skrev texten, och lade bilden efteråt. Om man gjorde tvärtom, det vill säga att man först placerade bilden och sedan skrev texten, så visades bilden korrekt i reklamboxen.

Andra problemet med BxSlider som vi märkte var att vid manuell förflyttning bakåt från de första bilderna till de sista, emot BxSliders standard förflyttning så åkte bilderna lite för långt, och sedan förflyttas de till rätt plats på ett sätt som de inte borde. Detta stör inte funktionen och förhindrar inte heller användningen av sidan, men är ett visuellt problem som vi inte hittade någon lösning på.

4.11 Webbfonter

Vi bestämde oss att använda open-source fonter som kan hämtas från tjänsten Google Fonts (Google). Det är fråga om fonter som är optimerade för webben och

alla moderna webbläsare stöder dem. De kan hämtas enkelt med CSS-funktionen @import från Google Fonts, varefter de kan tas i bruk med CSS-egenskapen font-family.

Fördelen med dessa fonter är att man inte behöver hänvisa skilt för varje operativsystem vilken font skall användas, istället anger man endast en font eftersom webbläsaren importerar och renderar fonten, istället för att använda sig av de fonter som finns färdigt i operativsystemet. En annan fördel är att det finns ett stort urval och det är enklare att hitta en font som passar in i layouten, så man är inte beroende av standardfonter.

En nackdel med webbfonter är att de inte renderas felfritt i alla webbläsare i vissa situationer, som t.ex. när bakgrunden bakom texten har en gradient färgövergång. Detta problem förekommer i nyaste versionen av Google Chrome, men i Internet Explorer 11 och Mozilla Firefox renderas webbfonter felfritt.

4.12 Jquery

Jquery är ett JavaScript-bibliotek, som innehåller många funktioner men är samtidigt litet till storleken och snabbt vid användning. (Jquery, u.å.)

Vi använde oss av Jquery i arbetet för att få animationerna i BxSlider att fungera.

5 Optimering av sidan

För att webbplatsen skall vara så snabb som möjligt så funderade vi på flera olika lösningar hur vi kunde göra sidans funktionalitet ännu snabbare, navigeringen lättare samt optimera sidan för sökmotorer. Personligen var vi nöjda med snabbheten, men försökte ändå få lite snabbare laddningstider med hjälp av olika metoder.

5.1 URL

URL, som står för Uniform Resource Locator (på svenska webbadress) är teckensträngen som identifierar resurser på nätet. (Wikipedia, 2013)

I detta stycke beskriver vi hur vi gjorde för att få en ren URL, istället för att visa den URL som WolfCMS genererar, vilken innehåller ett frågetecken som vi ville avlägsna.

En ren URL är både mer överskådlig för användaren och bättre med tanke på sökmotoroptimering, eftersom sidan man är på är utskriven, som t.ex. kontorama.fi/kontakt, istället för att det är flera genomslag och vilseledande information i URLen. För att få detta gjort så modifierade vi filerna .htaccess och config.php som finns i WolfCMS-mappen. Före utförandet av följande åtgärder så är det skäl att ta reda på om webbservern stöder denna funktion.

Först modifierade vi .htaccess, som är en konfigurationsfil med information som berättar åt webbservern hur webbsidor skall visas. Filen används i huvudsak till att modifiera URL:er, begränsa och hantera rättigheter och hantera cachen. (Wikipedia, 2014)

Filen heter från början _.htaccess, eftersom den inte används av sidan. För att ta i bruk denna så tas understrecket bort så filens namn blir .htaccess.

De ändringar som gjordes i .htaccess var att till en början med ge korrekt sökväg, som hade standardvärdet RewriteBase /wolfcms, men ändrades till RewriteBase /, eftersom sidan är direkt under rotkatalogen.

RewriteBase är en regel i Apache, som är ett sätt att modifiera inkommande URL-förfrågningar. Regeln kan effektivt ändra invecklade URL-adresser så att de blir till ren text som är enkel att tolka. (Apache, u.å.)

Sedan ändrades filen config.php, som finns i htdocs. Här ändrades värdet för ”'USE_MOD_REWRITE'” från false till true. Genom att ändra detta värde så berättas det för systemet att filen .htaccess skall användas.

När dessa ändringar var gjorda så prövade vi att URLen såg ut som den skulle. Frågetecknet var borta, vilket tyder på att allting gick som planerat.

5.2 Sökmotoroptimering

Sökmotoroptimering (härefter SEO, vilket står för Search Engine Optimization) är ett begrepp som används när man med olika metoder försöker få en webbplats att hamna så högt upp som möjligt i sökmotorers resultat, och på detta sätt få flera besökare till webbplatsen. För att en webbplats skall hamna så högt upp som möjligt i sökresultaten så skall man veta vad sökmotorernas algoritmer tar hänsyn till på en webbplats, och optimera dessa faktorer. (Wikipedia, 2014)

För att sökmotorer skulle hitta sidan lättare så började vi med att göra en ren URL, som vi beskrev tidigare. Det är önskvärt att ha rena URL:s , eftersom sökmotorernas spindlar som söker efter innehåll på nätet indexerar rena URL-adresser lättare än statiska, och klarar av att få träffar på rätt sidor när personer söker efter t.ex. Kontorsmaterial Ekenäs, när adressen är i form kontorama.fi/kontorsmaterial, istället för en URL som inte betyder något eller innehåller mycket onödig information.

Dessa spindlar, som även kan kallas crawlers eller bots, är de som söker informationen som sedan sparas i sökmotorns index-fil. Namnet spindel kommer från denna bots funktion, eftersom den ”kryper” över sidan och samlar information, och sedan för det till index-filen. Olika sökmotorers spindlar kan fungera lite olika, men grundprincipen är densamma för alla. (Kathy Ivens, 2003)

Även valet att inte använda undermenyer är bra med tanke på SEO, eftersom spindlarna som söker innehållet hittar mera information under samma sida, vilket gör det lättare för dem att indexera innehållet, och därmed få bättre sökträffar.

5.3 Responsive Grid System

För att innehållet skulle vara mer responsivt med tanke på användare med mobila enheter så använde vi oss av Responsive Grid System i footern och på kontaktsidan, var vi använde oss av kolumner.

Fördelen med Responsive Grid System är att man kan ha så många kolumner man själv vill, och man behöver inte ha ett visst antal kolumner över hela sidan, eftersom det går att variera. Andra fördelar är att man kan skapa kolumnerna efter

innehållet, och inte tvärtom. Det bakas enkelt in i befintligt HTML- och CSS-script, och använder sig av procentuella värden istället för pixlar, vilket gör det enkelt att anpassa in var som helst på webbplatsen. (Responsive Grid System, u.å.)

5.4 Normalize.css

Normalize.css är en färdig CSS-fil som kan laddas ner från nätet. (Normalize) Det är frågan om en CSS-fil, vars uppgift är att normalisera vissa CSS-funktioner så att olika webbläsare visar innehållet enligt de regler som finns i filen. Detta innebär att layoutens olika delar visas på samma sätt i olika webbläsare, även om de inte följer samma standarder. Normalize.css togs i bruk genom att ladda upp filen till root-katalogen på webbplatsen och sedan tillsattes CSS-funktionen @import till den primära CSS-filen som hänvisar till normalize.css.

5.5 Responsiv design

Responsiv design betyder att man bygger upp webbplatsen så att användarupplevelsen är bra oberoende om man använder en persondator eller en mobiltelefon för att besöka webbplatsen. Responsiv design är viktig i dagens läge, eftersom det mobila surfandet förväntas gå om det traditionella datorsurfandet inom några år. (Internetworld.idg.se, 2012)

Vi har gjort webbplatsen delvis responsiv, men inte till 100 %. Detta val gjordes eftersom surfplattor och moderna telefoner i dagens läge har så pass bra upplösning att det inte är ett så stort problem med surfande med mobila enheter som det ännu var för en tid sedan, och delvis för att vi inte hade detta som ett klart mål eller krav direkt från början.

5.6 Optimering av koden

För att få koden validerad körde vi först W3schools validator av webbplatser på sidan http://validator.w3.org/. Vi körde test på både HTML-scriptet och CSS-scriptet, och testerna slutfördes med färre fel än vi trott, och de flesta felen var

såpass harmlösa att vi beslöt oss att inte åtgärda dem, eftersom de inte påverkade sidans funktionalitet och utseende. Ett av felen, eller varningarna, var t.ex. att en border hade samma färg som bakgrunden.

5.7 Snabbhet

Vi hade redan i början av projektet, och det var även ett av kraven från kundens sida, att ha en så snabb webbplats som möjligt. Ungefär halvvägs i projektet så prövade vi ett online-test som mäter hur snabbt sidan laddar, för att se om vi var på rätt väg, vilket såg lovande ut. Testet visade att sidan laddades på 2,2 sekunder, och hade snabbare laddningstid än cirka 70 % av alla sidor testade via Pingdom. Det som tog mest tid att ladda var bilderna, vilket inte var någon överraskning, men gav oss lite riktlinjer att fundera över hur vi kunde få sidan ännu snabbare.

5.7.1 Jämförelse

För att ha någonting att jämföra med, så testade vi även några andra, lokala företags webbplatser. Vi körde testet på en webbplats med liknande animationer som webbplatsen vi utvecklar har, och den hade betydligt långsammare laddningstid, 5,9 sekunder. Vi prövade också en webbplats som var väldigt simpel och saknade animationer, och laddningstiden blev 2,2 sekunder. Detta var aningen snabbare än Kontoramas webbplats laddningstid, så nu hade vi en laddningstid som vi hoppades slippa under, även om webbplatsen vi arbetar på hade mera animationer än den webbplatsen som laddade snabbare.

Verktyget vi använde oss var ett online-verktyg för mätning av laddningstider som upprätthålls av Pindgom. Denna tjänst visar hur länge det tar att ladda varje fil som krävs för att webbplatsen skall fungera, går att testa med flera olika servrar och det går att jämföra sitt eget test mot de tester som övriga användare gör genom Pingdom. (Pingdom)

Vissa aspekter som vi måste ta i beaktande var att webbsidan inte ännu var på sitt rätta webhotell, vilket kan påverka den slutliga laddningstiden. I nuläget strävar vi att komma ner till två sekunders laddningstid, eller mindre. För att uppnå detta

mål skall vi optimera koden så bra vi kan, samt se över bildernas filstorlekar.

5.7.2 Testmetoden

Vi märkte dock efter en stund att vid upprepade tester så varierade laddningstiderna väldigt kraftigt. Dessa variationer kan förklaras med olika belastning av nätverket samt varierande belastning på webhotellet där sidan finns. Vi funderade på att mäta flera gånger och ta ett medeltal av värden, vilket vi till en början ansåg att inte skulle vara tillräckligt exakt, men senare beslöt vi oss för att använda denna metod.

Vår nästa idé var att köra webbsidan lokalt på datorn, och hitta ett lämpligt program för att pröva hur snabbt sidan laddar. Denna metod borde vara ganska exakt, så länge testerna körs på en och samma dator som har samma belastning hela tiden, eftersom brytningar och svängningar i nätverket inte slipper och påverka slutresultatet. Olika belastning på servern där webbsidan finns skulle inte heller påverka detta, eftersom vi kan kontrollera att belastningen hålls samma hela tiden.

Problemet med att köra snabbhetstestet lokalt hade dock en nackdel, vi kunde inte jämföra webbplatsen med motsvarande webbplatser som var ute på nätet. Detta innebär att vi inte skulle ha något att jämföra vårt resultat med, så vi beslöt oss ändå för att använda oss av Pingdoms snabbhetstest, och ta medelvärdet av ca 10 testkörningar.

När vi hade webbplatsen så gott som klar påbörjade vi snabbhetstesten igen. Vi hade planerat att pröva webbplatsens snabbhet som den är, men även med en modul som heter Funky Cache. Denna modul cachar sidorna och skall snabba upp laddningstiderna, eftersom det inte då behövs hämtas någon information från databasen. Tyvärr kunde vi inte förverkliga denna idé, eftersom Funky Cache inte fungerade ihop med nyaste versionen av WolfCMS. Detta leder till att testandet med ytterligare funktioner som skall snabba upp webbplatsen lämnas bort.

Medeltalet av 10 testkörningar gav oss tiden 1,17 sekunder. Detta ansåg vi vara en bra tid, och körde testet på några motsvarande, lokala företags webbplatser för att ha någonting att jämföra med. Den ena webbplatsen vi jämförde med innehöll en hel del grafiska element och gav sluttiden 3,11 sekunder. Den andra webbplatsen,

som är väldigt simpel och som vi antar att var statisk, gav tiden 1 sekund. Vi var nöjda med dessa tider även om webbplatsen vi utvecklar inte var snabbast, men med tanke på hur mycket innehåll och funktioner som skulle laddas på webbplatsen vi utvecklade så ansåg vi att 1,17 sekunder var bra i jämförelse med de två andra webbplatserna.

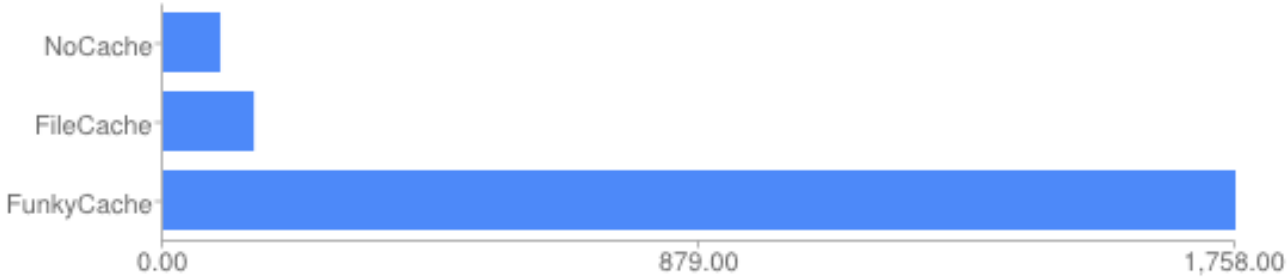
Slutligen körde vi ännu testet på samma webbplatser med Firefox inbyggda test. Detta test visade bättre laddningstider än Pingdoms test, i medeltal 30-50% snabbare, och slutresultatet avvek inte från den information vi fick av Pingdoms test.

5.8 Funky Cache

När vi gick igenom moduler som fanns tillgängliga till WolfCMS så stötte vi på en modul vid namnet Funky Cache. Funky Cache funkar genom att skapa en statisk fil av innehållet på webbplatsen, så att kommandon som görs på webbplatsen inte behöver gå via PHP-gränssnittet, och därmed blir sidan snabbare. Funky Cache kan bevisligen klara av slashdotting bättre än en sida som inte har Funky Cache installerat, se figur 4 nedan.

Med slashdotting menas när en större webbplats länkar till en mindre webbplats, vilket leder till en enorm ökning av trafik, och resultatet är ofta att webbplatsen blir överbelastad och slutar fungera.

När innehållet på webbplatsen uppdateras så utgår den cachade filen automatiskt, och sidan cahcas om automatiskt nästa gång den laddas. (WolfCMS, 2014)



Figur 4: Diagram över hur många förfrågningar per sekund en sida klarar av med och utan Funky Cache

Funky Cache integrerades enkelt genom att ladda upp filerna till mappen wolf/plugins/funky_cache på servern. Efter det skall man kontrollera att inställningen use_mod_rewrite i config.php filen har värdet true, vilket det i vårt fall redan hade eftersom inställningen ändrats i ett tidigare skede.

När detta är klart skall modulen aktiveras, vilket görs genom WolfCMS administratörgränssnitt. Efter aktiveringen skapas en ny flik som heter cache. Här skall man ännu kopiera en kodsnutt som skall skrivas in i filen .htaccess, som hämtar de cachade filerna istället för sidans ”riktiga” filer. Efter detta kan man ställa in vilken mapp filerna skall lagras i, vilket i vårt fall blev mappen/cache i roten på WolfCMS.

Det går även att vid varje enskild sida under fliken sidor att ändra om man vill att sidan skall cachas eller inte.

Under implementeringsskedet funderade vi på om denna plugin kunde ha någon negativ effekt på användandet av webbplatsen, men vi kunde inte komma på något negativt i detta skede, men beslöt för att lägga med test av denna funktion till det slutliga testet som vi planerat göra när allting börjar vara klart.

Vid implementering av Funky Cache till den slutliga versionen av webbplatsen stötte vi på problem. På testsidan fungerade Funky Cache som det var tänkt, men på slutliga webbplatsen så visades endast källkoden för den sidan man försökte besöka när sidan var cachad. Till början trodde vi att det blivit något enkelt fel vid implementeringen, men vid kontroll såg vi att det var gjort på rätt sätt. Vår nästa tanke var att menyns omit-pages funktion kunde ställa till med problem, eftersom detta inte fanns med på den första versionen av webbplatsen.

Vi undersökte saken närmare, och kom till den slutsatsen att omit-pages inte borde påverka Funky Caches funktion. Då insåg vi att på testsidan hade vi en äldre version av WolfCMS, och den slutliga webbplatsen var uppbyggd på en nyare version. Vi misstänkte att det var orsaken till problemet, eftersom vi med lite sökande på WolfCMS forum hittade användare som stött på liknande problem med andra moduler när de tagit i bruk en nyare version av WolfCMS.

Vi beslöt oss för att lämna bort Funky Cache i detta skede av arbetet, men ändå ha det som ett alternativ i framtiden, och följa med om det kommer en version som stöds av senaste versionen av WolfCMS.

5.9 Testkörning av sidan

För att säkerställa att webbplatsen fungerade som den skulle så bestämde vi oss för att testa den. Vi började med att själva testa att allting fungerade som det skulle, både funktionsmässigt och utseendemässigt, även om vi redan under arbetsprocessen testat att allting fungerar som det skall efter varje ändring. Det blev inte några stora överraskningar när vi gjorde våra egna tester, där vi bland annat ändrade på innehåll och skapade en ny sida, och granskade utseendet efter det. Problem som upptäcktes vid våra test beskriver vi närmare i ett senare kapitel.

Efter våra egna tester tog vi Kontoramas personal som testpersoner. Vi gav dem en snabb skolning i användningen av webbplatsen, och gav dem tillgång till användarmanualen vi skapat, och därefter bad vi dem utföra olika uppgifter för att se om allting gick smärtfritt och enkelt, och om användningen av sidan skedde så som vi hade tänkt oss. Detta arbetsskede ansåg vi vara viktigt, eftersom vi inte ville överlåta en produkt som på ytan ser bra ut, men som sedan har defekter.

6 Språkalternativ

Eftersom webbplatsen skall kunna användas av både svensk- och finskspråkiga så måste slutanvändaren ha möjlighet att välja vilket av språken som skall visas. WolfCMS har en modul för detta ändamål, men den valde vi att inte använda oss av på grund av flera orsaker.

En av de största orsakerna varför vi valde att inte använda oss av den inbyggda språkmodulen var att den inte påverkade menyn på något sätt, så även om man fick innehållet på sidan att fungera på flera språk, så var menyn alltid på det språk som den ursprungligen var skriven på, och vi hittade inget sätt att åtgärda detta på. En annan orsak att söka efter ett annat alternativ var att modulen verkade aningen halvfärdig och svåranvänd, vilket skulle innebära mera arbete för oss i utvecklingsskedet, och så ansåg vi att användarvänligheten för slutanvändaren inte skulle ha varit den bästa möjliga.

När vi började söka efter olika möjligheter att få sidan tvåspråkig så stötte vi på många diskussionstrådar var användare hade funderat på samma sak. Det fanns väldigt många förslag på hur problemet med tvåspråkigheten skulle lösas, så i det här skedet verkade det vara vår största utmaning att få sidan tvåspråkig på ett sätt som är lämpligt både för oss som utvecklare och även ur slutanvändarens synvinkel.

Vi funderade en tid på att vi måste utveckla en egen modul som skulle sköta tvåspråkigheten, vilket skulle kräva en massa tid och testas, och kanske inte ändå fungera riktigt som det är tänkt, så vi studerade några användares lösning på problemet närmare för att se vilket sätt som skulle vara det lämpligaste att gå efter.

Efter en hel del studerande av alternativ såg vi av en slump att när man skapade en sida i WolfCMS så gick det under fliken ”settings” att välja vilken layout som skulle användas just för den sidan. Då fick vi idén att skapa två olika layouts, en svensk och en finsk. Designen på båda layouterna skall vara identiska, men för att få det hela att fungera så var det några saker som vi måste ta i beaktande, och de största ändringarna hamnade vi göra med menyn, vilket vi kommer att gå in närmare på.

Valet att utnyttja layouts till att få sidan tvåspråkig är även bra med tanke på säkerheten och kommande uppdateringar till WolfCMS, eftersom layouts är en del av själva systemet, vilket betyder att sidan inte är beroende av tredjeparts moduler som skulle kräva egna uppdateringar i framtiden.

Vi blev verkligen överraskade över vårt fynd, och hur enkelt det gick att lösa tvåspråkigheten för sidan, eftersom detta alternativ inte hade nämnts i en enda diskussionstråd vi läst igenom, utan endast mer invecklade, svårtillämpade alternativ ansågs vara det bästa sättet att lösa det hela.

Eftersom vi gjorde detta val för att lösa tvåspråkigheten så måste slutanvändaren mata in svenska och finska data i två skilda sidor via administrationsgränssnittet. Detta är inte riktigt optimalt ur slutanvändarens synpunkt, men det var en kompromiss som vi valde att göra eftersom det inte gör inmatningen av nytt material märkbart svårare. Även informationen i reklamboxen som är skapad som snippets måste matas in skilt på svenska och finska, på liknande sätt som med sidor. Det är även möjligt att lägga till flera språk i framtiden om det skulle behövas.

6.1 Menyn

För att tvåspråkigheten skulle fungera ordentligt så var vi tvungna att ändra på koden till menyn. Vi var tvungna att göra ändringarna för att endast visa menyalternativ för det språket man har valt.

Vi funderade en god stund hur vi skulle förverkliga detta, och sedan kom vi på lösningen att använda oss av omit-pages. Detta är en egenskap i WolfCMS som möjliggör att man kan utesluta vissa sidor från att visas i menyn.

När man skapar en ny sida i WolfCMS får den automatiskt en egen ID. Genom att använda funktionen omit-pages och sedan i svenska layouten välja bort alla finska sidors ID:s så blir endast de svenska kvar, och endast menyalternativen till de svenska sidorna visas. Samma sak gjordes med den finska layouten, enda skillnaden är att där valdes alla svenska sidors ID:s bort.

För att detta skulle fungera om slutanvändaren vill skapa nya sidor så valde vi att på förhand reservera ID:s för svenska respektive finska sidorna. Vi ansåg att 10 sidor per språk skulle räcka till, så vi började med att skapa de svenska sidorna, och de fick ID mellan 1-10. Vi skapade innehåll på de sidor vi visste att skulle användas, och de som inte hade något innehåll i detta skede så valde vi att ange som ”hidden”, så att de inte kommer med i menyn.

Motsvarande sak gjordes för de finska sidorna, som fick ID mellan 11-20. De sidor som vi visste att skulle användas så skapade vi färdigt rubriker för, och resten angavs som hidden så de inte kom med i menyn.

När vi hade skapat sidorna klart så skrev vi in i finska och svenska layouten vilka sidor som skall väljas bort. För finska layoutens del blev det sidorna 1-10 som inte skall visas, och när svenska layouten är vald så visas inte sidorna 11-20. Kodexempel 8 på nästa sida visar hur koden för omit-pages användes.

Kodexempel 8: Funktion för att välja bort sidorna 11-20

```
<?php

$omit_pages = array(11,12,13,14,15,16,17,18,19,20);

if (in_array($menu->id, $omit_pages)) {

    continue;

}
```

För att få språkalternativet att fungera så behövdes ännu en länk till layouten som innehåller det andra språket. Detta löstes genom att laga två flaggor, en finsk och en svensk, som placerades i top-elementets högra kant. Endast en flagga visas åt gången, vilket betyder att om man är på svenska sidan visas finska flaggan, och tvärtom. För att länka till det andra språket så hade vi ett par olika alternativ, vi kunde skapa en enkel länk i HTML som leder till andra språkets första sida, och det andra alternativet var att med PHP-kod laga en kodsträng som automatiskt hittar över till andra språkets layout, och till den sidan som man för tillfället är på.

Slutligen så valde vi lösningen med en enkel HTML-kodsträng som länkar över till andra språkets layout, och till första sidan. Detta val gjorde vi för att det var enklast att förverkliga och PHP-funktionen vi funderade inte hämtar något extra värde till webbplatsen eftersom sidans besökare antagligen inte hoppar mellan språken, utan direkt väljer det språk som sidan skall visas på.

7 Uppdatering och flytt av webbplatsen

I det skedet när webbplatsen började vara färdigutvecklad så bestämde vi oss för att skapa en ny webbplats på ett annat webbhotell, som vi flyttar över all information till. Denna lösning kom vi till för att undvika möjliga problem vid uppdatering till den senaste versionen av WolfCMS, samt för att samtidigt slippa onödiga filer och mappar som kunnat uppstå under arbetets lopp. Den första versionen av webbplatsen skrotades inte helt, utan användes för test av moduler och olika lösningar som vi var osäkra på om vi skulle ha i den slutliga versionen, eller som vi inte genast ville implementera till den slutliga versionen av webbplatsen, utan ville testa först om de uppfyllde våra krav och önskemål och fungerade som de skulle.

Denna lösning visade sig ha både för- och nackdelar. En fördel var, som det var tänkt, att man kunde pröva lösningar på testsidan före de implementerades till den slutliga sidan. En nackdel som vi däremot inte tänkt på var modulernas kompabilitet mellan de olika WolfCMS-versionerna, vilket vi insåg när vi skulle implementera Funky Cache till den slutliga versionen.

Flytten till det slutliga webbhotellet kommer att ske i ett senare skede, utanför detta arbetes tidsramar. Denna lösning gjordes för att det befintliga innehållet på kontorama.fi är föråldrat och kan inte användas till den nya sidan. Detta innebär att nytt innehåll måste skapas av beställaren och personalen på Kontorama, och det har inte varit möjligt att göra ännu. En del innehåll har vi skapat för att få en liten bild av hur den färdiga sidan kommer att se ut, men det materialet kommer troligtvis inte att användas till den slutliga produkten.

Vi har dock avtalat med beställaren att utföra flytten till det slutliga webbhotellet när innehållet är skapat och inmatat på webbplatsen, vilket är planerat att utföra sommaren 2014, cirka två månader efter att detta arbete gjordes.

7.1 Vad krävs av servern

För att webbplatsen vi skapat skall fungera smärtfritt på nya webbhotellet så krävs det vissa egenskaper av servern.

- PHP version 5.1 eller högre
- MySQL version 4.1.x eller högre. Även Sqlite3 och postgreSQL stöds
- Stöd för PHP Data Objects (PDO)
- Stöd för Apache, alternativt IIS, Nginx, Hiawata eller Lighttpd

Den befintliga sidan ligger på en lokal leverantörs webbhotell. Vi kunde inte hitta någon information ifall deras servrar har stöd för det som WolfCMS kräver, så vi beslöt oss för att lämna detta möjliga problem till framtiden, eftersom webbplatsen inte flyttas till det slutliga webbhotellet inom detta arbetes tidsram. Om företaget inte erbjuder det som WolfCMS kräver så är enda alternativet att byta leverantör av webbhotell, vilket inte är ett stort bekymmer.

8 Statistik

För att göra det mer intressant för personalen att upprätthålla sidan och veta att den verkligen har besökare så bestämde vi oss för att implementera statistikfunktioner för att hålla koll på hur många träffar sidan får, och vilka vägar som användare kommer in på Kontoramas webbplats.

Programmet som vi beslöt att använda för statistikfunktionerna heter Piwik. Detta val gjorde vi eftersom det var ett bekant program från tidigare, det är gratis och man får väldigt mycket information ut av det.

Ett annat starkt alternativ var Google Analytics, men vi valde Piwik istället främst på grund av hur statistikuppgifter som samlas in behandlas; Piwik påstår att de inte delar uppgifter över sina användare vidare, medan vi är lite skeptiska över att data som Google Analytics samlar inte skulle skickas vidare till tredje part. Andra orsaker var att vi inte ville vara beroende av externa tjänster, och som redan tidigare nämndes, var Piwik bekant från förut.

8.1 Piwik

Piwik är ett mångsidigt open source webbanalysverktyg. Det erbjuder obegränsat lagringsutrymme, möjlighet till flera olika språk, medräknat finska och svenska, och så hävdas det att installationen av Piwik är enkel att genomföra. (Piwik, u.å.)

Med Piwik får man reda på vilken väg användaren har kommit till webbplatsen som man har statistikfunktionen installerad på, hur mycket besökare webbplatsen har, vilka sidor som är mest populära och hur ofta användaren besöker webbplatsen. Piwik har också en mobil applikation vilket möjliggör att man var som helst, när som helst kan hålla koll på statistiken för webbplatsen. Utöver detta så lovar Piwik att användares data är viktig för dem, och att de inte ger vidare statistikuppgifter över sina användare.(piwik, u.å.)

8.2 Implementering

Piwik installationspaketet kan hämtas från deras webbsida. Efter detta skapas en ny mapp i root-mappen på servern var webbsidan finns, och den namnges Piwik, var filerna placeras. Efter det går man via webbläsaren till sidans URL, och efter URLen skrivs /piwik/. Då kommer man till Piwiks installationsgränssnitt, var man skall mata in databasens namn, användarnamn och lösenord. Eftersom Piwik använder samma databas som webbsidan, så måste Piwik ha en egen ”table prefix”, vilken anges till ”piwik_”. Efter detta skapas man ett konto till Piwik, och efter detta skall ännu URL till webbsidan skrivas in. Därefter genereras en JavaScript-kod som skall placeras i layoutens HTML-kod. Därefter är webbsidan klar att analyseras.

Ytterligare en inställning som gjordes var att inte respektera användares webbläsarinställning Do Not Track (DNT). Detta är en inställning som berättar om användaren får bli spårad, som är aktiverad som standard i vissa webbläsaren. Om vi inte skulle ha ställt in detta skulle statistiken bli missvisande, eftersom de som använder webbläsare med denna inställning aktiverad som standard inte skulle visas i statistiken.

9 Användaren

För att underlätta administrationen och upprätthållningen av webbplatsen så skapade vi material i form av användarmanual, grafisk profil, färdiga användarkonton och installerade moduler som underlättar skapande och editerande av information. Dessa punkter går vi igenom i detta kapitel närmare.

9.1 Användarkonton

Så att personer som enbart skall ändra på innehåll inte slipper åt mer kritiska inställningar som har med sidans funktionalitet att göra så valde vi att skapa egna användarkonton åt alla som behöver tillgång till sidan, och begränsa rättigheterna på dessa.

Till en början tänkte vi att vi skulle installera en modul som håller logg på vad användarna gör, för att vid eventuella fel gjorda av användaren skulle kunna upptäckas och man kunde berätta åt användaren som gjort fel om detta, och undvika samma fel i fortsättningen. Denna idé slopades i slutskedet av projektet, dels på grund av att vi insåg att det inte var nödvändigt att ha denna funktion eftersom det är så några personer som kommer att upprätthålla sidan. En annan orsak var att vi ansåg att modulen som det var meningen att använda inte var så bra som vi tänkte oss, och den hade inte upprätthållits på en tid.

För att skapa en ny användare så måste man ha administratörrättigheter. Sedan går man via administratörgränssnittet till fliken användare. Här kan man hantera befintliga användare och skapa nya.

Vi beslöt oss att skapa ett konto som personalen på Kontorama skulle använda sig av för att editera innehållet på sidan. Vi beslöt oss att ge kontot mycket begränsade rättigheter, och valde kontotypen editor. Som editor har användaren inte tillgång till användare-fliken, och andra kritiska inställningar i administratörgränssnittet.

Vi märkte vid test av personalkontot att vi var tvungna att ge mera rättigheter för kontot, eftersom även snippets gömdes med det mest begränsade kontot, vilket skulle förhindra personalen att ändra på innehållet i reklamboxarna. När vi ändrade rättigheterna till developer så var allting tillgängligt förutom användare-

fliken. Detta betyder att användaren med developer-rättigheter har alla rättigheter som administratören har, förutom skapandet av nya användare. Detta ansåg vi till en början att var ett stort problem, men vi beslöt oss att i skolningen ta upp vilka delar av administrationsgränsnittet skall lämnas i fred. Även i användarmanualen som vi skapar tar vi upp vilka flikar som skall lämnas i fred, och vilka som skall användas vid upprätthållning av sidan.

9.2 Skapande av användarmanual

För att vem som helst skall kunna använda webbplatsen utan tidigare datorkunskaper så skapas det en användarmanual som berättar hur man utför enkla åtgärder, som hur man lägger till och byter innehåll i reklamboxarna, hur man uppdaterar innehåll med mera.

Vi började redan i ett ganska tidigt skede att skapa användarmanual för lätta åtgärder som att ändra innehåll och ladda upp bilder till sidan. Vi märkte dock efter en tid att olika lösningar som vi gjorde kunde påverka hur slutanvändaren skall upprätthålla sidan, så vi bestämde att vi lägger skapandet av användarmanualen på is tills vidare, och fortsätter med den när vi har allting klart, och ändringar inte mera sker.

När sidan var så gott som klar började vi med skapandet av användarmanualen pånytt. Vi funderade på förhand ut några punkter som vi ansåg att kommer att användas mest, och utifrån dessa punkter började vi skapa manualen.

De punkter vi valde att ta med var följande:

- Ladda upp bilder
- Lägga till en ny sida
- Uppdatera befintliga sidor och reklambox
- Allmänna redigeringsverktyg

Dessa punkter ansåg vi att var de som kommer att användas, så det var lätt att välja vad vi skulle skriva närmare om. Basfunktioner, som hur man kommer till administratörgränssnittet och hur man använder det, beskriver vi också i korthet eftersom det krävs för att kunna editera och skapa innehåll till webbplatsen.

När vi hade användarmanualen klar, var webbplatsen också i sådant skick att man kunde göra ett prov, vilket i detta fall var att ge användarmanualen till personalen på Kontorama och låta dem lägga till och byta lite innehåll på webbplatsen.

Genom detta test säkerställde vi oss om att användarmanualen är skriven på ett sätt som är lättförståeligt för en person som inte har så stor teknisk erfarenhet, och på samma gång ser vi att funktionerna på sidan fungerar som de skall. Vid testningsprocessen uppstod inga problem, så vidare åtgärder på användarmanualen behövde vi inte göra i detta skede.

9.3 Ckeditor

WolfCMS hade några texteditorer färdigt implementerade som vi tänkte använda oss av, men märkte att dessa inte var riktigt bra. Det var en hel del små problem som uppstod vid formatering av text, så vi sökte en annan texteditor för att underlätta formateringen av text och bilder som matas in på webbplatsen.

Efter att vi sökt igenom olika moduler för WolfCMS hittade vi en texteditor vid namnet CKeditor. Vi beslöt oss att ge denna en chans, och laddade ner den. Det visade sig att installationen av CKeditor var enkel, och vid de första testerna verkade CKeditor fungera som det var tänkt, så den blev vår slutliga texteditor.

9.4 Grafisk profil

Eftersom Kontorama inte tidigare haft någon grafisk profil att gå efter så beslöt vi oss att skapa en sådan i samband med utvecklingen av webbplatsen. Vår första tanke var att skapa profilen och utgå från den vid skapandet av webbplatsen, men sen fick idén att vi skapar webbplatsen först och sedan skapa den grafiska manualen utgående från lösningarna vi använt oss av i designen.

Vi utgick från den befintliga logon som Kontorama har, eftersom den inte skall ändras på. För att få fram vilka färger som används i logon så öppnade vi den i ett bildbehandlingsprogram, och på detta sätt fick vi fram exakta färger i HEX-format.

Eftersom Kontorama inte var i behov av en så väldigt djupgående grafisk manual så bestämde vi oss för att endast ta med det allra nödvändigaste, och förklara

allting på ett så lätt sätt som möjligt, vilket vi hoppas att leder till att manualen används.

De saker vi tänker ta fram i den grafiska manualen är hur logon skall placeras, och vilken logotyp som skall användas i olika sammanhang, exakta färgerna som skall användas, vilka fonter som skall användas, fontstorlek med mera för att få det hela mer enhetligt vid reklam- och informationsutskick.

Vi skapade även i samband med den grafiska profilen några olika versioner av logon, som passar för olika behov.

10 Kontaktsida

På kontaktsidan skall det förutom standard kontaktinformation i textformat finnas möjlighet att via en karttjänst snabbt se var i Ekenäs Kontorama finns, men också skicka e-post till Kontorama direkt via webbplatsen, utan att behöva logga in på sin mail och kopiera e-post adressen från webbplatsen. Denna lösning skall göra steget att ta kontakt med företaget mindre, och förhoppningsvis få kunder som annars inte skulle ha tagit steget för att skicka e-post att göra det.

10.1 Kontaktformulär

För att besökare på webbplatsens skulle ha möjligheten att enkelt ta kontakt med Kontorama så valde vi att implementera ett kontaktformulär under menyn kontakt. För att kunna skicka e-post krävs att användaren fyller i sin egen adress som man kan svara till, samt eget namn.

10.2 Google maps

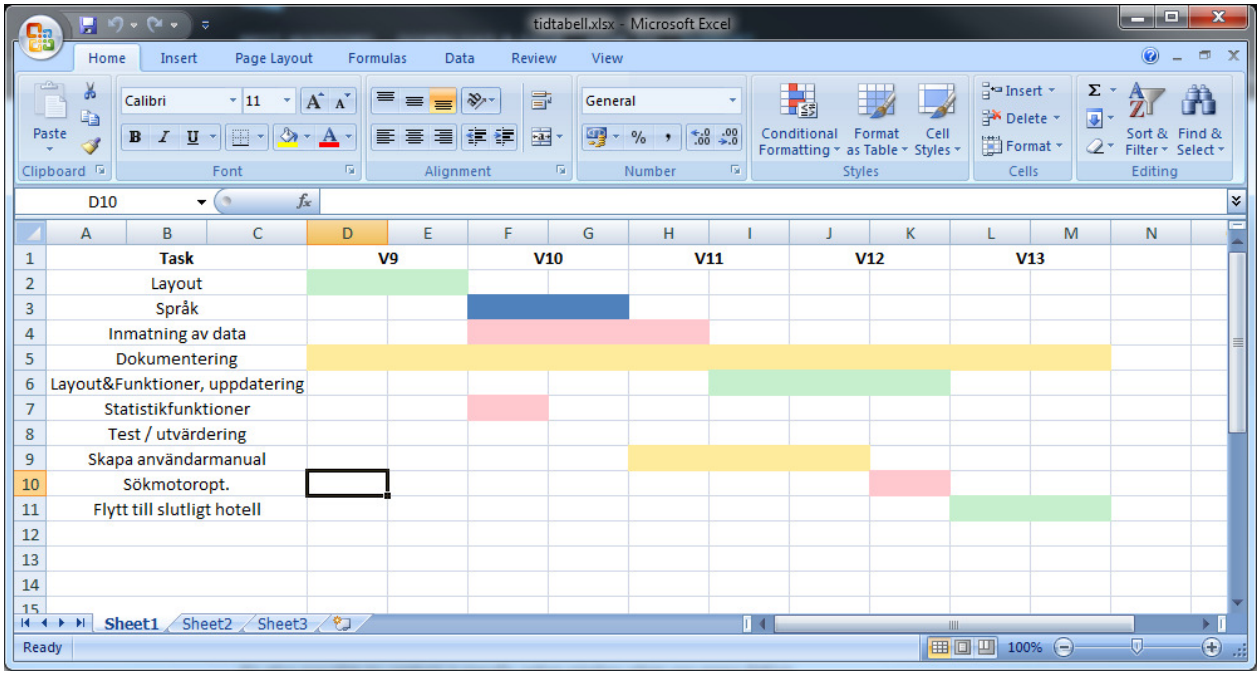
För att få mera innehåll samt göra det lättare för webbplatsens besökare att hitta till Kontorama så valde vi att använda oss av Googles karttjänst. Detta görs genom att söka upp gatuadressen på tjänsten maps.google.com, och sedan genererar tjänsten en iFrame-kodsträng som innehåller all information, inklusive utseendet

och storleken som behövs för att implementera funktionen på webbplatsen. Denna kodsträng läggs sedan in på önskad sida på webbplatsen, vilket i detta fall blev på kontaktsidan i vänstra kolumnen. För att få kartan att passa in kartan så justerade vi storleken en aning, och tog bort texten under kartfönstret som var en länk till Google Maps. Detta gjorde vi eftersom länken också finns med i själva kartfönstret.

11 Skapande av tidtabell

I slutskedet av projektet så började tidsgränsen för när vi måste vara klara med slutarbetet komma emot, så vi beslutade oss att skapa ett enkelt tidsschema och klart dela upp när olika skeden av webbplatsen skall göras, och under vilken tidsperiod. Detta underlättade för oss att se vad vi hade kvar att göra, vad vi skall göra en viss vecka, och så såg vi också om vi hade en arbetstakt som räckte till för att få arbetet klart inom den tid vi hade reserverat för slutförandet av arbetet.

Tidsschemat lagade vi i Excel, och för att få det överskådligt och tydligt så lade vi tidslinjen horisontellt och olika tasks fick vara lodrätt. För att markera vad som skulle göras under vilken tidsperiod så färgade vi det antalet celler som vi ansåg vara passligt, och hade olika färger för olika tasks för att de inte skulle smälta in i varandra, se figur 5 på nästa sida.



Figur 5: Tidtabell för slutförande av arbetet

Uppföljningen av tidtabellen gick bra, även om vi först var skeptiska till om det var värt att sätta ner tid på att skapa tidtabellen, eftersom vi tyckte att vi hade bra koll på läget. Det visade sig ändå att det var ett bra val att skapa tabellen, eftersom det var mycket enklare att med en enda överblick se vad som skulle göras nu och även de kommande veckorna.

Efter en tids arbetande mot tidsschemat så märkte vi att vi varit aningen optimistiska angående hur snabbt olika uppgifter skulle vara klara, så efter några veckor var vi en aning efter i tidtabellen. För att inte helt lämna bort tidtabellen och för att hålla arbetsprocessen strukturerad så uppdaterade vi tidtabellen i detta skede när den inte höll, för att igen kunna följa den. I samband med detta lade vi även till några uppgifter som vi inte tänkt på vid skapande av första versionen av tidtabell, och lade till några veckor extra arbetstid för att hålla målet realistiskt.

När webbplatsen började vara klar så märkte vi igen att vi var efter i tidtabellen. Vi konstaterade då att det var på grund av många mindre arbetsmoment och detaljer med webbplatsen som skulle utföras som vi inte räknat med, och så hade vi underskattat tidsbehovet för olika uppgifter. I detta skede bestämde vi oss för att låta tidtabellen vara som den är, och så snabbt som möjligt få allting klart. Detta val gjorde vi eftersom vi ansåg att det inte fanns behov av en tidtabell för att hålla koll på det lilla vi hade kvar att göra, och det skulle inte vara tidseffektivt att uppdatera och upprätthålla tidtabellen i slutskedet av arbetet.

12 Möte med beställaren

Vi hade under arbetets gång flera möten med personalen på Kontorama för att säkerställa att arbetet var på väg mot rätt håll. Första mötet hade vi när layouten var i ett tidigt skede. Då fick vi positiv respons så vi fortsatte arbeta på layouten vi tagit fram. När arbetet framskred och det gjordes ändringar i layout och placering av olika element på sidan så frågade vi med jämna mellanrum om allting såg bra ut, och personalen hade ingenting emot ändringarna vi gjorde.

Mot slutskedet av arbetet så hade vi lite flera frågor att ställa, så vi stämde träff med beställaren. De frågorna vi hade gällde främst layout, men även om det fanns behov av undermenyer, och så hade vi några förslag som kunde implementeras, såsom ett formulär som kan användas för att skicka e-post till Kontorama. I detta skede berättade beställaren att han tyckte bilden under menyn var lite för stor, och slutanvändaren kanske inte alltid förstår att det finns text nedanför den. Vi funderade på en lösning på detta, men kom fram till att slutanvändaren med största sannolikhet förstår att det finns innehåll längre ner på sidan, och att datorn som vi visade sidan på hade lägre upplösning än dagens moderna datorer, så det var lite missvisande. Vi tog även upp detta med beställaren och framförde vår syn på saken, och han godkände vår synpunkt, så bilden ändrades inte på.

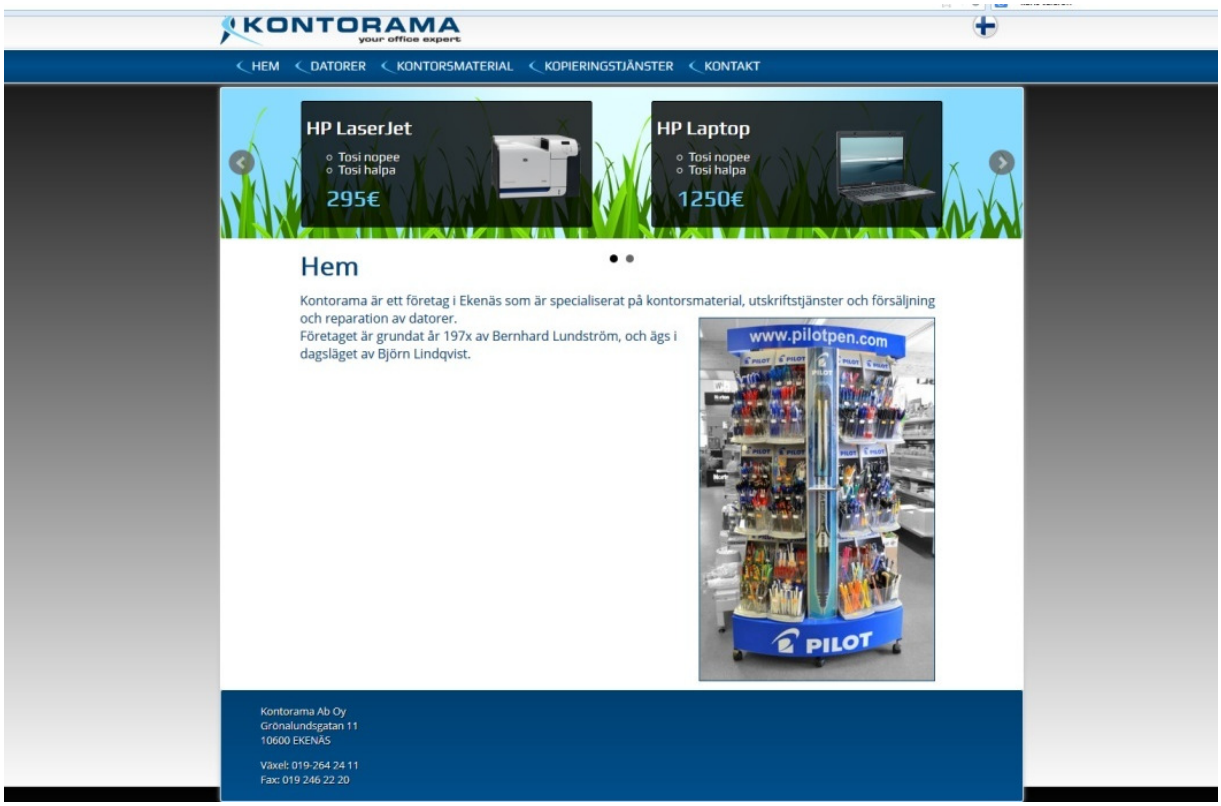
13 Problem och buggar

Under arbetsprocessen så stötte vi på en hel del problem och buggar med sidan. Stor del löste vi snabbt, men de som krävde lite mera arbetsinsats för att lösa går vi närmare in på i detta kapitel, och berättar hur vi gjorde för att lösa eller gå runt problemen.

13.1 Bakgrunden

Vi märkte att bakgrunden som var en gradient som gick från en mörkare grå till ljusare hade en vit balk längre ner på sidan. Vid närmare granskning så märkte vi att det inte endast var en ljus balk, det var flera balkar med olika gråa nyanser.

Detta problem var svårupptäckt, och syntes endast på en skärm med bra färgåtergivning. För att tydligare få fram problemet ändrade vi bakgrundsfärgen till en mörkare variant så att felet tydligare skulle ses, se figur 6.



Figur 6: Bild som visar balken i nedre kant av sidan

För att lösa problemet så försökte vi först med: height:100 %. Detta löste problemet delvis, men inte helt: på en skärm med högre resolution så visades bakgrunden på rätt sätt, men redan med full-hd resolution så kom balken tillbaka. Om man förminskade sidan ett klick med en full-hd skärm så försvann problemet. Vi märkte även att på en skärm med högre resolution var felet inte uppstod, så genom att förstora sidan en aning så uppstod problemet igen.

Nu när vi tagit reda på att problemet var beroende av sidans höjd, eftersom problemet endast uppstår när höjden blir större än div-elementet container, så försökte vi ännu med att lösa problemet. Det kändes en stund hopplöst när en enkel funktion kunde ställa till med så stora problem.

Vi antar att problemet är i CSS-standarden, närmare bestämt i funktionen linear-gradient, vilken vi använt oss av för att skapa bakgrunden. De alternativ vi hade i detta skede var att lägga en vanlig, grå bakgrund, laga en gradient bakgrund med hjälp av en bild eller försöka lösa problemet med linear-gradient i CSS.

Det prövades många olika lösningar, men slutligen så löstes problemet genom att ge CSS-egenskapen o-linear en fast höjd på 300 pixlar, och utanför detta område ha en enfärgad bakgrund. Den enfärgade bakgrunden skapades med hjälp av CSS-selektorn html, var bakgrundsfärgen angavs till grå, #eee.

13.2 Skyltfönstret

Vi upptäckte att skyltfönstret inte fungerade som det skulle. Problemet var att skyltfönstret var för långt mot höger, över containerns gränser. Detta problem uppstod endast när man använde webbläsaren Opera, i andra webbläsare vi prövade sidans funktionalitet med (Internet Explorer, Google Chrome och Firefox) fungerade allting som det var tänkt.

För att lösa problemet började vi med att studera koden närmare för att se vad som möjligtvis kunde orsaka problemet. Vi hade angett bredden för skyltfönstret med CSS-funktionen width: calc (100% - 2px); som är en CSS3-standard som möjligtvis inte webbläsaren Opera stöder, eftersom calc är en relativt ny funktion.

Lösningen på problemet blev enklare än vi trodde. Opera som vi hade installerat på datorerna var version 12, vilket visade sig vara en utgående version. Detta var skönt eftersom det inte var något direkt fel på vår kod, men samtidigt lite frustrerande eftersom webbläsaren Opera inte på något sätt meddelat att det finns en ny version att ladda ner, även om vi sökt efter uppdateringar till version 12. Detta orsakade att vi ändrade koden i onödan flera gånger när vi felsökte, och gick miste om en hel del tid som vi kunde använt till andra arbetsskeden.

När vi gick över till den nyaste versionen så var problemet borta, och vi beslöt oss för att inte sätta tid på att få det att fungera i Opera 12, eftersom det ändå var en utgående version, och det endast var fråga om månader innan den skulle ersättas helt av Opera 20.

13.3 Kolumner

Vi upptäckte att kolumnerna som används för kontaktuppgifterna inte var lika breda, även om det var meningen att det skulle vara så. Detta var ett problem som man inte upptäckte ifall man inte visste hur stora kolumnerna egentligen skulle vara.

För att lösa detta problem använde vi oss av Responsive Grid System, som även används i sidfoten.

14 Sammanfattning

När webbplatsen är klar så ser vi tillbaka på arbetet vi gjort och känner oss nöjda och lättade att vi fått allting klart. Kunden är nöjd med webbplatsens utseende och dess funktionalitet, och vi har uppnått de mål som vi tillsammans med kunden ställde.

Under arbetets gång så har vi försökt att ha träff med handledaren ungefär varannan vecka, och strävat efter att vid varje möte ha lite mera av arbetet skrivet, och mera på webbplatsen gjort varje gång det är möte. Dessa träffar har varit till stor hjälp, både för att få en annan persons åsikt om hur arbetet framskrider, och så har det varit lite press på att ha någonting nytt att visa upp varje gång. Även knuffar i rätt riktning och utvecklingsidéer har vi fått av handledaren, både vad gäller skrivandet och det praktiska arbetet.

Arbetsprocessen har inte varit en rak väg från början till slut. Det har varit många lösningar som vi till en början har tänkt att vi skall använda och de har verkat bra, men som sedan slopats av olika skäl. Även lösningar vi haft i tankestadiet har bytts ut mot lösningar som vi funnit lättare att förverkliga. Även valet av ett lite mindre CMS-system har haft både sina för- och nackdelar. Till nackdelarna kan räknas relativt liten användargrupp, så det var svårt att få hjälp via internet när det uppstod problem relaterade till WolfCMS, och en annan nackdel har varit att några moduler som verkat lovande och som vi tänkt använda oss av inte funnits längre, eller så har de varit föråldrade och ingen upprätthåller dem mer, så de har uteslutits av den orsaken.

En sak som vi märkte att tog onödig tid och skapade förvirring var kontroll av lösenord och användarnamn till de olika tjänsterna. Detta berodde på att det var många tjänster som det skulle skapas användarkonton till, och det blev i början lite blandning angående vilka lösenord som hörde till vilka tjänster, även om vi skapat lösenord som var lätta att komma ihåg, och vi var två som visste dem. Detta fick vi snabbt ordning på genom att ha en textfil var vi skrev ner vilka lösenord som hörde till vilken tjänst.

Om det är någonting som vi skulle ha gjort annorlunda, så skulle det definitivt vara att så snabbt som möjligt börja på slutarbetet och få det klart, och inte lämna det till sista minut som det blev i vårt fall. Även när vi skapat en tidtabell för slutförandet av arbetet så har vi ofta underskattat hur lång tid olika uppgifter tar att slutföra, och så har vi inte tänkt på många mindre uppgifter som måste göras. Detta ledde till att vi i slutskedet av arbetet fick bråttom att ha allting klart, när det var en hel del smått kvar att göra som vi inte räknat med, och som vi räknat att skulle vara klart på kort tid.

En annan sak som vi funderade på under arbetets gång och efteråt är hur webbplatsen skulle ha blivit om vi använt oss av Drupal som vi först tänkte. Det skulle ha varit enklare att hitta moduler som direkt skulle ha passat och som upprätthålls aktivt, och även lösning på olika problem som uppstår skulle troligtvis ha hittats snabbare och enklare jämfört med WolfCMS. Vi är ändå nöjda med resultatet och valet av WolfCMS, eftersom det resulterade i en webbplats som är visuellt tilltalande, har de funktioner som beställaren önskade sig, och ändå är koden lättförståelig och logiskt uppbyggd. Kodmässigt skulle det antagligen ha blivit lite mera omfattande med Drupal, samt att sidan skulle ha varit mer tungarbetad.

Att utföra arbetet och skapa dokumentation i form av detta slutarbete har varit en process som krävt mera tid och arbete än vi först hade tänkt oss, men i slutändan har det varit en intressant och lärorik upplevelse.

Källförteckning

WordPresskurser (u.å.) Vad är ett CMS? [Online]
http://www.wordpresskurser.se/tips/vad-ar-ett-cms/ [Hämtat 22.4.2014]

Drupal (u.å.), About Drupal. [Online]
https://drupal.org/about [Hämtat: 17.11.2013]

Docs.joomla, 2013, Evaluators [Online]
http://docs.joomla.org/Evaluators [hämtat 17.11.2013]

Joomla.org (u.å.), What is Joomla? [Online]
http://www.joomla.org/about-joomla.html [Hämtat 17.11.2013}

Dropbox.com,(u.å.) About Dropbox [Online]
www.dropbox.com/about [Hämtat 10.2.2014]

Arkku.net (u.å.), Tervetuloa Arkkuun [Online]
http://www.arkku.net [Hämtat 10.2.2014]

Notepad (u.å.), About [Online]
http://notepad-plus-plus.org/ [Hämtat 10.1.2014]

BxSlider (u.å.), The Responsive jQuery Content Slider [Online]
http://bxslider.com/ [Hämtat 12.2.2014]

Jquery (u.å.), What is jQuery? [Online]
http://jquery.com/ [Hämtat 12.2.2014]

Wikipedia.org 2013, Uniform Resource Locator [Online]
http://sv.wikipedia.org/wiki/Uniform_Resource_Locator [Hämtat 2.4.2014]

Wikipedia.org 2014, .htaccess [Online]
http://en.wikipedia.org/wiki/.htaccess [Hämtat 2.4.2014]

Apache.org (u.å.), Apache mod_rewrite [Online]
http://httpd.apache.org/docs/2.2/rewrite/ [Hämtat 2.4.2014]

Wikipedia.org 2014, Sökmotoroptimering [Online]
http://sv.wikipedia.org/wiki/Sökmotoroptimering [Hämtat 23.3.2014]

WolfCMS 2014, Funky Cache [Online]
http://www.wolfcms.org/repository/38 [Hämtat 18.3.2014]

Piwik (u.å.), What is Piwik? [Online]
http://piwik.org/what-is-piwik/ [Hämtat 7.4.2014]

Piwik (u.å.), Liberating Web Analytics [Online]
http://piwik.org/ [Hämtat 10.3.2014]

Pingdom (u.å.), Pingdom Website Speed Test [Online]
http://tools.pingdom.com/fpt/ [Hämtat 5.3.2014]

Normalize (u.å.) Normalize.css [Online]
http://necolas.github.io/normalize.css/ [Hämtat 13.4.2014]

Responsive Grid System (u.å.), Responsive Grid System [Online]
http://www.responsivegridsystem.com/ [Hämtat 15.4.2014]

Google fonts (u.å.) [Online]
https://www.google.com/fonts [Hämtat 16.4.2014]

Internetworld.idg.se (20.4.2012) Så funkar responsiv webbdesign [Online]
http://internetworld.idg.se/2.1006/1.444545/sa-funkar-responsiv-webbdesign
[Hämtat 22.4.2014]

WordPress (u.å.) Requirements [Online]
http://wordpress.org/about/requirements/ [Hämtat 24.4.2014]

Ivens, K. 2003.*Fortare, smartare Internet. Göteborg, Pagina Förlags Ab*

Kodförteckning

Figurförteckning

Bilaga 1

Användarmanual för kontorama.fi

2014

Contents

Inledning

Denna användarmanual beskriver de steg man bör behärska för att lägga till, editera och ta bort material från webbplatsen kontorama.fi

Logga in

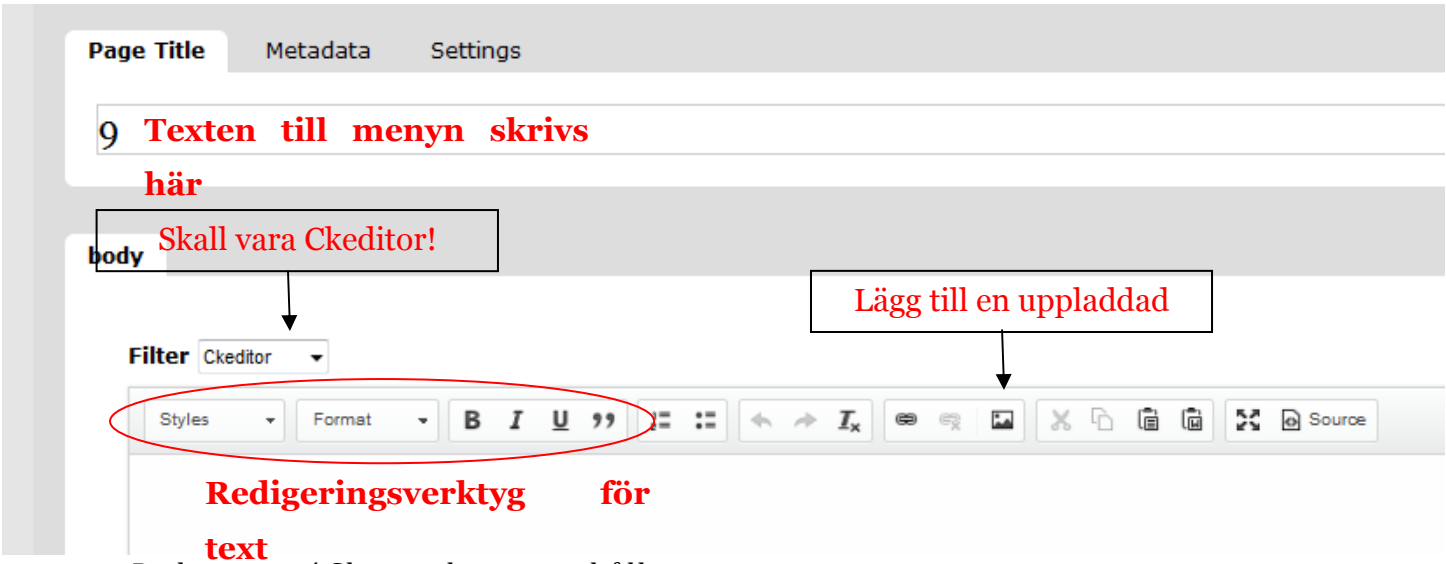
För att logga in til administrationsgränssnittet var man kan redigera sidans innehåll bör följande steg göras.

1. Gå till sidan www.kontorama.fi/admin via webbläsaren
2. Ange användarnamn och lösenord

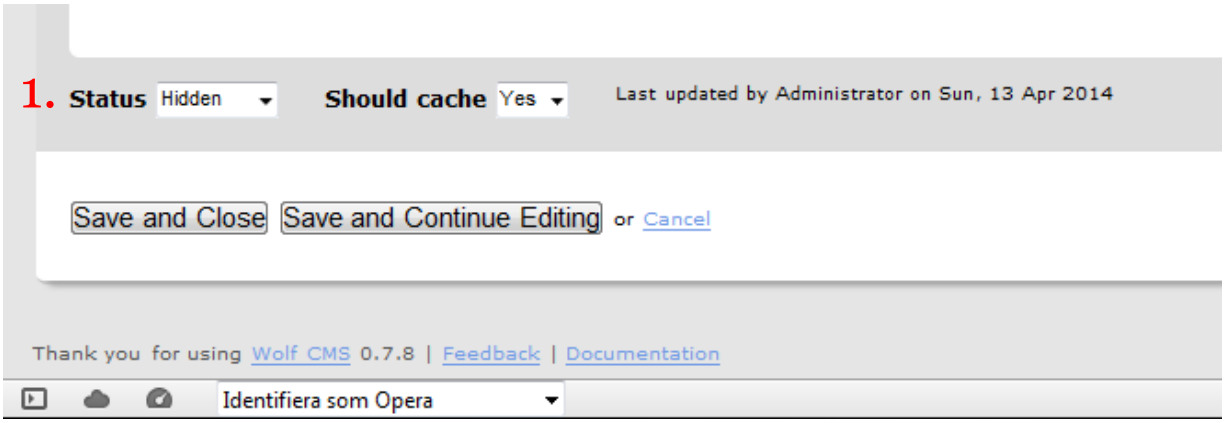
Efter dessa steg är du inloggad och kan redigera innehåll

Översikt

För att underlätta uppdatering och skapandet av nytt material på webbplatsen bör vissa grundegenskaper behärskas, se de två följande bilderna med förklaringar.



Redigering / Skapande av innehåll



1 Status

Det går att ha flera olika status på sidan, men endast två stycken är väsentliga, se nedan.

- Published – Sidan visas åt alla, skall endast användas på färdiga sidor
- Hidden – Sidan är gömd, visas inte. Om en sida är under uppbyggnad kan detta användas för att besökarna inte skall se en halvfärdig sida.

Ladda upp en produktbild

För att ladda upp en produktbild skall man gå till fliken ”files”. Bilderna placeras under mappen images, så man klickar på den mappen. Därefter väljer man ”upload file” till höger, väljer ”bläddra”, och därefter söker man upp bilden man vill använda och laddar upp den. När filen är uppladdad ser man en förhandsvisning på bilden, samt bildens namn till höger.

Skapa en ny sida

För att skapa en ny sida bör man göra följande:

1. Gå till fliken ”sidor”
2. För att skapa svenska sidan, välj ett av de lediga numren mellan 1-10. För finska sidorna gäller numrorna 11-20.
3. Tryck på numret du valt, och innehållseditorn kommer fram. Se till att filter är inställt på CKEditor.

4. När innehållet är klart, välj ”publish” istället för ”hidden” nere till vänster.
5. Tryck på ”Save and close” och innehållet är skapat.

Redigera sidor

För att redigera en sida gör man följande steg:

1. Gå till fliken ”sidor”
2. Välj den sidan som skall redigeras. Kom ihåg att även tillämpa ändringarna på fliken som är på det andra språket!
3. Redigera innehållet
4. Tryck på ”save and close” när det är klart

Redigera innehåll i reklambox

1. Gå till fliken ”snippets”
2. Här väljer man den reklamboxen man vill ändra på. Ändra även boxen som är på det andra språket.
3. Redigera innehållet på samma sätt som med sidorna. Lägg bilder först, texten efteråt.
4. Klicka ”save and close” så är det klart.