INGENJÖRSHÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

HEMSIDA FÖR ETT MUSIKBOLAG

Masoud Youssefi

Patrik Andersson

EXAMENSARBETE 2006

Kommunikation och Informationsteknik

INGENJÖRSHÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

HEMSIDA FÖR ETT MUSIKBOLAG

WEBSITE FOR A MUSIC COMPANY

Masoud Youssefi

Patrik Andersson

Detta examensarbete är utfört vid Ingenjörshögskolan i Jönköping inom ämnesområdet Datakommunikation och Informationsteknik. Arbetet är ett led i den treåriga högskoleingenjörsutbildningen. Författarna svarar själva för framförda åsikter, slutsatser och resultat.

Handledare: Magnus Schoultz

Omfattning: 10 poäng (C-nivå)

Datum:

Arkiveringsnummer:

Postadress:
Box 1026
551 11 Jönköping

Besöksadress:
Gjuterigatan 5

Telefon:
036-10 10 00 (vx)

Abstract

The purpose of this report is to answer the question:

How do you create a media player that can stream music for a website?

We started the work by searching the web and reading books about the subject and found several technologies to use to stream media files over the internet. We decided upon using the http protocol and created the media player itself in Macromedia Flash.

The website is mostly developed with PHP, JavaScript and CSS templates and the websites design was created with Macromedia Dreamweaver and Adobe Photoshop. We also created an administrative system for the media player with PHP programming.

The website contains a simple guestbook; the guestbook makes use of a database created with MySQL.

Our work resulted in a webpage with a fully functional media player that streams audio files together with a corresponding administrative system for it.

Sammanfattning

Nordic Steel är ett musik producerande företag som ägs av Jens Landegren och är beläget i Stockholm. Företaget tar främst emot beställningar från skivbolag men gör även egna produktioner vid sidan om, dessa egna produktioner vill företaget lansera i USA.

Frågeställningen för arbetet var:

Hur kan man skapa en media spelare som kan strömma musik för en hemsida.

Utifrån denna frågeställning började vi söka efter material för hur man kan skapa en media spelare som kan strömma musik över Internet. Vi hittade ett antal olika teknologier man kan använda sig utav och vi valde att använda oss av http protokollet. Själva media spelaren skapades i Macromedia Flash.

Vi utvecklade största delen av hemsidan med PHP programmering, JavaScript och CSS mallar. Designen på hemsidan utvecklades med hjälp av Macromedia Dreamweaver och Adobe Photoshop. Det skapades även ett administrativt system för media spelaren i PHP.

Hemsidan innehåller en enklare sorts gästbok som använder sig av en databas, som utvecklades i MySQL.

Vårt arbete resulterade i en hemsida med en helt fungerande media spelare som strömmar ljudfiler, tillsammans med ett lättanvänt administrativt system för media spelaren.

Nyckelord

PHP

SQL

Databas

Strömmande Medier

Innehållsförteckning

1 Figurförteckning

2 Inledning

2.1 Företagets bakgrund:

Nordic Steel är ett nyligen skapat enmansföretag som ägs av Jens Landegren, företaget har 2 anställda och är beläget i Stockholm. Företaget sysslar med musik produktion, de tar främst emot beställningar från skivbolag men gör även egna produktioner vid sidan om, dessa egna produktioner vill företaget lansera i USA. I dagsläget försöker företaget att lansera sig självt genom att skicka demonstrations skivor med sina produktioner till diverse musikbolag. Detta vill de effektivisera genom att samla produktionerna på en hemsida som de kan hänvisa till skivbolagen.

2.2 Vår bakgrund:

Vi håller på att slutföra vår ingenjörsutbildning inom datateknik med inriktning mot information och kommunikation. Under utbildningens gång har vi läst flera kurser inom ämnet användarvänlighet, vi har även läst kurser som behandlar områdena datanät, multimedia och webbdesign. De flesta kunskaper har använts för att konstruera en e-handelsplats från idéstadiet till en färdig produkt.

2.3 Problemformulering:

Det största problemet vårt företag har är att nå ut till en speciell marknad, specifikt den amerikanska musik marknaden med fokusering på de södra regionerna. För att lösa detta ville företaget ha en hemsida som ger reklam för dem. Deras huvudsakliga mål är att skapa en efterfråga på deras musik, för att uppnå detta måste de få företagets namn etablerat på marknaden. Själva produkten de vill få såld är deras musik och för att få folk intresserade av musiken måste företaget därmed lägga upp korta klipp från deras låtar. Detta skapar ett problem för företaget, eftersom folk då kan ladda ner deras låtar och det vill man ej.

Problem:

- Hur designar man en hemsida som intresserar just den marknaden, som dessutom är användarvänlig.
- Hur ska vi lyckas spela upp deras musik utan att den ska gå att ladda ner
- Vi måste göra en hemsida som de kan uppdatera själva
- Vi måste beakta användbarhetsaspekterna på sidan

2.4 Syfte och mål

Vårt mål är att göra en hemsida som ska göra reklam för företaget Nordic Steel i södra USA. På hemsidan ska man kunna höra klipp på musik som företaget producerat. De anställda på företaget har alla grundkunskaper i HTML kodning och Internet teknologi, så de kommer att sköta uppdatering av hemsidan själva. Vi måste därmed skapa en dynamisk sida som ska kunna gå att uppdatera med hänsyn till att de som ska sköta systemet bara har grundläggande kunskaper av HTML kodning. Syftet är att göra en hemsida som har en estetiskt tilltalande design men som fortfarande är användarvänlig, både för slutanvändarna samt för de anställda på företaget.

2.5 Avgränsningar

Från vårt första samtal med uppdragsgivaren så begärde dem att vi inte fick ha någon kostsam teknologi eftersom företaget arbetar efter en begränsad budget. Med kostsam teknologi menas t.ex. att de inte kan stå för en dedikerad Internet server och en dedikerad server för att strömma medierna som ska spelas upp på sidan. Företaget arbetar enbart i Windows miljö, så all programvara som används för att uppdatera hemsidan måste vara kompatibel med Windows programvaror.

2.6 Disposition

Rapporten följer Ingenjörhögskolans mall för examensarbeten och är uppdelad i fyra bitar.

Första biten går genom den teoretiska bakgrunden och ger en grundläggande beskrivning av de olika teknologier som använts under skapandet av sidan. Vi tar upp HTML, XML, PHP, kort om databas teorier, SQL, användarvänliga gränssnitt och strömmande medier över Internet.

Andra biten är själva genomförandet, där vi beskriver hur vi skapat sidan och löst problemen med att strömma medier på sidan. Vi beskriver även hur vi kommit fram till de beslut vi fattat.

Tredje biten presenteras det färdiga resultatet, detta främst med skärmdumpar av de skapade webbsidorna tillsammans med förklarande text.

Fjärde biten är slutsatsen och diskussionen. Där vi beskriver vad vi lärt oss av att göra detta examensarbete och även om hemsidans framtid.

3 Teoretisk bakgrund

3.1 Databas teori:

3.1.1 Allmänt om databaser:

En databas är en strukturerad förvaringsplats för information. För en databas att anses vara bra ska den vara organiserad på ett sådant sätt att man lätt kan hitta eller ändra informationen i den. Exempel på databaser kan t.ex. vara en telefonbok, där finns det lagrad information som t.ex. telefonnummer till myndigheter och privatpersoner.[1]

Det finns olika typer av databaser, relationsdatabaser, ”platta” databaser, hierarkiska databaser samt objektorienterade databaser. Vi ska här gå igenom platta databaser och relationsdatabaser, varav den senare är det som är mest aktuell i dagsläget. [2]

Den lagrade informationen i en databas kallas för en post, dessa poster kan sedan finnas under separata tabeller. T.ex. i telefonboksexemplet kan en tabell tillhöra privatpersoner, och ett annat företag. Under privatperson tabellen kan det finnas olika attribut som beskriver de så kallade posterna, t.ex. efternamn, förnamn, hemtelefon, mobilnummer. Det kan t.ex. se ut som på figur 3-1. En post här vore Andersson, Kalle, 012-345 67, 070-123 45 67. Och ett attribut vore Efternamn.

Efternamn	Förnamn	Hemtelefon	Mobilnummer
Andersson	Kalle	012-34567	070-1234567
Svensson	Erik	012-12345	070-678 90 12

Figur 3-1 Exempel på hur en tabell i en databas kan se ut

[1] http://www.pmdatautbildning.se/webbkurs/db/1DB.htm 2006-06-17 16:12
[2] Ibid.

3.1.2 Platta databaser:

Det finns även andra sorters databaser som inte är uppbyggda enligt tabeller. Den enklaste formen av en sådan databas är en så kallad platt databas, dessa databaser är egentligen en, oftast stor, fil där man sparar information i formen av en lista med en avgränsare mellan varje post. Man skulle kunna kalla platta databaser för ett register. En platt databas baserad på informationen i figur 3-1 skulle kunna se ut såhär

Efternamn, Förnamn, Hemtelefon, Mobilnummer | Andersson, Kalle, 012-34567, 070-1234567 | Svensson, Erik, 012-12345, 070-6789012

Sökning av dessa databaser är oftast tids konsumerande och dessa databaser blir oftast onödigt stora i filstorlek.[3]

3.1.3 Relationsdatabaser:

En relationsdatabas är en databas som är uppbyggd på relationer (även kallat tabeller), som t.ex. den i Figur 3-1. En relation (tabell) innehåller flera kolumner samt rader. Varje kolumn innehåller ett attribut, som t.ex. kundnummer eller efternamn, vissa av dessa attribut är identifierade attribut och resterande är rena egenskaper. Varje rad innehåller informationen som placeras under kolumnerna (även kallad dess värde). T.ex. i fallet ovan så innehåller kolumn 1 informationen ”Efternamn” och raderna under värden för attributet ”Efternamn”. I detta fall är värdet för rad 2 Andersson, vilket sedan faller under attributet ”Efternamn”. En tabell består som sagt av två olika sorters attribut, ett identifierande attribut som måste vara unikt (även kallat primär nyckeln), exemplet ovan har ej ett sådant, samt egenskapande attribut.[4] Nedan följer ett exempel på en tabell med ett identifierande attribut (Kundnumret) samt två egenskaper (Efternamn samt Förnamn).

Kundnummer	**Efternamn**	**Förnamn**
100	Andersson	Kalle
101	Svensson	Eric

Figur 3-2 En relationsdatabas med identifierande attribut

[3] http://dub.abm.uu.se/mus/mus_saml3.HTML 2006-06-17 16:38
[4] Ibid.

Att bygga en databas i tabeller gör det mycket lättare att snabbt söka och finna delar av informationen. Man skulle t.ex. kunna söka genom alla som heter Andersson i efternamn på ett mycket enklare sätt än med en platt databas. Som namnet anger innehåller relationsdatabaser så kallade relationer. En relation är en koppling mellan olika tabeller.[5] Det finns tre huvudtyper av relationer, vi har 1:1 relationer (kallat ett till ett relationer). 1:∞ (ett till många) och ∞:∞ (många till många).[6] Dessa olika typer av relationer beskriver hur många olika egenskaper de olika tabellerna har som relaterar till varandra.

I en 1:1 tabell har vi i tabell A ett värde A1 som har en relation till värdet B1 i tabellen B.

I en 1:∞ tabell har vi i tabell A ett värde A1 som har en relation till flera värden i tabellen B, t.ex. B1 B2 B3.

I en ∞:∞ tabell finns det flera värden i tabellerna A och B som har kopplingar, man vill helst undvika dessa ∞:∞ relationer eftersom en relationsdatabas ej kan hantera dessa relationer, utan man måste skapa ett hjälpobjekt, ett så kallat relationsobjekt, som hanterar dessa.

Som tidigare nämnt så består tabeller av antingen ett identifierande attribut eller egenskapande attribut. För att få en databas som är så bra som möjligt och ej har för mycket dubbellagrad information finns det tre så kallade normalformer man måste följa angående skapandet av dessa attribut.

[5] http://dub.abm.uu.se/mus/mus_saml4.HTML 2006-06-17 17:44
[6] http://www.pmdatautbildning.se/webbkurs/db/3DB.htm 2006-06-17 21:42

3.2 Normalformerna

3.2.1 Första normalformen (1NF)

Definition: En relation är på 1NF om dess termer är odelbara och uppträder endast en gång. [7]

I varje ruta i tabellen (d.v.s. varje attributs egenskap) får det bara finnas ett värde. Exempelvis om man har attributet telefon, så får man bara spara värdet för t.ex. hemtelefon. Man får ej spara värdena för hemtelefon, mobiltelefon och telefon till arbetet under samma attribut. Vill man ha med alla de tre olika telefonnumren och följa 1NF måste man skapa olika attribut för varje telefon. I detta exempel vore det då tre attribut som skulle kunna ha följande namn ”Hemtelefon”, ”Mobiltelefon” och ”Arbetstelefon”

3.2.2 Andra normalformen (2NF)

Definition: En relation är på 2NF om den är på 1NF och varje attribut beror på hela nyckeln. [8]

Det får med andra ord ej finnas vissa attribut som inte är beroende av relationens identifierare (nyckelattribut). T.ex. så följer inte följande exempel 2NF.

DELTAGARE: **Kursnr** **Personnr** Kursnamn Start

Förklaring, DELTAGARE är relationens (tabellens) namn, Kursnr Personnr är dess identifierande attribut (nyckelattribut) och Kursnamn och Start är dess egenskaper

Exemplet följer inte 2NF för kursen byter ej namn för varje deltagare på kursen. Därmed är identifieraren ”Personnr” inte direkt kopplat till ”Kursnamn”. För att denna relation ska följa 1NF och 2NF måste den delas upp i två olika relationer. T.ex. [9]

KURSER: **Kursnr** Kursnamn

DELTAGARE: **Kursnr** **Personnr** Start

Här är alla egenskaper beroende av de nyckelattribut som finns i relationen. T.ex. så är inte start lika för alla Kursnr eller Personnr.

[7] Apelkrans och Åblom, 2001, s.280
[8] Ibid. s.282
[9] Ibid.

3.2.3 Tredje normalformen (3NF)

Definition: En relation är på 3NF om den är på 2NF och ingen egenskap är transitivt beroende av nyckelbegreppet. [10]

Med 3NF försöker man undvika att en relation har en egenskap som även skulle kunna vara ett identifierande attribut. T.ex. [11]

FAKTURA: **Faktnr** Kundnr Kundnamn Medmera

I detta fall kan egenskapen Kundnr vara ett identifierande attribut eftersom Kundnamn går att härleda från denne. Därmed måste relationen delas upp i två olika relationer, som t.ex.

FAKTURA: **Faktnr** ↑Kundnr Medmera

Förklaring, med uppåtpilen (↑) menas att det finns en annan relation (tabell) med denna egenskap som identifierande attribut

KUND: **Kundnr** Kundnamn.

Denna uppdelning ser till att man slipper dubbellagra informationen ”Kundnamn” på varenda faktura. Nu behöver man bara lagra ”Kundnamn” på ett ”Kundnr” för varje kund.

[10] Apelkrans och Åblom, 2001, s.284
[11] Ibid

3.3 SQL (Structured Query Language)

3.3.1 Allmänt om SQL:

Under 1970-talet bedrev dataföretaget IBM ett relationsdatabas projekt, tanken med projektet var att ta fram ett system som tillät en att med enkla kommandon ta fram information från en databas. Utifrån detta projekt växte SQL fram. Målsättningen med SQL var att ta fram ett språk som var plattform och produkt oberoende, d.v.s. det skulle fungera lika bra på Unix som på Windows.[12]

SQL är ett ANSI (American National Standards Institute) samt ISO (International Standards Organisation) datorspråk skapat för att kommunicera med och ändra i databaser.[13] Med hjälp av olika SQL kommandon kommunicerar SQL programvaran med databasen för att få fram eller modifiera informationen lagrad i databasen. Språket är uppbyggt på det sätt att man frågar databasen om informationen (exempel följer i kommande underrubriker).

Även om SQL är en ANSI standard så finns det många olika versioner av SQL, som t.ex. Microsofts Access eller open source varianten MySQL. Dessa olika versioner skiljer sig oftast genom att de har SQL-kommandon som är specifika för just dem själva. Trots att olika företag har respektive versioner av SQL måste de alla ha vissa grundläggande kommandon för att vara kompatibla med ANSI standarden.[14] Exempel på vissa grundläggande kommandon är SELECT, INSERT, DELETE, UPDATE, CREATE och DROP.[15]

Det finns en hel del olika SQL kommandon, för att inte flyga ut ur ämnet allt för mycket ska vi enbart nämna ett fåtal som vi anser är viktiga att gå igenom. Värt är att notera att man ej bör använda svenskspecifika bokstäver såsom Å, Ä och Ö i SQL kommandon, vi kommer däremot använda dessa bokstäver i våra exempel för enkelhetens skull. Normalt sett brukar man ersätta bokstäverna Å, Ä och Ö med A och O. D.v.s. kan Anställda bli t.ex. Anstallda och Överskott bli Overskott.

[12] Overgaard, Eriksson och Ek, 2006, s.139
[13] http://www.w3schools.com/sql/sql_intro.asp 2006-06-22 13:24
[14] Ibid.
[15] http://sqlcourse.com/intro.HTML 2006-06-22 15:49

3.3.2 SELECT[16]

SELECT kommandot används för att välja ut data från databasen. Man kan med hjälp av SELECT välja ut data från en eller flera tabeller, och frågorna kan göras väldigt enkla eller väldigt avancerade. Ett exempel på en enkel SELECT fråga kan vara:

```
SELECT * FROM tabellnamn
```

Den frågan väljer ut allt (* är lika med allt) från tabellen. Om man vill ha lite mer specificerad svar, t.ex. om man söker efter en viss person med ett visst efternamn från tabellen, kan frågan utökas med WHERE kommandot, exempelvis:

```
SELECT * FROM tabellnamn WHERE efternamn='Svensson'
```

Den frågan väljer ut, och visar, alla från tabellen vars efternamn är Svensson. Om man skulle vilja söka efter en specifik person med ett specifikt för- och efternamn skulle frågan kunna utökas med AND kommandot, exempelvis:

```
SELECT * FROM tabellnamn WHERE efternamn='Svensson'
AND förnamn='Anders'
```

Här väljs alla ut vars efternamn är Svensson och förnamn är Anders. Än så länge har vi bara valt ut all data som finns i en specifik tabell, om man söker efter en specifik information, säg ett telefonnummer, så skriver man SELECT frågan lite annorlunda, exempelvis:

```
SELECT telefonnummer FROM tabellnamn WHERE
efternamn='Svensson' AND förnamn='Anders'
```

Nu väljs bara informationen ut som finns under kolumnen telefonnummer i tabellen, och enbart där efternamnet är Svensson och förnamnet är Anders.

[16] http://sqlcourse.com/select.HTML 2006-06-22 17:33

3.3.3 INSERT[17]

INSERT används när man vill skapa nya poster i en tabell, exempelvis:

> INSERT INTO anställda (Efternamn, Förnamn, Position, Telefonnummer) VALUES (’Svensson’, ’Anders’, ’Chef’, 012-345678)

Här läggs informationen Efternamn, Förnamn, Position, samt Telefonnummer in i tabellen anställda. Först anges vilken tabell man ska lägga in posterna i med hjälp av kommandot INSERT, sedan anges värdena med hjälp av kommandet VALUES. Värt är att notera att strängar skrivs inom apostrofer (’) medan numeriska värden inte ska skrivas inom apostrofer.

3.3.4 DELETE[18]

DELETE används för att radera poster från en tabell. Om man ej anger ett villkor för DELETE kommandot så raderar den alla poster i hela tabellen. En DELETE fråga kan se ut på liknande sätt:

> DELETE FROM anställda WHERE efternamn=’Svensson’

Hade man glömt att ange villkoret ”WHERE efternamn=’Svensson’” och bara skrivit:

> DELETE FROM anställda

Så hade alla poster i hela anställda tabellen raderats.

3.3.5 UPDATE[19]

UPDATE används för att uppdatera en tabell. Om man t.ex. har skrivit fel på en post, som t.ex. efternamnet på en anställd, så är det väldigt omständligt att först radera den posten sedan lägga till den igen med korrekt information. Istället används kommandot UPDATE. Det skulle kunna se ut såhär:

> UPDATE anställda SET efternamn=’Svensson’ WHERE efternamn=’Svennson’

Den koden ändrar informationen ”Svennson” till ”Svensson” i posten efternamn i tabellen anställda.

---

[17] http://sqlcourse.com/insert.HTML 2006-06-22 17:51
[18] http://sqlcourse.com/delete.HTML 2006-06-22 18:15
[19] http://sqlcourse.com/update.HTML 2006-06-22 18:36

3.3.6 CREATE[20]

CREATE används för att skapa tabeller. Detta kommando kan kännas överflödigt eftersom det finns en hel del programvara som tar hand om databas och tabell skapandet, exempelvis Microsoft Access. Men det tillhör ett av de SQL grundkommandon som finns, därför ansåg vi det viktigt att gå igenom. Ett exempel på hur man skapar en tabell vore.

```
CREATE TABLE anställda(förnamn varchar(20), efternamn
varchar(30), ålder number(3), PRIMARY KEY (förnamn))
```

Här skapas tabellen anställda med attributen förnamn, efternamn och ålder samt primärnyckel anges till attributet förnamn.

3.3.7 DROP[21]

DROP används för att radera en hel tabell tillsammans med alla poster och attribut som är sparade i den. En DROP fråga skulle kunna se ut såhär:

```
DROP TABLE anställda
```

Detta skulle radera hela anställda tabellen.

[20] http://sqlcourse.com/create.HTML 2006-06-22 19:41
[21] http://sqlcourse.com/drop.HTML 2006-06-22 21:16

3.4 Användbarhet

3.4.1 Användarvänlighet och användbarhet

Vad är användarvänlighet? Det har visat sig vara en fråga som är väldigt svår att svara på. Folk som arbetar med datorer i sin vardag har oftast en allmän uppfattning om vad som är användarvänligt eller inte. De har emellertid mycket svårare att konkretisera vad som är användarvänligt. Föreställ er att ett företag beställt ett datasystem med kravet att systemet ska vara användarvänligt. När systemet sedan levereras så uppfattas den kanske ej som användarvänlig av uppdragsgivaren, medan systemutvecklarna anser den vara användarvänlig. En sådan tvist skulle förmodligen systemutvecklarna vinna, eftersom ordet användarvänlighet i sig inte är ett mått på hur ett system ska vara utvecklat. Vidare kan man fråga sig vad ordet användarvänligt system innebär. Ett system som är vänligt mot användaren? Enligt Allwoods definition av ordet användarvänlighet uppkommer:[22]

- Åtkomlighet
- Förenligt med och stöd för människans mentala funktionssätt
- Individualisering
- Hjälpresurser

Denna definiering är ofta väldigt luddig, och leder gärna inte till ett bättre system. Det är helt enkelt inte konkret nog.

Ordet användarvänlighet har blivit synonymt med begreppet användbarhet. Användbarhet däremot har en definition som är mycket mer konkret. Vilket enligt den internationella standardiseringsorganisationen ISO följer som:

”Den utsträckning till vilken en specificerad användare kan använda en produkt för att uppnå specifika mål, med ändamålsenlighet, effektivitet och tillfredställelse, i ett givet användningssammanhang. (ISO 9421-11, 1998)” [23]

Denna definition är mycket lättare att använda eftersom den är mycket mer konkret och gör det lättare att föra en diskussion huruvida det utvecklade systemet är användarvänligt eller inte. Definitionen beskriver även användbarhet som en mätbar storhet, man kan t.ex. mäta och säga att inom en specifik uppgift är system A 20 % mer användbar än system B. Även om definitionen i sig är väldigt konkret, så är det viktigt att inse att definitionen inte är en absolut storhet utan ett relativt begrepp. [24]

[22] Gulliksen & Göransson, 2002,s.57
[23] Ibid., s.62
[24] Ibid., s.62

Diverse användbarhetsexperter har ytterligare konkretiserat denne definition och gjort ytterligare detaljerade definitioner av den. Nielsen har sammanfattat användbarhet delen av definitionen med dessa huvudpunkter: [25]

- **Lätt att lära:** Systemet ska vara lätt att lära sig, så användaren snabbt kommer igång med arbetet

- **Effektivt att använda:** Systemet ska vara effektivt att använda när användaren väl lärt sig systemet.

- **Lätt att komma ihåg:** Man ska kunna komma tillbaka till systemet efter ett tags frånvaro och kunna använda systemet utan att behöva lära sig allt igen.

- **Få fel:** Systemet ska vara så väl utfört och förklarande att användaren ska slippa göra många fel. Om användaren nu skulle göra fel ska man kunna återvända till situationen direkt innan felet uppstod.

- **Subjektivt tilltalande:** Man ska tycka om att använda systemet. Det ska kännas tilltalande helt enkelt.

[25] Ibid., s.66

3.4.2 Innehållsdesign

Den främsta anledningen att en användare besöker en hemsida är på grund av innehållet, därför bör en hemsidas, eller datasystems, design frambringa innehållet, och inte tvärtom. När en användare besöker en sida tittar denne efter intressant innehåll och skummar genom rubriker och allt annat som kan visa vad sidan handlar om.[26] Därför bör man göra innehållet så lättläst och överskådligt som möjligt för att uppnå en så användbar sida som möjligt.

Studier har visat att det går ungefär 25 % långsammare att läsa från en dataskärm än på papper, samt att det är mycket obehagligare att läsa på en skärm. Folk ogillar även att behöva rulla texten.[27] På grund av detta bör man hålla textmassorna så korta som möjligt men samtidigt så informativa som möjligt. Studier visar även att cirka 80 % av användarna enbart ögnar genom texter på en dator. [28] Man bör därför utforma textstyckena i åtanke att de ska va snabba att läsa, informativa samt ha en överblickande design.

Detta kan åstadkommas med hjälp av beskrivande rubriker och underrubriker, punktlistor och att markera nyckelord med t.ex. färgad text. Men all design, oavsett hur bra den må vara, är värdelös om texten i sig inte är läsbar. För att göra texten så läsbar som möjligt finns det några läsbarhetsregler som bör följas, först och främst bör texten och bakgrunden ha färger som ger en hög kontrast, bäst är svart text med vit bakgrund.[29] Texten bör även bestå av ett lagom stort teckensnitt och bör bestå av sans-serif teckensnitt, sans-serif betyder utan klackar, och man ska undvika att använda mycket versaler.

[26] Nielsen, 2001, s.100
[27] Ibid., s.101
[28] Ibid., s.104
[29] Ibid., s.125

3.4.3 Navigation

En annan viktig aspekt med hemsidor är navigering. Vad hjälper det om man har en väldigt lättläst sida om man ej hittar informationen man söker efter eller om man ej vet var på sidan man befinner sig? Först och främst ska det vara helt klart och tydligt för användaren var denne befinner sig någonstans. Man bör därför ha med en logotyp på varje sida, helst i det över vänstra hörnet, logotypen ska helst även vara en länk tillbaka till startsidan.[30] Sedan bör navigationsgränssnittet följa användarens förväntningar, navigationen ska helst inte bestå av överraskningar utan användaren ska lätt kunna känna igen sig. Detta för att försäkra sig om att användaren ska kunna surfa på sidan effektivt.

Besökaren ska även lätt kunna avgöra var denne varit och kan gå, d.v.s. vilka sidor användaren besökt och vilka sidor som kan besökas. Om användaren vet vart man varit slipper man besöka samma sidor om och om igen i onödan, och det kan underlätta att snabbare ta sig dit man faktiskt vill gå. Ett sätt att påvisa vilka sidor som besökts är länkfärger. Man ska helst använda standardfärgerna för länkar, d.v.s. blåa understrukna för obesökta länkar och lila understrukna för besökta länkar för detta är något de flesta redan känner till och kommer ihåg. Sen bör man skapa ett strukturerat hierarkiskt system för länkarna, speciellt om sidan kommer att innehålla många länkar. Det går inte att ha t.ex. 100 länkar på startsidan utan att förstöra läsbarheten och användarvänligheten. Ett strukturerat länksystem kan uppnås t.ex. genom huvudlänkar och underlänkar.

Cirka 50 % av alla användare är sökbenägna och cirka 20 % är länkbenägna.[31] Som namnet antyder så vill sökbenägna användare använda sig av sökfunktioner och de letar först och främst efter sökfunktionen på sidan. Medan länkbenägna användare använder sig först och främst av länkar för att hitta informationen de söker. Eftersom en sådan hög siffra av användarna är sökbenägna bör sökfunktionen finnas på alla sidor, den bör vara lätt att hitta och lätt att förstå sig på. På hemsidor där sökfunktionen har hög prioritet bör den vara synlig hela tiden, så den bör läggas tillsammans med sidans logotyp på startsidan.[32]

[30] Ibid., s.189
[31] Ibid., s.224
[32] Ibid., s.168

3.5 Protokoll

3.5.1 Allmänt om protokoll

I datornätverk förekommer kommunikation mellan entiteter i olika system. Denna kommunikation är bara möjlig om datorerna använder sig av samma språk annars förstår de inte varandra. De behöver använda ett språk som alla parter kan förstå och där kommer protokoll in. Ett protokoll är en samling regler för vilken syntax, semantik och synkronisering som ska användas för att datakommunikation ska vara möjlig. I datakommunikation är protokoll ett begrepp för att beskriva de paketformat som transporteras mellan två datasystem och i vilken ordning det ska utväxlas.[33]

Generellt så har det flesta protokoll följande funktioner:

- Detektering av fysisk uppkoppling
- Handskakning, term för när datorerna etablerar villkor för datakommunikation
- Hur meddelanden ska starta och sluta
- Formatering av meddelande
- Felhantering vid datakommunikation
- Detektering av fel
- Avsluta sessioner

Det finns flera olika protokoll som kan implementeras av både hårdvara, mjukvara eller en kombination av båda. Det finns även protokoll optimerat just för strömmande media.[34]

[33] Jensen, Gjelstrup och Berti, 2000, s. 33-35
[34] Kunkel, 2001, s. 7-9

3.5.2 Strömmande media protokoll

Tcp/Ip protokoll skapades för att data skulle kunna transporteras från ett system till en annat tillförlitligt. Problemet med detta var att när multimedia introducerades krävdes det mer av protokollen för att kunna tillgodose denna service. När en användare nyttjar strömmande media finns det alltid alternativ för vilket protokoll som ska användas. Oftast väljer applikationerna åt användaren eftersom uppgiften kan vara svår att utföra för den oerfarne. Generellt sett försöker applikationer först med Real Time Streaming Protocol (RTSP) eller Microsoft Media Server Protocol (MMS) via ett UDP protokoll, detta är det optimala valet för bra media överföring. Om detta inte går så försöker applikationerna med RTSP/MMS fast via TCP protokollet. Om inget av dessa alternativ fungerar används Hypertext Transfer Protocol (http) som ger sämre kvalitet på överföringen.[35]

3.5.3 Open System Interconnection (OSI referensmodell)

För att datakommunikation mellan två datasystem ska vara möjlig så krävs det komplexa program och enheter som hanterar detta. För att göra det mer överskådligt har ISO tagit fram en modell som delar in kommunikationssystemen i olika skikt. Med hjälp av denna modell kan man definiera standarder och verktyg som används vid kommunikation.

Det finns 7 skikt

1. Fysiska skiktet
2. Datalänkskiktet
3. Nätverksskiktet
4. Transportskiktet
5. Sessionsskiktet
6. Presentationsskiktet
7. Applikationsskiktet

När det gäller strömmande media är vissa skikt mer relevanta än andra. Dessa är Nätverksskiktet, Transportskiktet och Applikationsskiktet.[36]

[35] Ibid., s. 12
[36] Jensen, Gjelstrup och Berti, 2000, s. 39-42

3.5.4 Nätverksskiktet

Om ett nät är seriekopplat med routrar ska systemen kunna adressera varandra med unika adresser. Skiktets uppgift är att ordna en adresstruktur och se till att datapaketen når sina destinationer. Alla protokoll baseras på IP som i sig ligger i nätverksskiktet. Nätverkstandarder för detta skikt är IP, IPX och x.25.[37]

3.5.4.1 IP (Internet Protocol)

Med hjälp av IP behöver inte data skickas direkt till mottagaren utan kan färdas olika vägar tack vare att den vet den unika IP adressen hos mottagaren. Klienter kan ha olika operativsystem men har fortfarande en IP adress vilket gör IP adresser till den minsta gemensamma nämnaren som alla system kan kommunicera via. Adressen är i siffror på 4 byte och kan se ut så här 255.255.255.255. När data skickas så packas mottagarens och sändarens adress in i en ram med själva innehållet och på så sätt hittar varje individuellt paket fram även fast de tar olika vägar. Mottagarens dator sätter ihop dessa paket sedan så att datan blir läsbar.[38]

Huvudfunktionen för IP är att ta emot data från TCP och UDP och packa om det för att sedan skicka vidare till mottagaren.

3.5.5 Transportskiktet

Detta skikt erbjuder säker datatransport mellan system som är direkt kopplade till varandra eller över ett större nätverk. Detta skikt är det första skiktet som stödjer peer-to-peer kommunikation som betecknar kommunikation mellan jämställda parter, t.ex. direkt mellan två användare, utan att passera en server eller kommunikationscentral. I detta skikt ligger protokollen TCP (Transmission Control Protokoll) och UDP (User Datagram Protocol) som bygger på IP funktionerna.[39]

[37] Ibid., s. 325-328
[38] Ibid., s. 348-354
[39] Ibid., s. 435-440

3.5.5.1 UDP (User Datagram Protocol)

UDP är ett förbindelselöst protokoll för överföring av enskilda paket över IP och ligger i Transportskiktet. Detta protokoll skiljer sig från TCP på det sättet att den inte uppehåller en bestående förbindelse mellan sändare och mottagare och den gör heller inga försök att skicka om paket om de har tappats bort av nätverket. När UDP-paket skickas iväg är det inte säkert att de kommer fram och om de kommer fram så kan de dyka upp i en annan ordning än de hade vid sändningen. Detta är en följd av att protokollet IP inte ger några garantier om paketens ordning. Om applikationen behöver pålitlig dataöverföring bör man använda TCP istället. UDP begränsar inte mängden data som skickas av ett applikationsprogram, för att undvika att nätet blir överlastat. Detta innebär att ett applikationsprogram som använder UDP kan belasta nätverket maximalt och förstöra för andra användare.[40]

UDP är väldigt bra för strömmande media för att det är så simpelt, den anpassar sig bra vid paket förlust och bromsar inte upp flödet på grund av säkerhetskontroller. Protokollet fortsätter att skicka och ta emot även fast det blir paketförlust och det är viktigt när det gäller media som spelas upp.[41]

3.5.6 Applikationsskiktet

Applikationsskitet är det skiktet som är närmast själva användaren. Skiktet gör det möjligt för användaren att få tillgång till informationen som överförts via nätverket, detta görs med hjälp av en applikation. I detta skikt ligger protokollen Real Time Streaming Protocol (RTSP), Microsoft Media Server Protocol (MMS), Hypertext Transfer Protocol (http) och File Transfer Protocol (FTP).[42]

[40] Ibid., s. 450-452
[41] Kunkel, 2001, s. 10-11
[42] Ibid., s. 11-12

3.5.6.1 Real Time Streaming Protocol (RTSP)

RTSP liknar http på många sätt i sin syntax och hur den exekverar. Det som skiljer dem åt är att RTSP ger användaren kontroll över hur mycket bandbredd som ska användas medan http tar så mycket som möjligt. Detta protokoll utvecklades för att fungera med unicast och multicast nätverk. Unicast är när varje användare som kopplar upp mot servern får en exklusiv ström av media medan Multicast är ett sätt att kommunicera som innebär att samma meddelande skickas till flera mottagare utan att behöva adresseras till var och en individuellt.

Protokollet är lokaliserat i applikationslagret i OSI referensmodellen. Den är också en öppen standard vilket betyder att företag kan använda sig av teknologin utan att oroa sig för copyright skydd. RTSP används bland annat av Apple och RealSystems i deras applikationer.[43]

3.5.6.2 Microsoft Media Server Protocol (MMS)

Detta protokoll är utvecklat av Microsoft och används till exempel i program som Windows Media Player när det gäller strömmande media. Protokollet fungerar så att den skickar kommandon via TCP och media via UDP eller TCP beroende på vilket alternativ som är optimalt för just den överföringen. Detta protokoll liknar RTSP väldigt mycket så därför har Microsoft gått över till det protokollet i sin senaste version av Windows Media Player.[44]

3.5.6.3 Hypertext Transfer Protocol (http)

Hypertext Transfer Protocol var länge standard för att transportera strömmande media även fast den ursprungligen var optimerad för statisk överföring. Den fördel http har över andra protokoll är att brandväggar och nätverk känner till den och kan då acceptera överföring smidigare. Detta är en av de få fördelarna med detta protokoll. Http protokollets största nackdel är att den är designad för att skicka och ta emot statisk data, därmed blir den strömmande datan känsligare för avbrott och fördröjningar. Felen visas i form av försämrad bild och ljudkvalitet eller förlorade delar av uppspelningen.[45]

[43] Ibid., s. 12-13
[44] Ibid., s. 13-14
[45] Jensen, Gjelstrup och Berti, 2000, s. 47-48

3.5.7 Media spelare

3.5.7.1 REAL Systems

REAL var bland de första som kom ut med en applikation som kunde strömma media över Internet. Redan 1994 lanserades Real Audio, ett program som kan strömma ljudfiler. 1997 släpptes REALPlayer som kunde strömma både ljud och video. Konkurrensen med Microsoft och Apple var en stor drivkraft för företaget som släppte flera versioner av sin spelare med nya funktioner varje gång.

REAL Audio använder sig av en encoder-decoder (komponent för att behandla en ström av information, även kallad codec) som heter ATRAC3 som är ett resultat av ett samarbete med SONY. Med denna codec är det möjligt att strömma ljud i CD kvalitet och surround sound. REAL Audio är optimerat för SMIL (Syncronised Multimedia Intergration Language) som är ett programspråk liknande HTML. Så om man vill ha en REAL Audio spelare som strömmar media på en hemsida är SMIL bäst att använda.[46]

3.5.7.2 Windows Media Series

Microsoft var sena med att se fördelarna med att ha ett program som kan spela upp strömmande media över Internet. Så sent som 1999 släppte Microsoft denna funktion till Windows Media Tools. Både ljud och video kunde strömmas med hög kvalitet med detta program.

Microsoft Media Player 9 är den senaste versionen av mediespelare från Microsoft. Den använder sig av ACELP.net (Algebraic Code Excited Linear Prediction) codec för att strömma ljud och är optimerad för Intelsystem med Internetprotokoll. Denna codec har med hjälp av optimerad paketdistribution minimerat felen som kan uppstå när man strömmar media. Ett problem med mediespelaren är att det är svårt att modifiera utseendet på spelaren om man vill integrera den på en hemsida.[47]

[46] Kunkel, 2001, s. 50-62
[47] Ibid., s.104-110

3.5.7.3 Quicktime

Quicktime är en mediespelare utvecklad av Apple. Det som är unikt för applikationen är att den är oberoende av vilken plattform som den används på. Den är också kompatibel med majoriteten av media typer som finns ute vilket gör produkten attraktiv för ljud och video utvecklare.

Quicktime Player använder AAC (Advanced Audio Coding) codec som är en öppen komprimerings standard designad som en efterträdare för MP3. AAC är kompletterad med flera andra codec för att maximera effektiviteten i kodningen. Detta gör att den är en av de mest avancerade codec som finns på marknaden.[48]

[48] Ibid., s.154-166

3.6 Programmering

3.6.1 XML

XML står för eXtensible Markup Language och är ett programmeringsspråk och ett filformat som härstammar från SGML (Standard Generalized Markup Language). Man utvecklade XML när man upptäckte att det allt för avancerade språket SGML var för begränsat för att uppfylla de krav som kom med den ökande användningen av Internet. Språket är en standard från W3C (World Wide Web Consortium) och släpptes 1996 men är inte helt komplett än, man kompletterar än idag detta språk.

XML påminner om HTML-språket men skiljer sig på det sättet att XML inte är bundet till att göra hemsidor bara. XML är ett metaspråk som tolkar andra programmeringsspråk och är därför oberoende av vilken mjukvara, hårdvara och skriftspråk den kan arbeta med. Språket har fått flera olika standarder för olika användningsområden. Det finns P3P (integritetsskydd på Internet), IOPT (avancerade formulär), XQuery (programmeringsgränssnitt), SVG (vektorgrafik) och XForms(webbsidor och mobilt Internet) som alla baserar på XML.[49]

XML är textbaserat och ger användaren möjlighet att beskriva ett dokuments struktur med hjälp av taggar (’<’ och ’>’). Inom dessa taggar skriver man kommandon och attribut som ska utföras av programmet som använder dokumentet.[50]

Själva språket skrivs oftast i Unicode med insprängda taggar och kan se ut på följande sätt.[51]

```
<?xml version="1.0" encoding="iso-8859-1"?>
<inbetalning>
<mottagare>
<namn>Nordic Steel</namn>
<konto typ="postgiro">123456-1</konto>
</mottagare>
<kund>349</kund>
<belopp valuta="SEK">237.00</belopp>
</inbetalning>
```

[49] http://www.xml.se/xml/vad.HTML 2006-07-12 16:38
[50] http://msdn.microsoft.com/XML/Understanding/default.aspx?pull=/library/en-us/dnxml/HTML/understxml.asp 2006-07-14 13:22
[51] http://www.xml.se/xml/vad.HTML 2006-07-12 16:54

3.6.2 PHP

PHP står för PHP: Hypertext Preprocessor och är ett open source scriptspråk som körs gentemot en server. PHP är utvecklat för webbprogrammering och kan inbäddas med HTML. PHP började tas fram i slutet av 1994 av en man som heter Rasmus Lerdorf, men det var först vid 1997 som PHP verkligen började ta fart och bli populärt. PHP är ett open source programmeringsspråk, vilket betyder att vem som helst får göra ändringar till språket och dess syntax. PHP utvecklades av folk som ansåg att de redan existerande programmeringsspråken inte var idealistiska för deras arbete. De ville ha ett programmeringsspråk som underlättade skapandet av dynamiska webbsidor och programvaror, samt som var lätt att bädda in i redan existerande HTML sidor. Iden var att PHP skulle vara snabbt att köra och säkert, ha förbättrad säkerhet och att PHP skriptet inte skulle vara synliga för slutanvändaren.[52]

PHP kan användas för att skapa dynamiska sidor med relativt enkla medel och PHP har inbyggt stöd för databaser. PHP skriptet exekveras direkt på webbservern och inte på användarens dator, detta möjliggör för utvecklaren av skriptet att gömma den bakomliggande koden. Skriptet man skrivit exekveras på servern och skapar ett HTML dokument som slutanvändaren ser. PHP kan även skapa bland annat PDF eller XML dokument.[53]

PHP kod ser ut som en blandning mellan C programmering och HTML kod. Man kan inbädda PHP kod inom HTML. Servern kan urskilja PHP genom en startande tag ”<?PHP” samt en avslutande tag ”?>”. Följande två exempel visar hur PHP skriptet ser ut. Samt vad användaren ser om denne försöker kolla på den bakomliggande koden för sidan:

```
<HTML>
   <body>
       <?PHP
       echo "Hej, jag är snäll!";
       ?>
   </body>
</HTML>
```

Ovan är PHP skriptet. Nedan följer vad användaren skulle se om denne skulle inspektera den bakomliggande koden för sidan.

```
<HTML>
   <body>
            Hej, jag är snäll!
   </body>
</HTML>
```

[52] http://www.zend.com/zend/art/intro.PHP 2006-07-18 17:54
[53] http://se.PHP.net/manual/sv/intro-whatcando.PHP 2006-07-18 18:12

4 Genomförande

4.1 Intervju

4.1.1 Första intervjun vid januari 2006

Vid en första intervju med vår kontaktperson Jens Landegren på företaget Nordic Steel framgick vad de väntade sig av hemsidan, samt vilka krav de hade på den. De gav oss väldigt fria händer när det gäller det mesta på sidan, de hade enbart tre direkta krav, vilka var att sidan ska kunna spela upp musik som ska strömmas genom nätet. Samt att sidans design ska återspegla företagets image, vilket innebär att sidan ska se metallisk ut, helst med kromfärger och dylikt. Det sista kravet var att all teknologi som används på sidan inte får vara kostsam sådan, eftersom företaget jobbar med en strikt budget. Företaget hade sedan några önskemål, men det var inga krav, på vad de gärna ville att sidan skulle innehålla. De ville ha ett ”content management system” (CMS) samt en sorts gästbok.

Efter intervjun satte vi oss ner och diskuterade vad vi anser oss kunna göra och inte kunna göra. Och det framkom rätt snabbt att ett CMS skulle vara alldeles för avancerat för oss. Vi tog då kontakt med företaget och frågade dem om de skulle tänka sig att kunna uppdatera sidan på ett annorlunda sätt. Vilket de gick med på, det bestämdes då att en av de anställda på företaget skulle uppdatera nyheter och dylikt på sidan direkt i HTLM sidorna. Resten ansåg vi oss kunna klara utan några större problem.

4.1.2 Andra intervju vid Juni 2006

Det var tänkt att vi skulle påbörja examensarbetet vid januari 2006, men tyvärr hade vi alldeles för mycket att stå i under perioden mars till juni, så vi bestämde oss för att skjuta upp examensarbetet om det vore möjligt. Vi tog kontakt med företaget och frågade om det vore acceptabelt för dem om vi försenade arbetet med hemsidan till sommaren/hösten. Företaget hade inga som helst problem med det, och vi bestämde att vi skulle ha ytterligare ett möte i början av juni. Under tiden mellan januari till juni utvecklades våra programmeringskunskaper väldigt mycket, speciellt inom området PHP och JavaScript, och vi ansåg oss därmed kunna fullfölja företagets originella krav om ett CMS.

4.2 Skapandet av media spelaren

4.2.1 Val av teknologi

Företaget ville ha ljudfiler som skulle strömmas, med detta i åtanke kollade vi igenom vilka system, eller teknologier, vi hade till vårt förfarande och vi bestämde oss för att använda Macromedia Flash för att skapa vår media spelare för att strömma ljud filer i. Macromedia Flash använder sig av http protokollet för att strömma media i. Http har lägst kvalitet av alla protokoll tillgängliga för att strömma medier i. Den största anledningen till att vi valde att strömma media i Macromedia Flash var mest på grund av de ekonomiska krav vi hade på oss, vi kände även att vi ville lära oss mer om Macromedia Flash och hur man utvecklar Flash dokument.

4.2.2 Praktiskt genomförande

Flash har en inbyggd media spelare, men den är väldigt simpel och väldigt statisk och att ändra design samt funktionalitet på den är väldigt krångligt. Så vi bestämde oss för att försöka göra vår egen media spelare med hjälp av Macromedia Flash inbyggda programmeringsspråk ActionScript 2.0. Det stod klart och tydligt för oss att vi måste skapa media spelarens funktioner med hjälp av ActionScript, men vi hade inte tillräcklig med kunskap för att utföra detta. Så vi började arbetet med att samla in information om hur man kan skapa strömmande media spelare med hjälp av ActionScript, när vi ansåg oss ha tillräckligt med information påbörjade vi kodningsarbetet och designen av media spelaren.

Arbetet påbörjades med att söka upp information om hur man kan göra media spelare i Flash. Det finns redan en massa media spelare som är färdigskapta som man kan använda sig av, men då måste man mer eller mindre följa deras design och funktionalitet. Så vi bestämde oss för att försöka på vår egen och i värsta fall, om vi ej lyckas, använda en redan existerande media spelare. Eftersom vi redan hade existerande media spelare och dess färdigskrivna kod var det inte så svårt att utveckla vår egna spelare. Med hjälp av de färdigskrivna media spelarna kunde vi se hur man skulle kunna utveckla sin egen, vi använde oss mycket av färdigskrivna funktioner för själva strömningen av musikfilerna och ändrade mest på koden för funktionaliteten, utseendet och inladdning av låtarna.

Först skapade vi en enkel design med ett fåtal knappar för att kunna testa funktionaliteten av media spelaren längs med tiden vi implementerade funktionerna. Sedan påbörjade vi programmeringsarbetet, det största problemet bestod av att implementera ljudfilerna till media spelaren och skapa låtlistan för dem.

Det skulle fungera om man la in ljudfilerna och skapade låtlistan direkt i Flash dokumentet. Men detta förfarande skulle kräva att man för varje ändring skulle vara tvungen att gå in i Flash dokumentet och manuellt lägga till och ta bort låtar. Detta skulle vara allt för krångligt och avancerat för de anställda på företaget. Låtlistan och låtarna var därmed tvungna att ligga utanför Flash dokumentet. Därmed behövdes det en funktion för att bädda in dem i media spelaren.

Flash har inbyggt stöd för att läsa från XML dokument och extrahera data från dem, detta utnyttjade vi till full grad. XML dokumentet innehåller information om låtens titel samt dess filnamn och sökvägen till filen. Det kan t.ex. se ut såhär:

```
<?xml version="1.0" ?>
<songs>
<song display="Changes" url="./mp3/2Pac Changes.mp3" />
<song display="Dancing Queen" url="./mp3/Dancing Queen.mp3" />
</songs>
```

Det enda som behövs ändras för att lägga till eller ta bort låtar från media spelaren är därmed det externa XML dokumentet. Media spelaren läser av XML dokumentet varje gång den startar och skapar en intern låtlista utifrån den.

För att underlätta skötandet av XML dokumentet skapade vi ett CMS med hjälp av några PHP skript som skulle modifiera XML dokumentet och underlätta uppladdningen av mp3 filer till servern. Den första versionen av media spelarens CMS krävde att användarna laddade upp låtarna till servern manuellt och sedan ändrade XML dokumentet i en sorts text editor. Detta system ansåg vi vara allt för krångligt och inte användarvänligt alls.

Så vi började koda ett helt nytt system som skulle vara så automatiserat som möjligt och användarvänligt som möjligt. Det färdiga CMS bestod av två delar. Ena delen var till för att ladda upp filer till mp3 katalogen på servern (där alla låtar ska komma att ligga). Den andra delen var till för att radera låtar från mp3 katalogen på servern. XML dokumentet skapas sedan automatiskt med hjälp av ett skript som läser av alla filer i MP3 katalogen och skriver sedan in informationen om deras titlar och filnamn i XML dokumentet.

Media spelaren läste sedan av låtarnas titel och längd utifrån själva mp3 filerna. Detta genom att läsa av de så kallade ID taggarna på mp3 filerna samt genom att läsa av hur lång låten man valt att spela upp är. En ID tagg innehåller information om bland annat låtens titel, artist, album etc.

4.3 Programmeringsfasen, utveckling av sidan

4.3.1 Val av metod:

Med tanke på vår utbildning, samt inriktningen av programmeringsspråk inom den, hade det naturliga valet för utvecklingsarbetet varit ASP.NET med VB.NET som programspråk och Microsoft Access som databas. Vi valde dock att inte använda oss av ASP.NET, utan av PHP, HTML samt JavaScript. Detta val gjordes av ett fåtal anledningar, en av de största anledningarna var för vi ansåg oss bemästra PHP kodning mycket bättre än VB kodning, samt för att hela iden med arbetet var att utveckla en enklare hemsida åt ett företag och vi ansåg att PHP skulle täcka alla krav företaget ställde. Den sista anledningen var att vi ville lära oss mer om PHP och kunna tillämpa det i ett skarpt fall och såg här en stor chans till detta.

4.3.2 Praktiskt genomförande

För att koden ska vara lättförstådd, strukturerad och lätt att felsöka använder vi oss av ungersk notation. Enligt ungersk notation ska man döpa variabler med ett prefix som ska motsvara dess datatyp. Exempelvis ska en label ha t.ex. prefixet lbl före variabeln. Dessa prefix bestäms i förväg av utvecklarna och ett dokument med en lista över variabler samt dess prefix skapas. Detta dokument har utvecklarna sedan tillgängligt vid behov. Exempel på ungersk notation kan se ut på följande sätt:

```
lblStyle.textAlign = "left";
strTitle = ”Song title”;
```

Detta underlättar t.ex. förståelse av koden, samt om något fel skulle uppstå i systemet, t.ex. att orden ”Song title” inte skrivs ut, kan man härstamma felet till att en sträng är felskriven (str står för string vilket är sträng på svenska). Det är då enkelt att söka upp alla tillgängliga strängvariabler snabbt och fixa problemet.

Vi valde att inte utveckla vårt system med ett objektorienterat förfarande utan med ett procedurellt förfarande. Objektorienterat uppbyggda system är mer eller mindre ett måste när man ska utveckla stora system. Eftersom vårt system skulle komma att vara relativt liten skulle det räcka med ett procedurellt utvecklat system. Den största skillnaden mellan de bägge är att i ett objektorienterat system skrivs generella funktioner som kan användas till mer än en specifik uppgift. Medan i ett procedurellt system används funktioner som är skapade att utföra enbart en specifik uppgift.

Funktioner används inom utvecklingen av programmen vid största möjliga mån. Användning av funktioner minskar först och främst utvecklingstiden, samt hjälper koden att bli mer lättförstådd och lätt att felsöka. En funktion är ett antal instruktioner som man bestämt sig att hålla ihop. Istället för att skriva exakt samma instruktioner i t.ex. 40 olika PHP skript kan man skriva det i ett separat skript och sedan påkalla funktionen där den behövs. Exempelvis kan en funktion se ut på följande sätt:

```
function footer() {
echo ”<hr>”;
echo ”<br>”;
echo “Nordic Steel”;
}
```

Denna funktion skriver ut en linje samt skriver ut en ny rad med texten ”Nordic Steel”. Istället för att skriva ut de tre raderna överallt man vill använda detta kan man skriva ut namnet på funktionen (footer()).

För att säkerställa en gemensam design på hemsidans sidor använder vi oss av så kallade CSS-mallar (Cascading Style Sheets). En CSS-mall är en stilmall där man ställer in t.ex. bakgrunds färg, typsnitt, teckenstorlek med mera. Denna mall påkallas sedan i varje sida. Anledningen till att använda en CSS-mall är för det underlättar att styra designen på sidan. Om man t.ex. vill ändra bakgrundsfärg på alla sidor sköts detta genom mallen, istället för att gå in och ändra alla sidor manuellt. Nedan följer ett exempel på hur en formatering i en CSS-mall kan se ut:

```
Body
{
        font-family: Verdana, Arial, Helvetica, sans-serif;
        font-size : 14px;
        color : #000000;
        background-color: #EEEEEE;
}
```

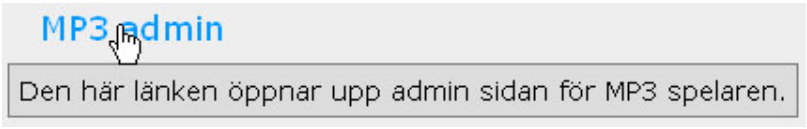
I exemplet ovan har bakgrundsfärg formaterats, samt typsnitt, teckenstorlek och teckensnittsfärg. Genom att t.ex. ändra teckenstorlek i CSS-mallen ändras den överallt på sidan.

För att underlätta förståelse av koden, samt underlätta för framtida ändringar eller påbyggningar, används informativa kommentarer. Kommentarerna ska skrivas på ett sätt att en utomstående ska kunna sätta sig in i programkoden och förstå den utan större problem. Felsökning kan även göras lättare med utförliga kommentarer. Det kan t.ex. se ut som på följande exempel:

```
//kollar om katalogen innehåller mp3 filer och skriver ut dem på sidan.
if(!($fil == "." | |$fi l== "..") && eregi('mp3', $fil))
{
        echo "$fil<br>";
}
```

Hade man uteslutit kommentaren hade koden vart mycket mer svår att förstå, men med hjälp av kommentaren förstår man snabbt vad koden utför.

Vi använde oss även av JavaScript för att utveckla små hjälpmeddelanden på admin delen av sidan, samt för att styra storleken dynamiskt på de olika sidorna istället för att använda en statisk storlek, vilket skulle innebära att vissa sidor skulle kräva en scroll i onödan eftersom sidorna skulle vara olika stora beroende på innehåll. Exempelvis kan ett hjälpmeddelande se ut på detta sätt:



Figur 4-1 Exempel på hur ett hjälpmeddelande kan se ut

I exemplet ovan dyker det upp ett hjälpmeddelande under länken som visar vart länken leder någonstans.

Vi använde oss av MySQL för att skapa databasen. När rapporten skrevs innehöll databasen en tabell som används för gästboken.

5 Resultat

Vårt examensarbete resulterade i en hemsida med en inbyggd strömmande mediespelare samt ett användarvänligt system för att underhålla sidan. Företaget var nöjda med hemsidan som den var när denna rapport skrevs, men vi har planer på att i framtiden lägga till funktioner som underlättar underhållet av hemsidan. Bland annat en administrativ del för att sköta sidans länk sektion, samt en administrativ del för att sköta om media sidan.

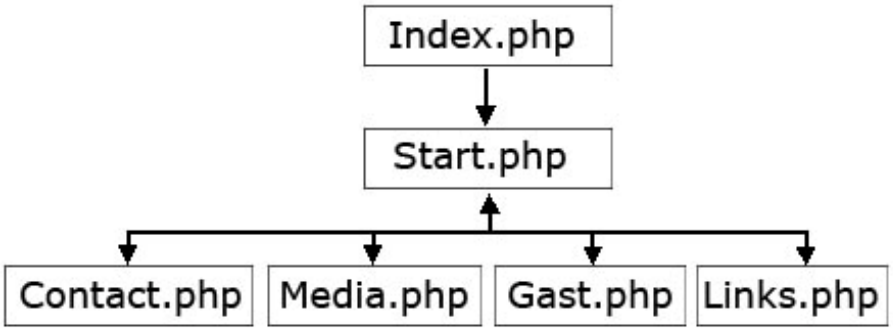
Största fokus på sidan har lagts på mediespelaren samt systemet för dess underhåll. Mycket arbete har lagts ner på programmeringen av funktionerna för uppladdningen av låtar och administrering av forumet. Detta gör det mycket lättare för arbetsgivaren och de anställda på företaget att uppdatera och sköta om sidan utan vår hjälp.

Mediespelaren är skapad för att vara så automatiserad och lätt underhållen, som möjligt. Användarna av systemet behöver inte uppdatera låtlistan till mediespelaren manuellt, allt sköts med hjälp av programmerade funktioner.

När vi skapade gränssnittet för sidan utgick vi från företagets krav, samt för de regler som finns för att skapa användarvänliga system eller sidor. Vi har utefter dessa krav och regler försökt skapa en så lättanvänd sida som möjligt, både för företaget samt användarna, men samtidigt försökt hålla den så nära företagets krav som möjligt.

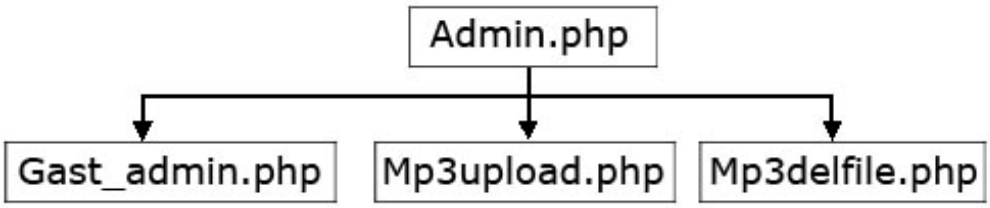
Vi skapade hela sidan från grunden med hjälp av PHP programmering. Sidan består av två delar. Den ena delen är menad för besökarna och är den innehållsrika delen. Den andra delen är menad för företaget självt och är till för att underhålla sidan.

5.1 Hemsidans navigering och sidor



Figur 5-1 Navigationsöversikt över startsidan

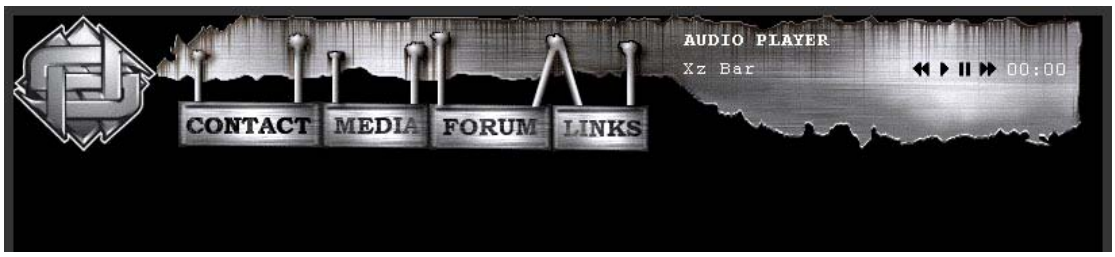
Figur 5-1 visar ett flödesschema över hemsidans struktur. Hemsidan har en huvudsida, index.PHP, varifrån det är länkat till alla andra sidor. Start.PHP laddas först in som startsida, index.PHP är bara ett ramverk för länkar och dylikt. Man kan nå vilken sida som helst oavsett vilken sida man är på.



Figur 5-2 Navigationsöversikt för adminsidan.

Figur 5-2 visar ett flödesschema över adminsidans struktur. Adminsidan finns ej länkad från huvudsidan (vilket framgår i figur 5-1). Detta för att folk ej ska råka snubbla in i adminsidan. Det går att nå vilken sida som helst från adminsidan, men ej vice versa. Man måste gå tillbaka till adminsidan för att byta sida. Som det framgår ur figuren kan man administrera gästboken, uppladdningen av låtar till mediespelaren samt radering av låtar från mediespelaren.

5.1.1 Index.PHP



Figur 5-3 Index sidan

Index sidan innehåller menyn, dess länkar, mediespelaren samt en dynamisk del menad för innehåll. Sidan är uppdelad i två delar. Den övre innehåller menyn som är statisk, bortsett från media spelaren. Den undre delen av sidan är reserverad för dynamisk inladdning av andra sidor, det är här hemsidans själva innehåll visas. Det finns ingen synlig skiljegräns mellan menyn och den dynamiska delen av sidan.

Index sidan finns med på varenda sida man besöker på hemsidan, förutom på adminsidorna.

Mediespelaren är belägen längst upp till höger på sidan. Den innehåller låt titel (Xz Bar är titeln på låten som spelas just nu), hur mkt av låten som spelats, samt de vanligaste funktionerna man förväntas hitta på en mediespelare, såsom paus/play funktion samt valmöjligheten att byta låt. Det finns även en förklarande text ”AUDIO PLAYER” så besökare kan få en uppfattning av vad sidan kan handla om.

5.1.2 Start.PHP

Figur 5-4 Startsidan

Detta är startsidan för hemsidan. Här hamnar man så fort man kommer in i hemsidan, eller om man klickar på företagets logo som är placerad i övre vänstra hörnet. Startsidan innehåller lite information om företaget och om dess affärsidé.

5.1.3 Contact.PHP

Figur 5-5 Kontakt sidan

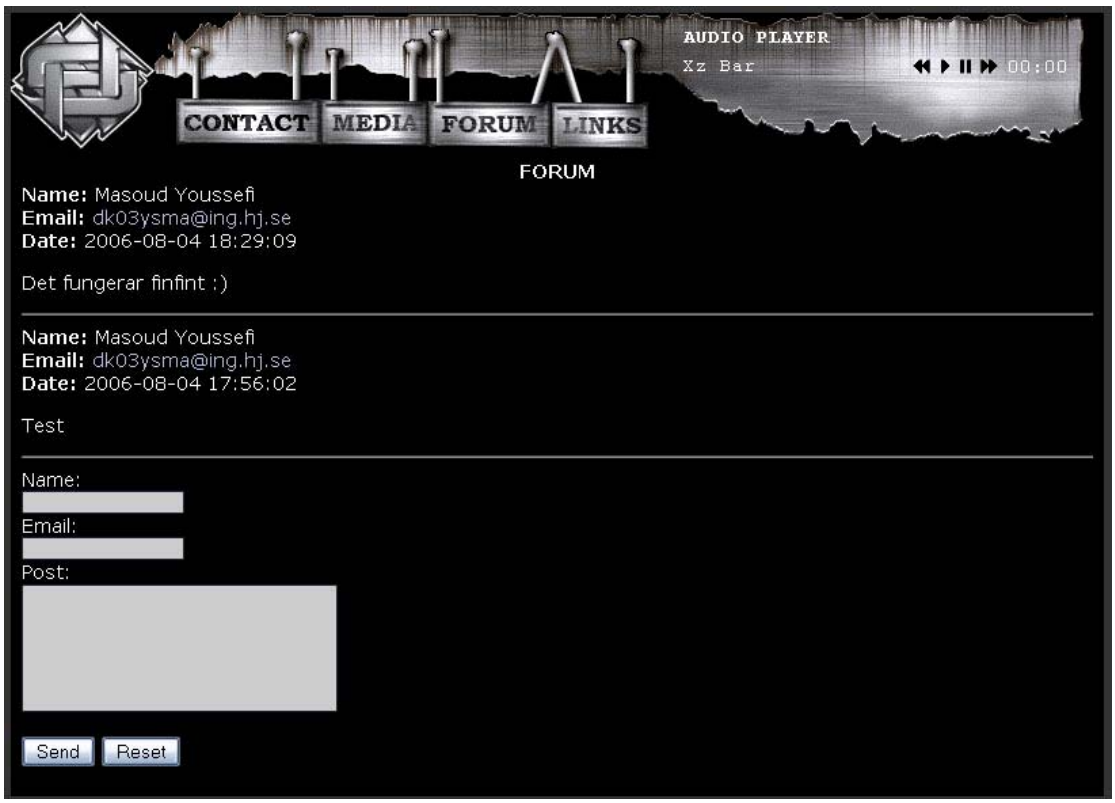
Kontakt sidan innehåller bilder på ägarna av, samt artisterna som jobbar på, företaget. Under bilderna finns respektive persons namn. På kontakt sidan finns även en länk till deras gemensamma e-post samt en länk till gästboken.

5.1.4 Media.PHP

Figur 5-6 Media sidan, där besökarna kan ladda ner olika medieklipp

Media sidan kommer att innehålla nerladdbara låtar och videoklipp, t.ex. på musikvideos. När detta examensarbete skrevs hade företaget ej bestämt vilket innehåll de ville ha med på media sidan, den ser därmed rätt tom ut.

5.1.5 Gast.PHP



Figur 5-7 Gästbok sidan, en enklare gästbok för besökare att skriva i

Gästbok sidan är till för folk att skriva i, antingen om de vill komma i kontakt med företaget, eller om de bara vill skriva något för skoj skull. Det krävs att man ska fylla i sitt namn och ett inlägg, e-post är däremot inte obligatoriskt. Gästboken fyller automatiskt in datum och tid. Gästboken använder sig av en databas där all om inläggen samt dess användare information finns.

5.1.6 Links.PHP

Figur 5-8 Länk sidan, denna sida innehåller länkar till andra hemsidor

Denna sida ska innehålla länkar till andra sidor framöver. När detta examens arbete hade skrivits fanns de inga specifika länkar företaget ville ha med, därmed är denna sida rätt tom för tillfället.

5.1.7 Admin.PHP

Figur 5-9 Inloggning till adminsidan

Trots att admin delen av hemsidan är skild från huvudsidan och det enda sättet man kan komma åt den är genom att skriva in direkta länken till den i ens Internet bläddrare, så ansåg vi att en sorts inloggnings funktion vore att föredra. Ifall folk skulle råka snubbla in i adminsidan.

Figur 5-10 Adminsidans startsida

Startsidan för adminsidan innehåller enbart tre länkar. Inte mycket jobb har lagts ner på designen av denna sida.

5.1.8 Gast_admin.PHP

Figur 5-11 Adminsidan för gästboken

Detta är adminsidan för gästboken. Längst ner på sidan finns en länk tillbaka till startsidan för admindelen. Varje skrivet inlägg har med användarens namn, e-post, inlägg, datum inlägget var skrivet, inläggets id nummer i databasen samt en knapp för att radera inlägget. Id numret är medlagt ifall företaget själva skrivit något i gästboken och vill ändra innehållet så kan de göra detta genom att ändra i databasen.

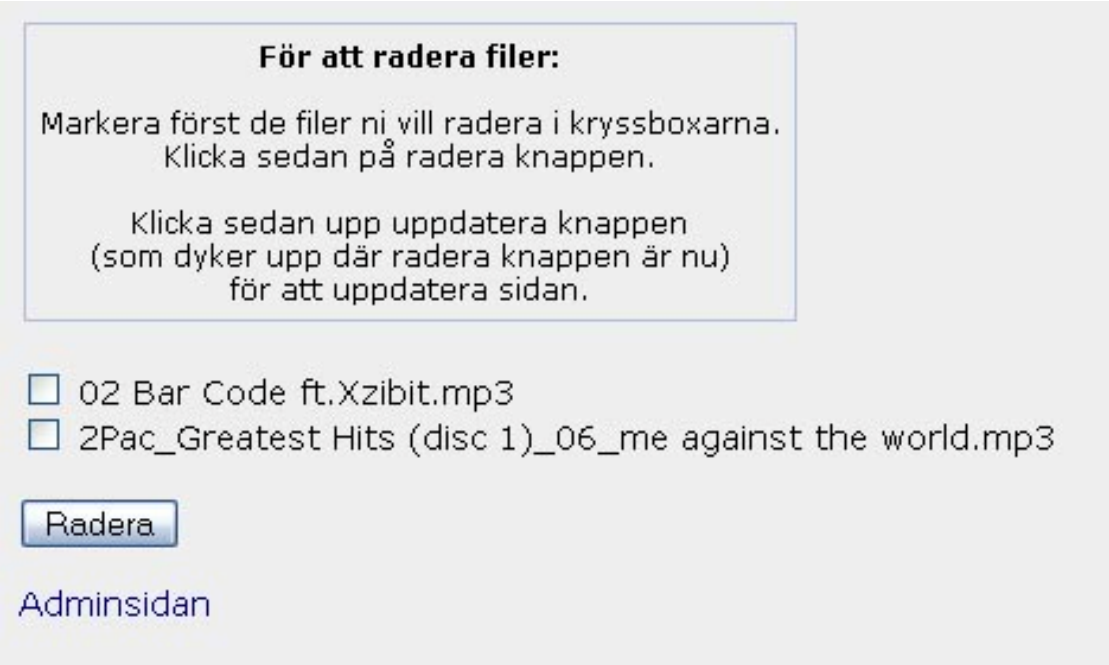
5.1.9 Mp3upload.PHP

Figur 5-12 Sidan för att ladda upp mp3 låtar till mediespelaren

Denna sida laddar upp mp3 låtar till mediespelaren. Uppdateringen av mediespelaren är automatiserad, det enda användaren behöver göra är bläddra fram vilken låt denne vill ladda upp och trycka på "skicka" knappen. Mediespelaren uppdateras automatiskt med den nya låten.

Längst upp på sidan finns en förklarande text för att vägleda användaren. Sidan är uppdelad i två delar, översta delen är till för att ladda upp låtar och understa delen visar vilka låtar mediespelaren innehåller. Längst ner på sidan finns det även en länk till startsidan för admin delen.

5.1.10 Mp3delfile.PHP



Figur 5-13 Sidan för att radera mp3låtar från mediespelaren

Denna sida används för att radera mp3 låtar från mediespelaren. Uppdateringen av mediespelaren är automatiserad, det enda användaren behöver göra är att markera de låtar denne vill radera och klicka på radera knappen.

Även denna sida har en förklarande text längst upp. På mitten av sidan är alla låtar som är inlagda i mediespelaren uppskrivna. Till vänster om dem finns en checkbox som används för att markera vilka låtar man vill radera. Längst ner är det länkat till startsidan för admin delen.

6 Slutsats och diskussion

När vi påbörjade arbetet hade vi ingen som helst aning om hur vi skulle gå tillväga. Vi visste vilka krav vi hade på oss och vad vi var tvungna att uppnå, men inte hur vi skulle nå dit. Än mindre visste vi hur vi skulle utforma examensarbetet och vad tyngdpunkten i dokumentet skulle ligga på. Efter en diskussion med vår handledare bestämde vi att fokusera examensarbetet på strömmande medier och utformningen av dessa.

När vi påbörjade skapandet av mediespelaren hade vi det i bakhuvudet att vi borde göra det så enkelt som möjligt för företaget att kunna uppdatera den med ett administrativt system. Eftersom vi hade lite kunskap och erfarenheter av PHP vid detta skede visste vi att ett sådant system skulle kunna utvecklas med hjälp av PHP. Vi bestämde oss därmed för att försöka utveckla systemet för detta med PHP programmering.

Ursprungligen var det tänkt att vi skulle lära oss mest om strömmande medier, hur de fungerar och hur man kan implementera dem i webbsidor. I slutändan blev det att vi lärde oss mest om PHP, databaser i MySQL och webbprogrammering. Under arbetets gång har vi lärt oss väldigt mycket mer om PHP och MySQL och har med hjälp av våra nyfunna kunskaper kunnat uppnå alla de krav och mål vi hade satt. Emellertid tog det oss en hel del tid att lära oss allt detta, vilket ledde till att massa tid från utvecklandet av hemsidan gick åt för att lära sig PHP. Vi förlorade även en del tid på dålig planering av vissa moment, t.ex. själva utvecklandet av sidan. Detta i sin tur ledde till att alla funktioner vi gärna ville ha med på sidan, men inte hade som krav, inte hann bli färdiga eftersom vi var tvungna att pausa utvecklandet av sidan för att skriva på examensarbetet. Vi ville bland annat ha en funktion för strömmande av videofilmer.

När detta examensarbete skrevs fanns det en enklare sorts gästbok på hemsidan, vi ville även implementera en sida där företaget skulle kunna ladda upp diverse media filer, som t.ex. videoklipp och låtar, med hjälp av en dynamisk administrativ sida och skriva medföljande förklarande text till dem, så de slipper koda i HTML sidorna manuellt.

Trots detta är vi nöjda med arbetet vi gjort och vi tog till mycket ny kunskap inom speciellt webbprogrammering. Vi har även lärt oss vikten av att planera mycket noggrannare. Värdefull tid gick förlorad p.g.a. dålig planering vilket resulterade i att själva sidan, dess design och funktioner, blev lidande.

7 Referenser

7.1 Publicerade referenser

Apelkrans, Mats och Åblom, Carita (2001)OOS/UML – *En objektorienterad systemutvecklingsmodell för processorienterad affärsutveckling*, Studentlitteratur, Lund, ISBN 91-44-02138-0.

Overgaard, Jörgen; Eriksson, Ulrika och Ek, Jesper (2006) *PHP 5 Programmering*, Pagina Förlags, ISBN 91-636-0800-6.

Gulliksenn Jan och Göransson, Bengt (2002) *Användarcentrerad systemdesign, Studentlitteratur*, Lund, ISBN 91-4-02029-5.

Nielsen, Jakob (2001) *Användbar webbdesign*, Liber AB, ISBN 91-47-03612-5.

Kunkel, Tobias (2001) *Streaming media*, Peason Education Deutschland, ISBN 0-470-84724

Jensen, Stig; Gjelstrup, Arne och Berti, Valentino (2000) *Data kommunikation*, Liber, ISBN 91-47-01590-x

7.2 Icke publicerade referenser

Vad är XML?,
http://www.xml.se/xml/vad.HTML (Acc 2006-07-12 16:38)

Zend Technologies - Articles - Introduction to PHP,
http://www.zend.com/zend/art/intro.PHP (Acc 2006-07-18 17:54)

SQL introduction,
http://www.w3schools.com/sql/sql_intro.asp (Acc 2006-06-22 13:24)

SQLCourse - Lesson 1: What is SQL?,
http://sqlcourse.com/intro.HTML (Acc 2006-06-22 15:49)

SQLCourse - Lesson 3: Selecting Data,
http://sqlcourse.com/select.HTML (Acc 2006-06-22 17:33)

SQLCourse - Lesson 5: Inserting Into a Table,
http://sqlcourse.com/insert.HTML (Acc 2006-06-22 17:51)

SQLCourse - Lesson 7: Deleting Records,
http://sqlcourse.com/delete.HTML (Acc 2006-06-22 18:15)

SQLCourse - Lesson 6: Updating Records,
http://sqlcourse.com/update.HTML (Acc 2006-06-22 18:36)

SQLCourse - Lesson 4: Creating Tables,
http://sqlcourse.com/create.HTML (Acc 2006-06-22 19:41)

SQLCourse - Lesson 8: Drop a Table,
http://sqlcourse.com/drop.HTML (Acc 2006-06-22 21:16)

Introduktion till databaser - Lektion 1: Introduktion till databaser,
http://www.pmdatautbildning.se/webbkurs/db/1DB.htm (Acc 2006-06-17 16:12)

Samlingar på nätet - 3. Databasteori,
http://dub.abm.uu.se/mus/mus_saml3.HTML (Acc 2006-06-17 16:38)

Museer på nätet - 4. Relationsdatabaser i praktiken,
http://dub.abm.uu.se/mus/mus_saml4.HTML (Acc 2006-06-17 17:44)

Att skapa en objektmodell - Lektion 3: Objektmodellen,
http://www.pmdatautbildning.se/webbkurs/db/3DB.htm (Acc 2006-06-17 21:42)

8 Sökord