Janne Carlsson

RESERVERINGSSYSTEM FÖR SITTPLATSER SOM WEBBTJÄNST

Företagsekonomi och turism

2011

VASA YRKESHÖGSKOLA

Utbildningsprogrammet för informationsbehandling

ABSTRAKT

Författare	Janne Carlsson
Lärdomsprovets titel	Reserveringssystem för sittplatser som webbtjänst
År	2011
Språk	svenska
Sidantal	50
Handledare	Kimmo Paulaharju

Detta lärdomsprov behandlar uppbyggnaden av ett internetbaserat reserveringssystem för sittplatserna i Lumivaaras kyrka.

Tanken med lärdomsprovet är att ge en inblick i uppbyggnaden av systemet. Det skall ske genom att presentera själva systemet, samt de tekniker, programmeringsspråk och verktyg som använts.

Läsaren kommer att få en presentation av de olika delarna i systemet, men den huvudsakliga presentationen kommer att inriktas på själva sittplatsernas presentation i systemet, eftersom det har varit den största delen av arbetet.

Reservationssystemet är främst uppbyggt med PHP, XHTML och CSS. Som databas har MySQL använts.

Ämnesord PHP, MySQL, XHTML, CSS, Reservationssystem

VAASAN AMMATTIKORKEAKOULU

UNIVERSITY OF APPLIED SCIENECS

Utbildningsprogrammet för informationsbehandling

ABSTRACT

Author	Janne Carlsson
Thesis	A Seat Reservation System as a Web Service
Year	2011
Language	Swedish
Pages	50
Supervisor	Kimmo Paulaharju

This thesis described the development of a web –based system for the reservation of seats at Lumivaara church.

The aim was to give an overview of the system. This happened through a presentation of the system itself, the techniques as well as the programming languages and tools used.

The reader is given a general overview of the different parts of the system, but the main area of concentration will be the seats that are shown in the system, as this section was the most time consuming and labourous part of the whole process.

The seat reservation system is mainly built with PHP, XHTML and CSS. As a database MySQL was used.

Keywords PHP, MySQL, XHTML, CSS, Reservations -system

INNEHÅLL

FÖRTECKNING ÖVER BILAGOR

1 INLEDNING

1.1 Bakgrund

I utbildningsprogrammet för informationsbehandling har jag kommit i kontakt med en del programmering och webb -design. Dessutom har jag länge varit intresserad att lära mig PHP -programmering. I samband med att välja ämne för lärdomsprovet så fick jag möjlighet att komma igång med PHP –programmering. Via min handledare Kimmo Paulaharju fick jag som uppdrag att göra ett system för reservering av sittplatser åt Lumivaaras Kirkko.

Eftersom jag inte hade någon kunskap om PHP –programmering från förr så genomgick jag en snabbkurs med min klasskamrat Mathias Renlund, som är en erfaren PHP –programmerare. Med honom gick jag igenom grunderna i PHP och hade möjlighet att under arbetets gång söka efter hjälp. Mathias Renlund lånade mig en bok, PHP5 programmering (Overgaard, Eriksson & Ek, 2004) som jag använde som snabbreferens under arbetets gång, samt gjorde övningsuppgifter ur den, vilket gav mig en lämplig startpunkt för slutarbetet.

1.2 Uppdrag

Uppdragsgivaren för detta arbete är Lumivaaras kyrka och som kontaktperson fungerade Kari Rapo, som är anställd av skolan. Tanken med systemet var att aktiva kyrkobesökare skulle kunna gå in på kyrkans hemsida och reservera en sittplats åt sig själv.

Lumivaaran kirkko är en kyrka i Karelen som efter krigstiden hamnade på den ryska sidan av gränsen. Kyrkan rymmer 720 sittplatser som uppdelas mellan kyrksalen och läktaren.

1.3 Lärdomsprovets syfte och mål

Beställarens behov var att erbjuda ett system för reservering av sittplatser för användare. Kraven på systemet kommer att tas upp i ett senare kapitel och de är grundpelarna som hela slutarbetet bygger på.

Det är självklart att en webbtjänst skall vara användbar och fundera enligt standarder som datoravändare är vana vid. Så som placering av innehåll, vilket man bör beakta vid tillverkning av en tjänst.

Målet var att bygga ett reserveringssystem som höll beställarens krav och dessutom var användbart och fungera som den central plats för reservering av sittplatser.

1.4 Avgränsningar av lärdomsprovet

Lärdomsprovet kommer att behandla det slutliga resultatet av reserveringssystemet eftersom arbetet pågått i ett halvår och under tiden som jag har lärt mig PHP så har jag kommit till bättre och smidigare lösningar som implementerats. Därför kommer jag endast att redogöra för de funktioner som blev gjorda och presentera vissa metoder som använts flitigt i utförandet.

2 TEORI

Reserveringssystemet är uppbyggt med ett antal olika programmeringspråk och metoder som alla kanske inte känner till. Detta kapitel kommer att presentera språken och beskriva de metoder som har använts vid programmering.

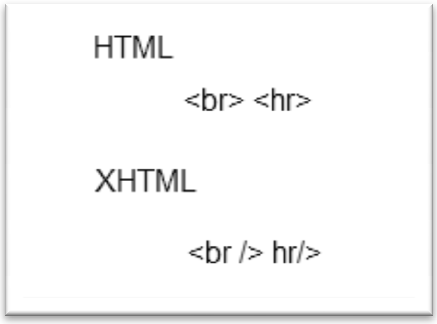
2.1 XHTML

Programmering av webbsidor sker vanligtvis med HTML *”Hypertext Markup Language”*. Detta är ett programmeringsspråk som använder sig av element för att berätta åt en webbläsare hur sida skall se ut. Ett element är uppbyggt med en start och stop –*tag*, som berättar om för webbläsaren att innehållet mellan dessa två skall tolkas på ett visst sätt.

XHTML står för *”Extensible Hypertext Markup Language”*, som är en striktare form av HTML. När man skriver XHTML så måste man vara noga med att allting skrivs på rätt sätt och vissa regler skall hållas. Vanlig HTML –kod kan skrivas mer eller mindre rätt men detta resulterar i att webbläsare måste tolka koden före den visas (Duckett, 2010, s.30-33). Med XHTML är koden redan rätt skriven så allt sker lite smidigare.

Regler för användning av XHTML är följande (w3schools, webbsida 2011):

- Elementen måste vara väl strukturerade.
- Elementen måste alltid avslutas.
- Elementen måste alltid vara i små bokstäver.
- Dokumentet måste ha ett root element.

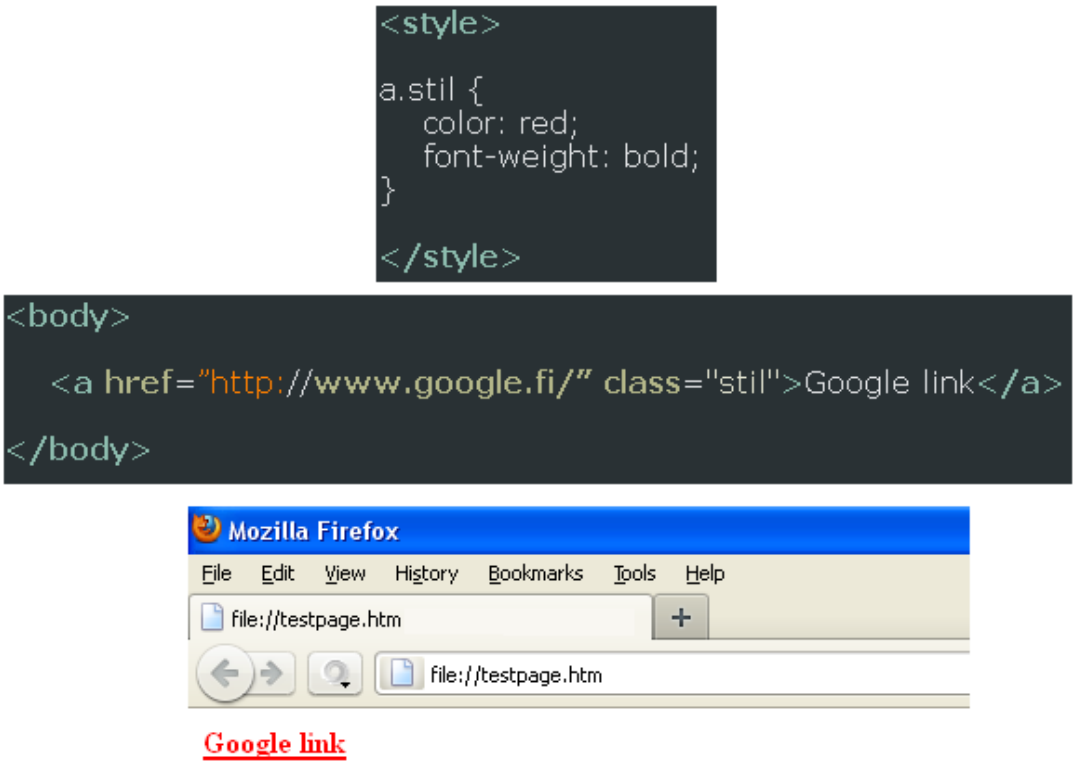


*Figur 1. Exempel på XHTMLs krav på avslutande av en tag.
 gör en ny rad och <hr> en linje.*

2.2 CSS

CSS är en förkortning av *”Cascading Style Sheets”* och används för att formatera HTML –element d.v.s. ge dem en viss design (Schmitt, 2008, s.15). Genom att använda CSS kan man komma åt alla HTML –element på en gång eller ett specifikt element åt gången. CSS kan användas till att både modifiera utseende och positionering av element, dessutom så erhålls stilen på underliggande element.

Man kan namnge olika element med id, namn eller klass vilket gör det möjligt att komma åt de olika elementen som man vill ge en viss stil. Varje HTML –element har ett utgångsvärde som används om inget har angivits. När ett CSS –dokument kopplas till ett HTML –dokument så överskrivs dessa inställningar. Läsningen av koden sker uppifrån ner, vilket resulterar i att CSS som läggs in efteråt överskrider i sin tur föregående CSS –kod. Detta är användbart när man använder sig av samma kod på flera sidor, men en specifik sida behöver något element modifierat.



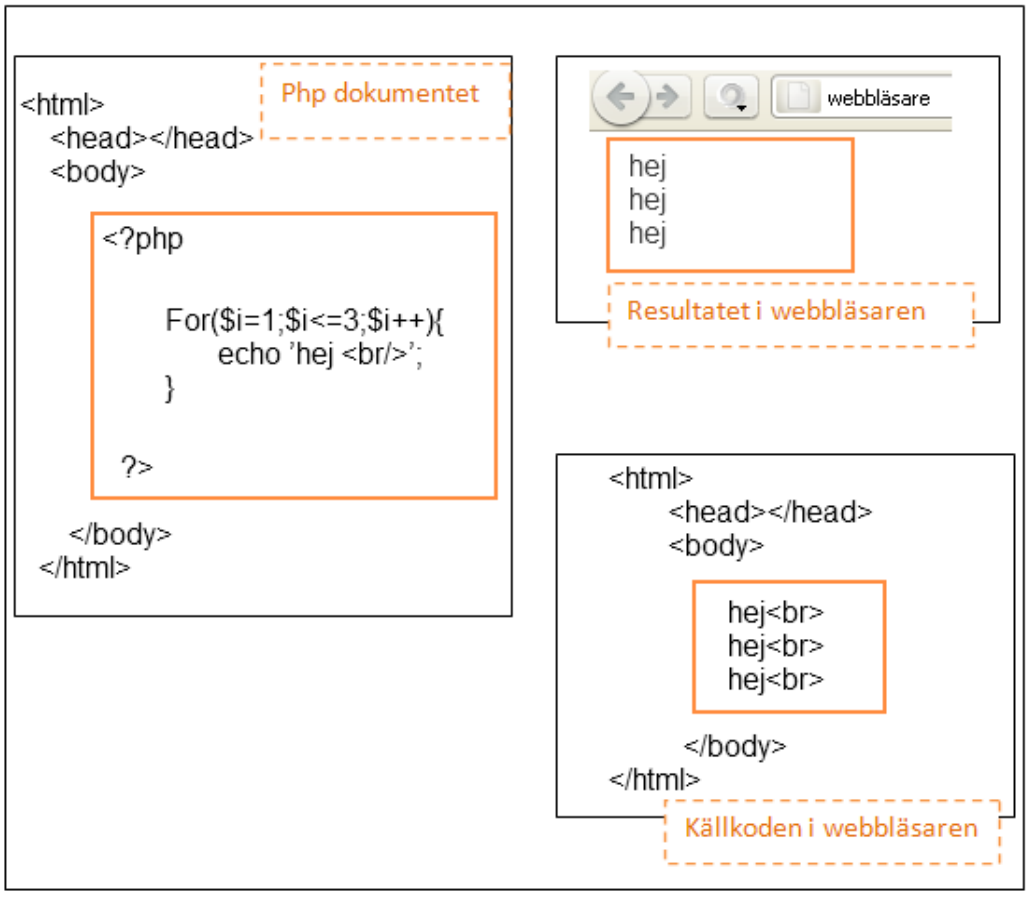
Figur 2. Exempel på CSS –kod och HTML –kod, samt resultatet av koden. Här omändras ett a –element som är en link till google. I mittersta rutan är koden som HTML och den innehåller en ”class” variabel som har namnet ”stil” som kopplas ihop med CSS a-elementet som har namnet ”stil”.

2.3 PHP

PHP är ett webb –utvecklingsspråk skrivet för webb –utvecklare av webb –utvecklare. PHP står i dagens läge för *”PHP: Hypertext Preprocessor”*, ett äldre namn är *”Personal Home Page Tools”*. PHP är ett programmeringsspråk som körs på en webbserver och används vanligtvis för att göra webb-applikationer i kombination med en webb –server så som Apache (Suehring, Converse & Park, s.3).

När en användare öppnar en webbsida så är det webbservern som tar emot informationen och därmed genom användning av PHP producerar HTML –koden som webbläsaren sedan tolkar och resultatet syns på användarens bildruta.

I figur 3 finns ett exempel på hur PHP –fungerar. I exemplet så är det en *for –loop* som tre gånger skriver ut hej i webbläsaren. Om man är bekant med HTML så ser man att det är frågan om en webbsida, det inringade kodstycket är PHP –kod som skriver ut *hej* samt HTML –elementet *”
”* som gör en ny rad. Om man högerklickar på en webbsida och väljer att se källkoden så syns resultatet på samma ställe som i PHP –dokumentet men endast resultatet av loopen kommer fram dvs. hej
. PHP –koden är dold för användaren.



Figur 3 – Förklarning av PHP gjord kod.

2.4 MySQL

MySQL är ett open source system för hantering av SQL relations databaser (Suehring m.fl. 2009 s.44). SQL står för *”Structured Query Language”* och är ett instruktionsspråk för databaser. Genom att använda MySQL med PHP så har man möjlighet att från en webbsida hantera information som finns i databasen.

PHP innehåller funktioner för att ta kontakt med en MySQL –databas och för att utföra diverse tänkbara uppgifter. I figur 4 nedan så finns det ett exempel på SQL –språket för att ge en uppfattning om hur det ser ut. Man väljer en tabell och ur tabellen söker man efter en användare *”Kalle”* i kolumnen som heter *”Användar_Namn”*.

```
SELECT * FROM Användar_tabellen
WHERE Användar_Namn = ”Kalle”;
```

Figur 4. Exempel på SQL –språket.

2.5 JavaScript

Javascript är ett programmeringsspråk som i realtid följer med vad användaren gör i webbläsaren och finns till för att öka interaktiviteten på HTML –sidor (w3schools, webbsida 2011).

Under arbetets gång ville jag hålla mig till PHP, men för att göra systemet så användarvänligt som möjligt behövdes det också lite Javascript i form av en pop-up ruta som bekräftar borttagnignen av en användare. Att göra motsvarande funktion med PHP skulle ha blivit onödigt invecklat.

3 PLANERING

Detta kapitel kommer att presentera planeringen av systemet. Förutom kraven på systemet som kommer att presenteras så skulle systemet bli så användarvänligt och funktionellt som möjligt. Eftersom inlärningen av PHP har skett under arbetets gång så påbörjades arbetet från grunden, istället för att använda något befintligt system för t.ex. användarhantering. Även om exempel har använts så har dessa gåtts igenom och modifierats enligt behov.

3.1 Kravspecifikation

Beställarens krav på reserveringsystemet var följande:

- Användare skulle reservera en sittplats åt sig själv.
- Man skall som användare se vem som sitter på en viss sittplats.
- När man beställer en sittplats så hör den till personen tills han eller hon tar kontakt med en systemadministratör.
- Sittplatser skall kunna för-reserveras av administratören åt nio olika byar:
    - Harvio
    - Huhtervu
    - Ihala
    - Kalksalo
    - Kesvalahti
    - Kuhkaa
    - Kumola
    - Oinaanvaara
    - Tervajärvi
- Sittplatser som förreserveras av administratören åt vip –personer.
- Det skulle finnas 720 sittplatser.

3.2 Krav på funktioner

Alla funktioner som har gjorts har mer eller mindre utvecklats under arbetets gång. I början framstod dock vissa funktioner som skulle komma att behövas, för att bygga ett fullständigt system.

- Användare
    - Hantering av användare
    - Registrering av nya användare
    - Borttagning av användare ur databasen
    - Redigering av användare
    - Återhämtning av lösenord

- Sittplatserna
    - Framställning av sittplatserna i reserveringssystemet
    - Funktion för reservation av sittplatserna för användare
    - Funktion för förreservation av sittplatser för byar och vip
    - Funktion för att reservera sittplats åt användare som inte har tillgång till internet.

3.3 Användarkonto

En användare är en person som besöker kyrkan ofta och han eller hon kan förekomma i tre olika former:

- Vanliga användare syftar på personer som har tillgång till internet och gör därmed ett eget användarkonto genom att registrera si på sidan.
- Administrativa användare har naturligtvis tillgång till hela systemet och kan redigera användaruppgifter, sittplatser och reservationer.
- Användare utan internet som existerar endast genom en administrator och de här kontona kan inte användas till att logga in på sidan.

Användarkonton kan redigeras vid behov, vanliga användare kan göras till administratörer vilket gör det möjligt att ha flera administratörer och man kan ta bort administrativa rättigheter vid behov.

3.4 Användargränssnitt

Användargränssnittet gavs det inga kriteriet för, utan det utformades enligt följande krav för en funktionell och användbar webbsida:

- Lätt att navigera
- Funktionellt för innehållet
- Skall fungera i Mozilla Firefox och Internet Explorer
- Färgval som kan kopplas ihop med Lumivaara

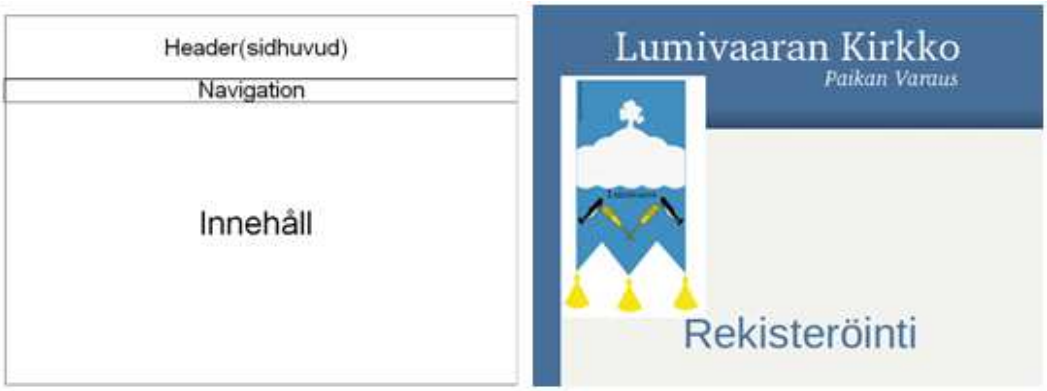
Den grundläggande tanken bakom användargränssnittet var att dela upp innehållet i delar som alla hade en bestämd bredd och var med relativ positionering i mitten av rutan, oberoende på storleken av besökarens webbläsare, därmed kunde höjden på de olika delarna bestämmas enligt behov.

Som bakgrundsfärgen för innehållet valdes en nedtonad vit färg, istället för att ha en vit färg som skulle ha varit väldigt skarp för ögat. Sedan valdes en blå färg som bakgrundsfärg, den baserar sig på Lumivaaras vapen som hade en ljusare blå färg som tonades ner.

Figur 5. Lumivaaras vapen.

För att få lite mer djup i gränssnittet så användes skuggor som ger en illusion av att innehållet ligger ovanför bakgrunden, vilket ger en stiligare helhet. En liten detalj med skuggan är också att knapparna i menyn som är passiva ser ut att ligga bakom innehållet. En logo gjordes också för att ge sidan någon sorts identitet, genom att med en ”finare” font och olika storlekar skriva ut ”Lumivaaran Kirkko – paikanvaraus”.

Navigationen gjordes vertikal eftersom det passade bättre på en sida som fungerar som ett reserveringssystem, därmed inbesparades utrymme för innehållet som speciellt vid själva sittplatsgränssnittet tog mycket utrymme.



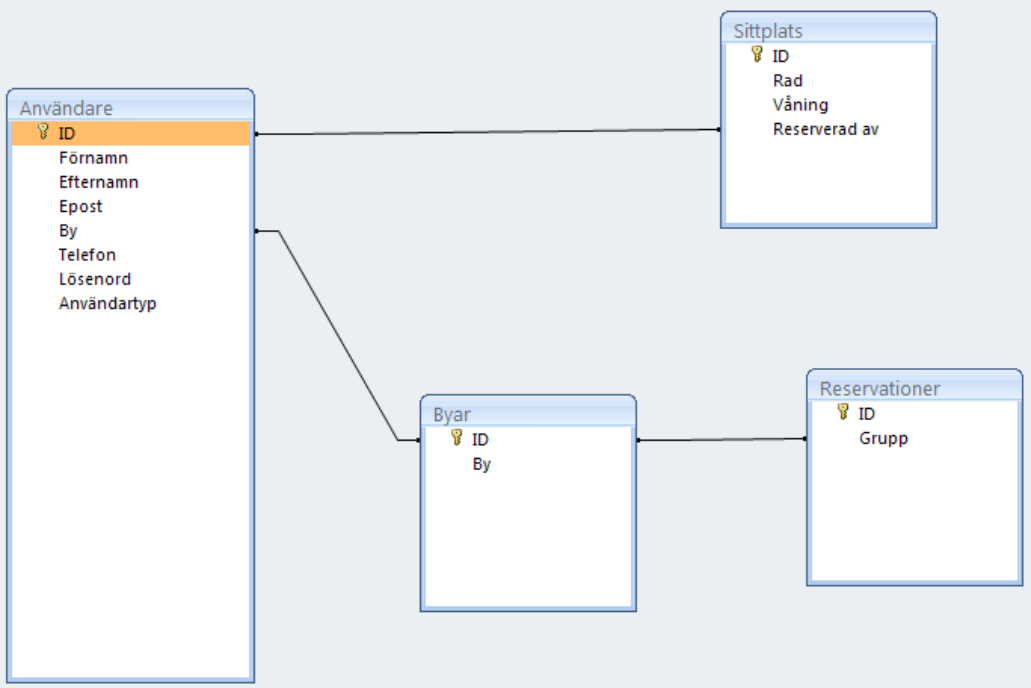
Figur 6. Grundiden för layouten och slutliga färgvalet.

3.5 Databasen

När man gör webb –applikationer eller sidor med PHP så har man för det mesta någon sort av system för att hantera användare. Det finns därför många färdiga alternativ som man kan fritt bygga på, men eftersom jag inte var någon van programmerare vid det här laget så bestämde jag mig för att bygga ett enkelt system själv. Därför gjordes användar –registret i databasen.

Det blev totalt fyra olika tabeller:

- Användare, innehåller alla användare
- Byar, innehåller alla byar. Det blev enklare att ha dem i en skilld tabell eftersom de används på flera olika ställen och dessutom gick det lätt att koppla ihop byarna till de andra tabellerna i databasen.
- Sittplatser, innehåller de upptagna sittplatserna. När en sittplats blir reserverad så görs en ny post med sittplatsens id dvs. tabellen innehåller inte alla sittplatser färdigt utan de görs vartefter de blir upptagna.
- Reserverade sittplatser, syftar på de sittplatser som är förreserverande åt olika grupper av administratörer.



Figur 7. Figur över databasens tabeller. Bilden är endast en överblick över hur tabellerna sitter ihop och vilket innehåll de har. På grund av datasäkerhetsskäl så har namnen på de olika tabellerna och innehållet omändrats.

4 GENOMFÖRANDE

4.1 Verktyg

Vardagligen så använder jag mig av Linux Mint 10 och arbetade därmed också med lärdomsprovet i den miljön. För att kontrollera att webbsidan fungerade som tänkt i Internet Explorer och Firefox så måste ändå Windows XP användas för att få tillgång till Explorer och samtidigt kunna göra omändringar så att sidan såg likadan ut i båda webbläsarna.

Mathias Renlund gav råd om att använda WAMP som webbserver, men eftersom jag använder Linux gällde det att hitta något annat alternativ. WAMP är en förkortning för en kombination av webbapplikationer, W syftar på Windows, medan A på Apache, M på MySQL och P för PHP/Perl/Python (Suehring m.fl. 2009, s.44). Som motsvarande lösning till Linux hittades LAMP som hade bytt namn till XAMPP.

XAMPP (XAMPP, webbsida 2011), är en Apache webbserver som färdigt innehåller MySQL, PHP och Perl. Den är gjord av Apache Friends som är ett icke-vinststrävande projekt, som grundades på våren 2002 av Kai 'Oswald' Seidler och Kay Vogelgesang (Apache Friends, webbsida 2011).

WAMP är en webb utvecklingsmiljö för Windows. Som gör det möjligt att göra webbapplikationer mer Apache, PHP och MySQL databas (Wampserver presentation, webbsida 2011).

Gedit är gnome miljöns officiella text editor (Gedit, webbsida 2011) och var färdigt installerat med mitt operativsystem. Dessutom sökte jag efter något Notepad++ liknande texteditor för programmering och resultatet pekade på att Gedit var det bästa alternativet.

Notepad++ är en gratis texteditor som stöder många olika programmeringsspråk (Notepad plus plus, webbsida 2011).

GIMP, står för '*GNU Image Manipulation Program'* (Gimp Intro, webbsida 2011). Det är ett bildhanteringsprogram med ett brett utbud av verktyg. Alla grafiska delar gjordes med detta program, samt kyrkans planritningar modifierades för att kunna använda dem i sittplatssystemet.

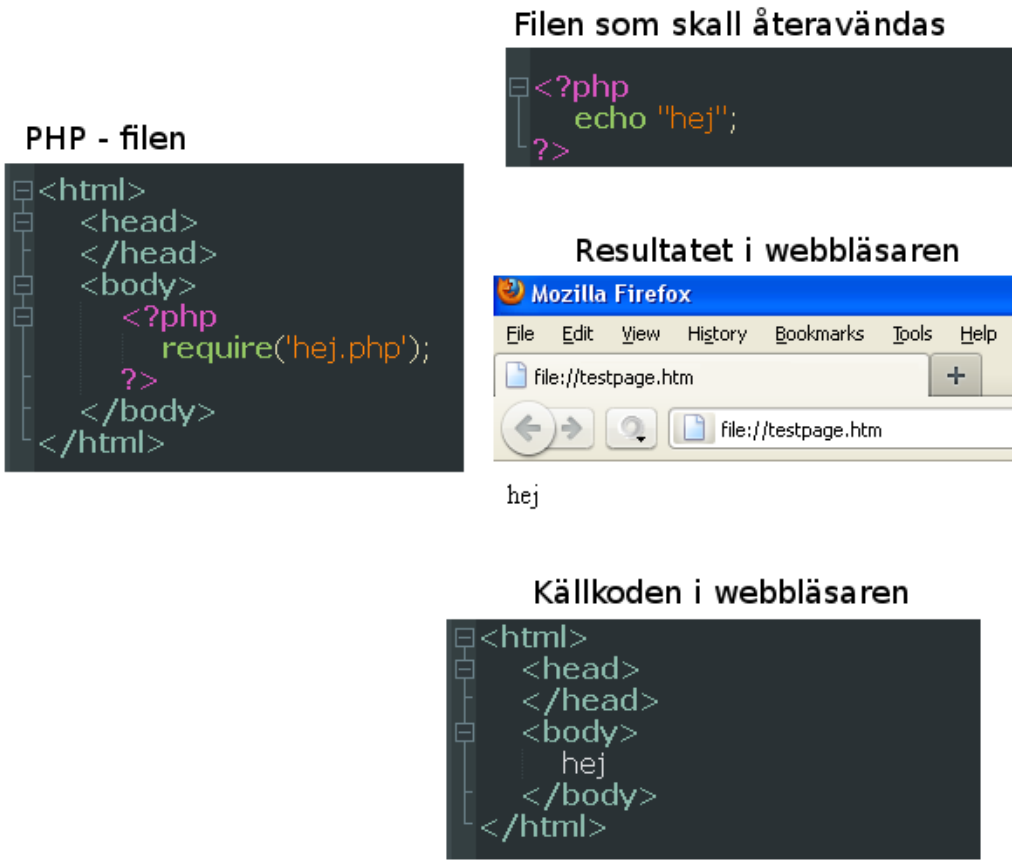
Test Mail Server Tool är en SMTP server emulator för att testa mail funktioner som man programmerar (Test Mail Server Tool, webbsida 2011). Det är svårt att testa mail funktionen utan att ha en egen server och diverse inställningar. Den kollar efter program som använder sig av en viss port för att skicka epost och snappar upp meddelandet och visar det genast i en ruta från Outlook.

4.2 Metoder

Detta kapitel kommer att presentera ett antal olika funktioner och metoder som använts vid tillverkning av systemet.

4.2.1 Require

PHP har en inbyggd funktion som kallar *require*, denna är släkt med en annan funktion som heter *include*. Det som de här två funktionerna har gemensamt är att när koden läses av servern så tas en annan fil med dvs. allt innehåll ur den utomstående filen kommer att ”skrivas” ut där require finns. Skillnaden mellan require och include är att require förutsätter att man skall ha tillgång till filen, om filen inte finns kommer programmet att avbrytas medan include körs fastän filen skulle utebli.



Figur 8. Exempel på PHP require funktionen

4.2.2 GET, POST och SESSION

Här är tre viktiga funktioner som finns inbyggda i PHP. De är alla till för att överföra information från en sida till en annan och de har olika sätt som lämpar sig för olika behov.

GET använder sig av webbsidans URL -adress genom att lägga till sitt värde på slutet av adressen. Sedan kan man hämta värdet från adressbalken i webbläsaren och använda det. Detta är dock synligt för användare och kan vara en säkerhetsrisk om man inte har tagit åtgärder emot manipulation av URL -adressen.

POST skickar däremot värdena osynligt och är bra när man skickar information som ingen skall ha tillgång till.

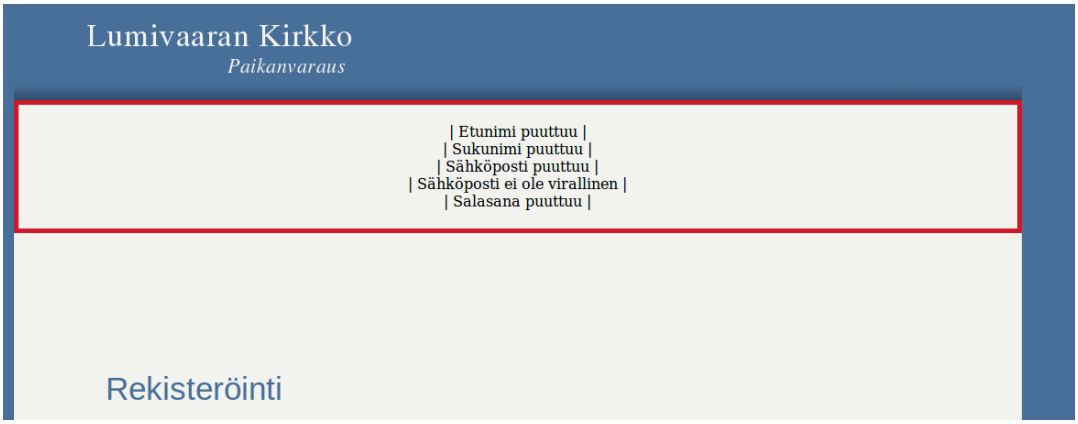
SESSION sparar värdena till servern och lagras en viss tid. De tas bort automatiskt efter att tiden har gått ut eller så kan man programmera att sessionerna avslutas vid t.ex. utloggning.

4.2.3 HTML sida som funktion

Vid tillverkning av ett system för hantering av användare, så hittade jag ett exempel som hade all HTML -kod i en funktion, som användes flitigt. Genom att ha innehållet innanför funktionen så kunde man kolla de inmatade värdena på samma webbsida. I vanliga fall så skulle värdena skickas till en annan sida och sedan returneras, men istället så påkallas HTML -funktionen och resultatet blir att man kan sköta t.ex. genomgången av inmatningen på en och samma webbsida.

4.2.4 Felmeddelande

Detta är en mycket simpel funktion som består av en variabel och ett HTML -element. Felmeddelanden finns till för att berätta om för användare om fel vid inmatning t.ex. registrering av en ny användare. Om längden på lösenordet är för kort eller epost -adressen inte innehåller ett ”@” tecken. När ett fel uppstår så sparas denna i en variabel och felmeddelande funktionens uppgift är att visa en ruta med felmeddelandet. Rutan finns i koden men är inte synlig förutom om det har uppsått något fel.



Figur 9. Felmeddelande rutan. Exempel är felmeddelande vid registrering av användare.

4.2.5 Datasäkerhet

Det finns en del olika sätt som har använts för att göra systemet säkrare. POST har redan tagits fram vars uppgift var att skicka data osynligt, vilket är det bästa sättet att t.ex. skicka lösenord.

Vid inmatning av information har två olika funktioner använts som är till för att rengöra den inmatade informationen. Det finns något som kallas *'injection attacks'* som går ut på att i inmatningsrutor mata in programkod som påverkar körningen av operationer i systemet, för att få otillåten tillgång till systemet. Syftet med dessa två säkerhetsfunktioner är att ta bort HTML -kod och SQL -kod som kunde ha blivit inmatad i ett inmatningsfält t.ex. vid inloggning. Funktionerna finns inbyggda i PHP och heter *mysql_real_escape_string* som tar bort *SQL och htmlspecialchars* som tar bort HTML. Efter att dessa två funktioner har körts så återstår innehållet ur inmatnings strängen som den ska.

Användar lösenorden är krypterade med en annan PHP funktion *md5* som krypterar en sträng med md5 kryptering. Denna kryptering fungerar endast en väg dvs. ett krypterat lösenord kan inte bli okrypterat med någon funktion. När en användare registrerar sig själv på sidan sparas det angivna lösenordet i krypterad form och endast användaren i fråga känner till lösenordet. När användaren sedan loggar in så krypteras lösenordet före det jämförs med användarens lösenord i databasen.

En sista metod för att säkerställa att någon inte slipper att ställa till med problem är att kontrollera värden som inmatas dvs. att en siffra verkligen är en siffra och att siffran uppfyller vissa krav så som storlek.

4.2.6 IF och SWITCH –satser

När man programmerar använder man sig av test som antingen är sanna eller falska. Genom att göra dessa test så kan man berätta om för datorn vad som skall hända om testet ger ett sant eller falskt svar (Suehring, Converse & Park 2009, s.60). Det finns två sätt att testa dessa på.

Det första är IF -satser, som jämför om ett värde överensstämmer med ett förutsatt värde, ifall det stämmer fortsätter operationen på ett spår. Om det inte överensstämmer så händer det något annat. Ifall en IF-sats inte stämmer överens med de förutsatta värden så finns det ELSE -satser och som namnet tyder så är detta vad som händer när IF -satsen inte passar. Vid behov av mer fall så kan man använda sig av ELSE IF som fungerar likt IF -satser genom att ange ett värde som skall stämma överens med ett annat värde.

```php
<?php

if($språk === "svenska"){
    echo 'Hej på dig !';
}
else if($språk === "finska"){
    echo 'Moikka !';
}
else {
    echo 'Hello there !';
}

?>
```

Figur 10. If-sats som kollar i en variabeln språk innehåller svenska, finska eller något annat och resultatet skrivs ut som en hälsning på respektive språk.

Det andra sättet är SWITCH -satser, vilket är en sats som anger olika sorters *case* som syftar på resultatet av ett värde som skall kollas. SWITCH används likt IF -satser men det är mer passande till operationer som har ett större antal fall. Likt IF -satser så har också SWITCH ett läge ifall inget av de förutsatta reglerna överensstämmer med det förutsatta värdet och den heter *default.*

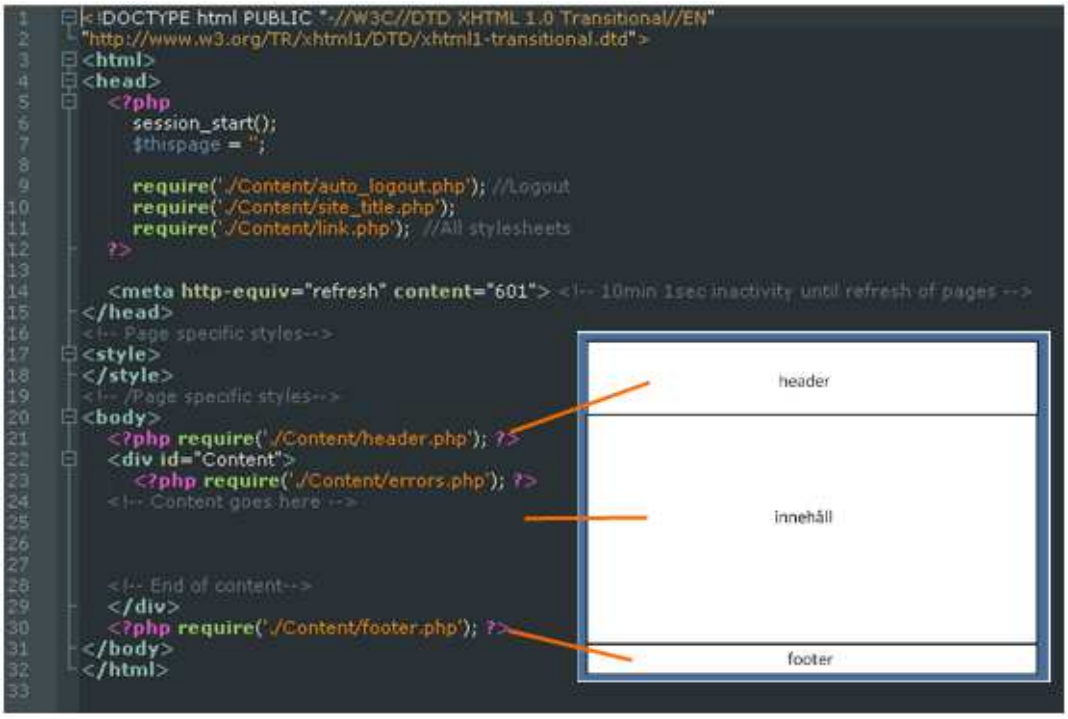
```php
<?php

    switch($språk){
        case 'svenska':
                echo 'Hej på dig!';
                break;

        case 'finska':
                echo 'Moikka !';
                break;

        default:
                echo 'Hello there !';
    }

?>
```

Figur 11. En switch –sats som kollar om språket är svenska, finska eller något annat och resultatet blir en hälsning på respektive språk.

4.3 Sidornas uppbyggnad

När man gör webbsidor så skall i princip alla sidor se likadana ut, detta kräver vanligtvis att man manuellt använder samma kod på alla sidor och ändrar om innehållet. När man har tillgång till en webbserver däremot så har man möjlighet att kringgå detta, genom att använda sig av *require* funktionen. Alla sidor har ett skelett bestående av filer som tas in med *require* som innehåller de delar som är likadana på alla sidor. Sedan kan man fritt mata in innehållet.

Det finns många sätt att lätt producera likadana sidor, men detta sättet fungerade helt bra. Genom att göra en mall -fil som innehållaner grundstrukturen så kan man spara tid på att inte skriva om sådan kod som skall finnas på varje webbsida.



Figur 12. Grunden för varje webbsida och en enkel överblick mellan layouten och koden.

4.3.1 Headern

Headern innehåller navigationsmenyn samt information om vem som är inloggad och en utloggnings knapp. Om man inte är inloggad så uteblir alla funktioner ur headern. Vid tillverkning av navigationen användes en guide (CSS DropDown Menu, Tutorial. webbsida 2011) som stegvis beskrev hur man skall med HTML elementet *list* göra en meny. Här användes också en *SWITCH* -funktion för att fålla reda på de olika knapparnas utseende, dels för att se om den inloggade användaren skulle ha tillgång till administrations-fliken, samt för indikation över vilken sida som var aktiv. Genom att ha en variabel på varje sida så håller man reda på vilken knapp som skall vara aktiv och namnet på knappen ändras enligt behov, så att förutbestämda CSS -stilar kan användas.

Information om den inloggade användaren kommer från en SESSION som startas vid inloggning. Detta är en liten detalj, men den används också av systemet för att veta vem informationen på sidan skall riktas till t.ex. den inloggade användarens sittplats har ett visst utseende för att urskiljas ur mängden.

Figur 13. Headern i utomstående vy.

Figur 14. Headern som inloggad användare.

Figur 15. Headern som inloggad administratör.

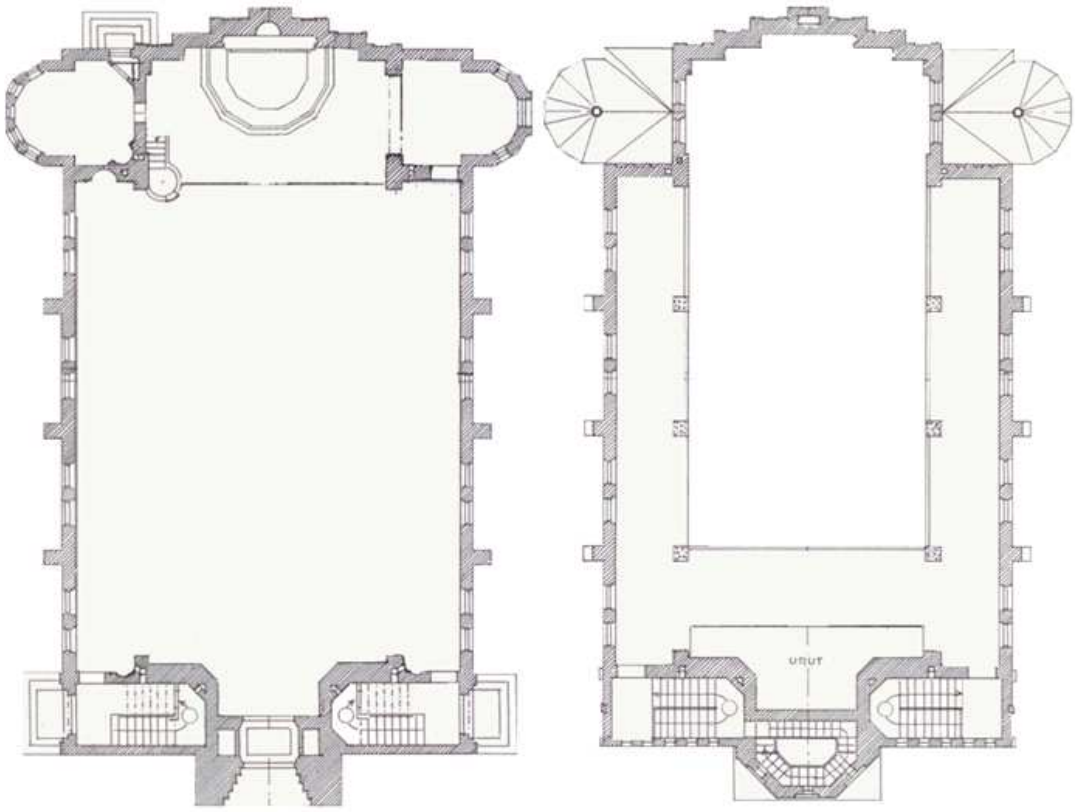
4.4 Sittplatserna

Detta kapitel kommer att redogöra för själva reserveringsgränssnittet och funktionerna som används. Det finns tre olika vyer av kyrkan i systemet som finns till för olika uppgifter och de använder alla samma funktioner medan resultatet blir annorlunda.

4.4.1 Planritningar

Vid början av lärdomsprovet skickade beställaren inskannade planritningar av kyrkan, dessa var handritade med utsatta bänkrader och anteckningar. För att kunna använda dem så måste de modifieras. Allt som inte behövdes målades över i GIMP med samma vita färg som valts till bakgrunden för innehållet. Efter att planritningarna för kyrksalen och läktaren var gjorda så modifierades storleken på dem för att få dem att rymmas i rutan för innehållet.

Antalet sittplatser som skulle finnas var 720 stycken uppdelat på salen och läktaren. Att komma underfund med antalet bänkar per rad var ganska tidskrävande. På läktarens planritning fanns det några bänkar som hade utsatta sittplatser och dessa användes för att få en referens för hur sittplatserna skulle rymmas. Antalet rader var utsatta på de ursprungliga planritningarna, så man hade åtminstone en startpunkt vid beräkning av sittplatser. Dessutom måste också beaktas att det fanns kortare rader och rader som hade stolpar. Sist och slutligen så passade allt in.



Figur 16. De modifierade planritningarna som används.

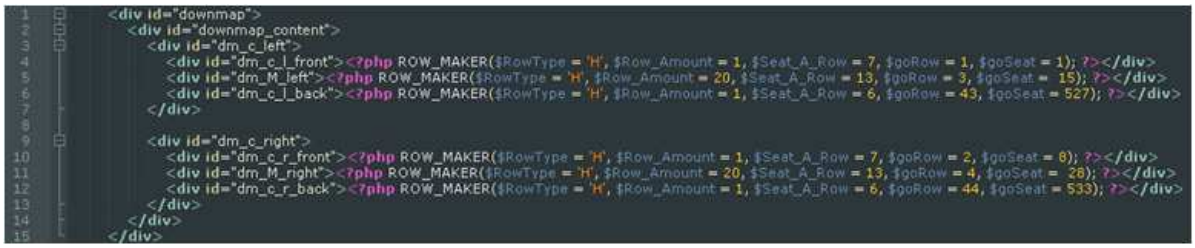
4.4.2 Utritning av sittplatser

Grundtanken med att rita ut sittplatserna fick jag som ett tips av Mathias, nämligen användning av *table* -elementet i HTML. Nästa steg var att med loopar rita ut sittplatser, vilket krävde en räknare av bänkrader för att hålla reda på vilken bänkrad som görs, samt en räknare för själva sittplatserna. Resultatet blev att man fick bänkrader precis som man ville ha dem och med lite CSS -stil så fick man dem att hållas till en viss storlek, vilket gjorde allt mer överskådligt.

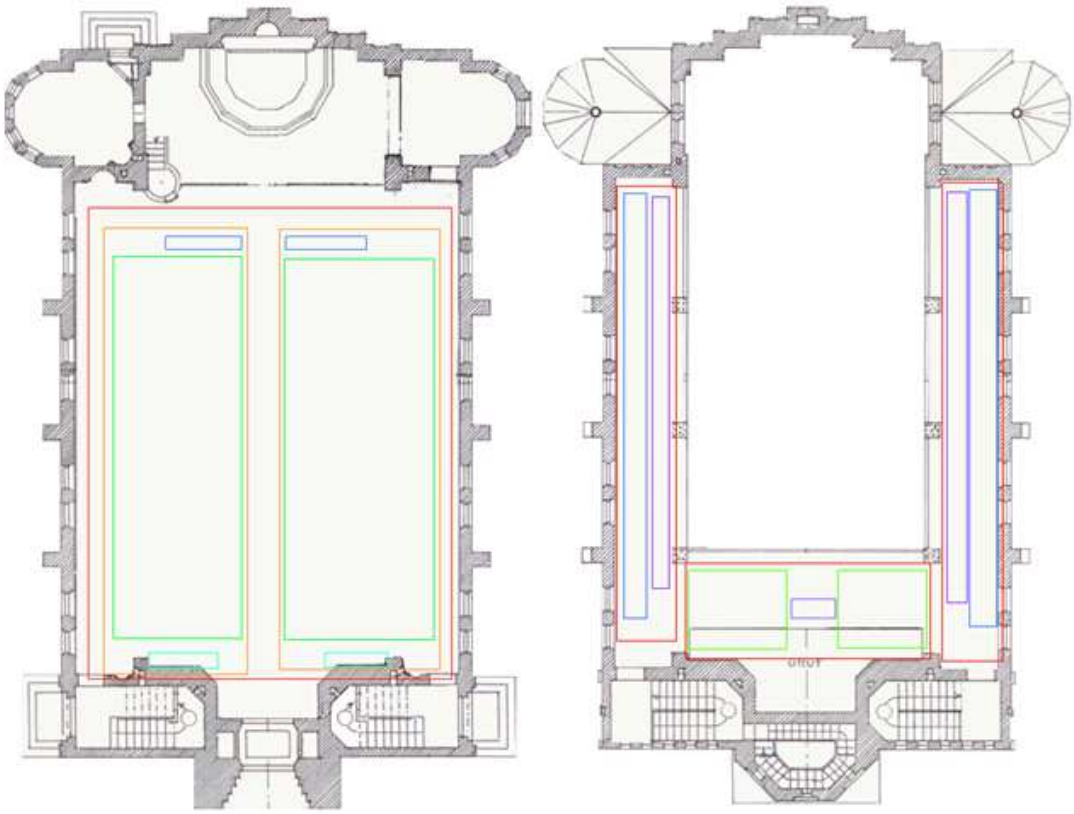
Det finns dock några kortare rader i kyrkan vilket krävde att de ritades ut skilt dvs. sittplatsutritningen hamnade in i en funktion som tog emot information om hur många bänkrader som skulle ritas ut, samt hur många bänkar per det skulle finnas.

Nästa steg var att på något sätt rita ut stolparna som finns i salen, detta gjordes genom att göra en fil innehållande alla stolpar enligt radnummer och bänkposition. När sittplatserna ritas ut så finns det en IF -sats som kollar om sittplatsen är en stolpe och ritar i så fall ut en stolpe istället för en sittplats.

Följande steg var att positionera sittplatserna i kyrkan så att de hölls innanför planritningen, detta kunde göras genom att ha div -element för varje grupp av sittplatser som byggde på planritningen. Detta kan föreställas som lager i ett bildhanteringsprogram dvs. understa lagret var planritningen som hade en ruta för sittplatserna, som i sin tur innehåll de skilda grupperna av bänkrader.



Figur 17. Filen innehållande layouten för sittplatserna dvs div-elementen. Detta är kyrksalen, en liknande fil finns för läktaren (Bilaga 2). Bilaga 1 är en förstorad bild av denna.



Figur 18. Sittplatsernas layout framställt grafiskt. Varje rektangel är ett div –element som anges i figur 17.

När utritningen av sittplatserna i kyrksalen var klar hade man en bra uppfattning om hur man skulle gå till väga med bänkarna på läktaren. Skillnaden här var dock att det fanns bänkrader som skulle vara vertikala, vilket gjordes genom att modifiera HTML *table* elementet så att endast en cell per rad ritades ut. Funktionen innehållande utritning av sittplatser blev då uppdelad med en IF -sats för horisontala och vertikala bänkrader.

Vid det här skedet var numreringen inte som man vill ha den, utan sittplatsernas numrering löpte från rad 1 sittplats 1 framåt och fortsatte sedan på följande sida, vilket resulterade i att de två raderna längst fram var rad 1 och 24, med respektive numrering av sittplatserna. För att få resultatet till rad 1 och 2 osv. så måste man ange en start -ID för sittplatserna och bänkraderna.

Till slut behövdes utritningen av stolpar modifieras pga. av den nya numreringen, eftersom bänkradernas id förändrades och ritades ut på fel ställe.

4.4.3 Sittplats funktionerna

För att inte behöva skriva om programkoden för varje sida som visar sittplatserna, så finns det tre olika funktioner som kopplas ihop med *require* till sidan som skall använda dem. Den första kallas *Row_Maker* och denna funktion tillkallas när man skall rita ut bänkrader. Den tar emot följande parametrar vid körning:

- RowType, typ aw rad dvs. vertikal eller horisontal rad.
- Row_Amount, antalet rader.
- Seat_A_Row, antalet sittplatser per rad.
- goRow, id som radnumreringen skall börja på.
- goSeat, id som sittplatsnumreringen skall börja på.

```
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 20, $Seat_A_Row = 13, $goRow = 3, $goSeat =  15); ?>
```

Figur 19. ROW_MAKER funktionens. I figur 17 så ser man att detta kodstycke finns mellan varje div –element.

Varje gång *Row_Maker* körs så tillkallas en annan funktion som heter *Seat_Maker* vars uppgift är att ta reda på vilken sorts sittplats det är frågan om. *Seat_Maker* tar emot radnummer och sittplatsnummer från *Row_Maker* och tar reda på om sittplatsen skall vara en pelare eller en sittplats, om resultatet är en sittplats så kollar funktionen ännu om sittplatsen är ledig för vem som helst eller reserverad för någon by eller vip-person. Efter det så utdelas klass -namn för motsvarande grupp, vilket ger koppling till en CSS -stil som används för den gruppen.

Det sista steget finns i funktionen *the_Seat*, vars uppgift är att placera ut den färdiga sittplatsen. Den tar emot information om sittplatsens nummer, radnummer och klass namnet för CSS -stilen som redan är vald. Funktionen kontrollerar vilka sittplatser som är upptagna och vilken av de upptagna sittplatserna tillhör den inloggade

personen, vilket påverkar utseendet på hur den slutgiltiga knappen skall se ut. Till slut så väljs knapparnas utseende beroende på sidan som skall ha dem. I reserveringsvyt är det en typ av knappar som används medan i översiktsvyet används mindre knappar och administrativavyt använder istället för knappar kryssrutor. När sittplatsen är utsatt så fortsätter loopen tills alla sittplatser är utritade.

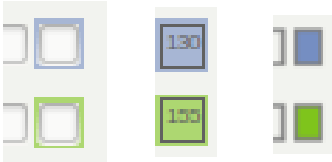
4.4.4 Sittplatsernas grafik

Gränssnittet för sittplatserna skulle vara så enkelt som möjligt, användes endast XHTML och CSS för att göra diverse sittplatser, istället för att använda bilder. Det besvärliga var att komma på ett system att ha de olika sittplatserna att ha ett visst utseende beroende på om sittplatsen är upptagen eller ledig, samt om sittplatsen är reserverad åt en grupp eller ifall sittplatsen är upptagen av den inloggade personen.

Lösningen för sittplatserna var att ha en bakgrund som ger mellanrum för knapparna och kan ha en färg som hänvisar till en förreserverad grupp. Och en knapp ovanpå som har en ram och specifik färg beroende på sittplatsen. I praktiken så gjordes detta genom att ha en fast storlek på cellen som innehöll sittplatsen och en fast storlek på div –elementet som var lite mindre än cellen för att få mellanrum mellan ramen på div –elementet och cellen. Skillnaden mellan de olika vyerna blev då att själva sittplatsen i cellen ändrades enligt behov t.ex. kryssrutor för administrativt vyläge.

Figur 20. Utseendet på sittplatserna i olika vylägen. Administrativt, reservations och översikts –vyläge.

Figur 21.Sittplatser i reservationsläge. 36 är en ledig plats, 37 är den inloggade personens sittplats och 38 är en upptagen sittplats.



Figur 22. Sittplatser som är reserverade för olika grupper, i olika vy lägen. Administrativ-, reserverings- och översiktsvy.

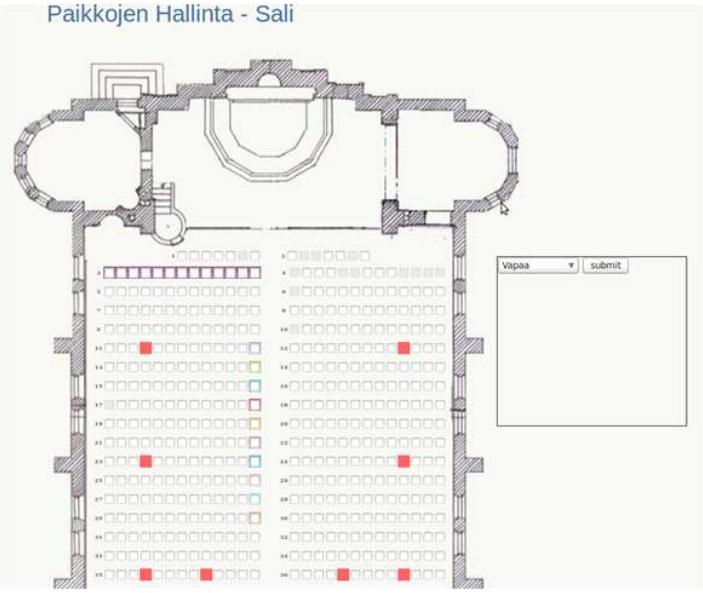
4.5 De olika vyerna

Detta kapitel kommer att presentera de olika vyerna som finns i systemet och som alla använder samma funktioner.

4.5.1 Administrativavyt

Det är logiskt att börja med det administrativavyt, eftersom det är här som bestäms vilka sittplatser som är reserverade för diverse olika grupper.

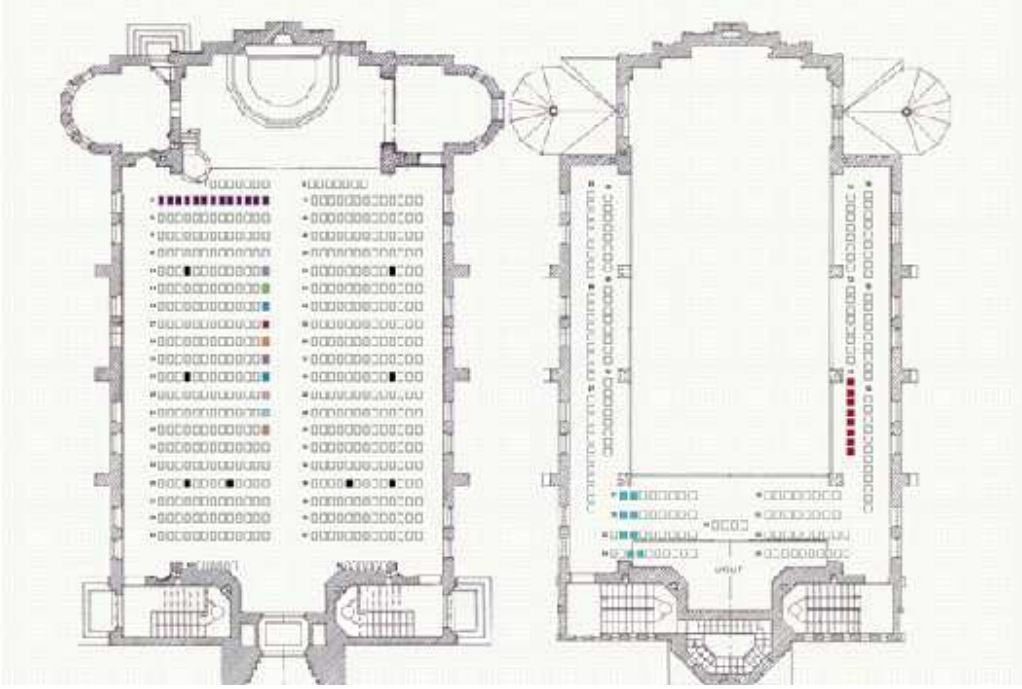
I detta vyläget så är sittplatserna representerade av kryssrutor, ifall de har färg som bakgrund är de redan reserverade för en viss grupp. Om kryssrutorna inte är aktiva så betyder det att en sittplats är upptagen och den måste frigöras före den kan reserveras åt en grupp. Huvudsakliga iden med detta vyläget är att en administrator kan kryssa i de sittplatser som skall reserveras åt en viss grupp och väljer sedan gruppen och klickar på knappen för att verkställa valet, vilket gör att de valda rutorna får en färg som är associerad med den valda gruppen.



Figur 23. Administrativa vyt av sittplatserna.

4.5.2 Översiktsvy

Översiktsvyn var till en början ett test för att klura ut diverse saker med positionering och numrering av sittplatser. När systemet blev klart så kunde översiktsvyläget användas som huvudsida för reserveringen med länkar till reserveringsvyna för kyrksalen och läktaren. Översiktsvyläget är det enda vyläget där man kan se hela kyrkan med sittplatserna på en gång. Här finns också färgerna som representerar de olika grupperna.



Figur 24. Översiktsvyläget.

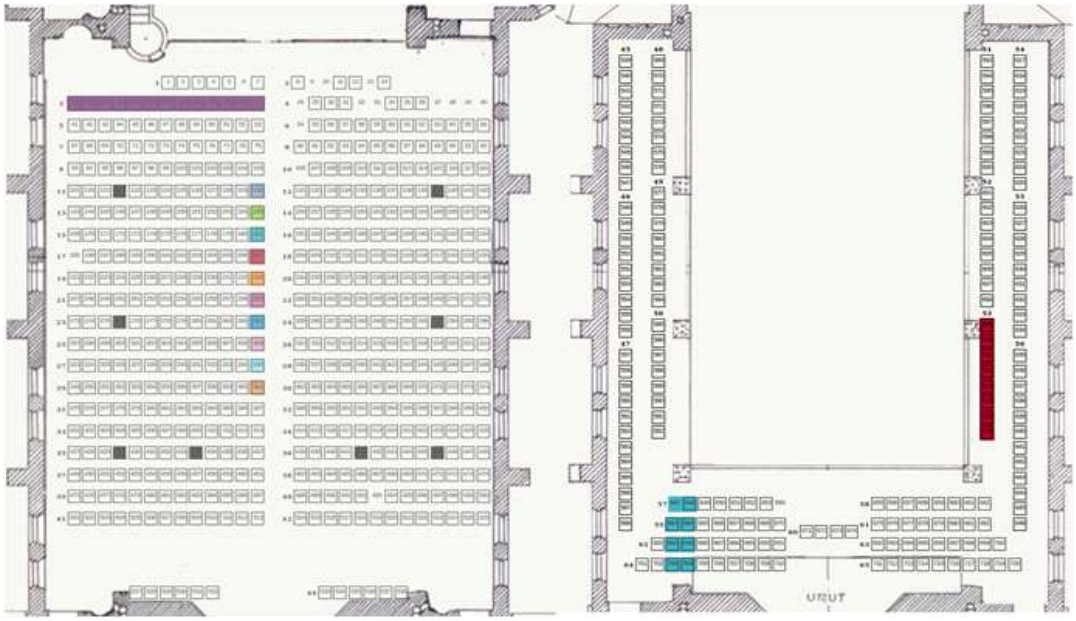
4.5.3 Reservationsvy

Reservationsvyläget är den huvudsakliga vyn som gör det möjligt för användare att reservera en sittplats åt sig själv.

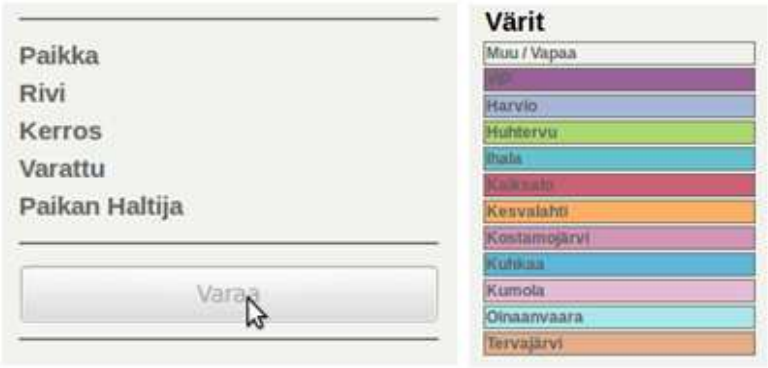
När man väljer en sittplats så finns det en ruta på högersida innehållande information om den, samt en beskrivning över de olika byarnas färger.

Vid reservering av en sittplats så klickar man på reservera och då ser man genast att bänkens utseende har förändrats för att hänvisa åt användaren att den här sittplatsen är din. Ifall användaren klickar på en by eller vip -reserverad sittplats och inte hör till byn i fråga, så är knappen för reservering oanvändbar, samt om användaren har redan den tillåtna mängden sittplatser.

Som administratör har man också en knapp för att frigöra sittplatsen samt möjlighet att reservera vilka sittplatser som helst oberoende av den grupp som är reserverad eller antalet sittplatser som administratören redan har.



Figur 25. Reservationsvyläget.

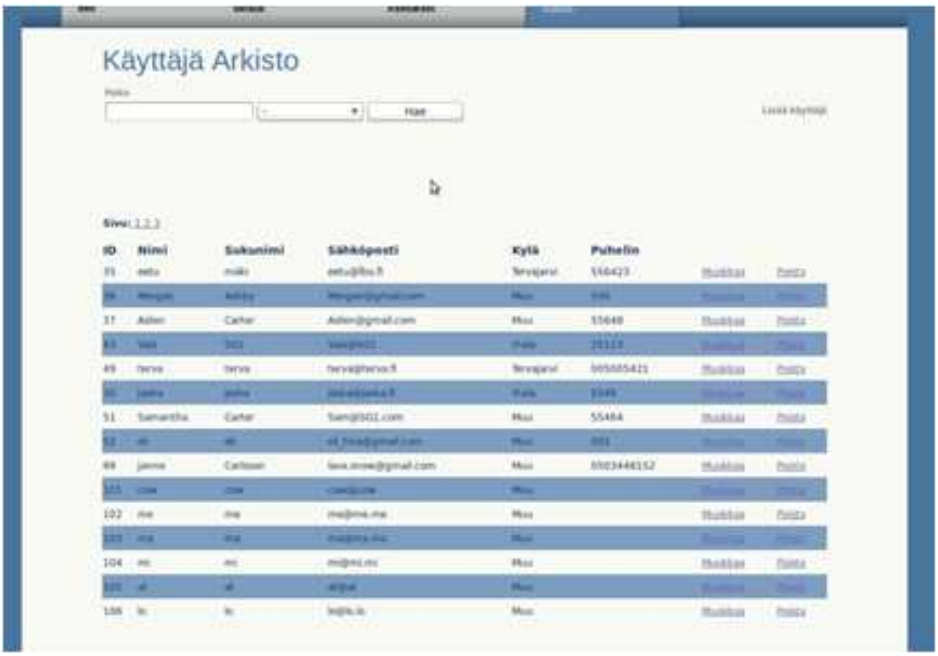


Figur 26. Reservations fönstret och färgerna för diverse grupperna.

4.6 Andra funktioner

4.6.1 Administration av användare

För att göra ett system för hantering av användare så användes följande guide som förklarar hur man skall göra ett PHP system (Basic PHP System, webbsida 2011). För att göra tabellen mer användarvänlig så användes en annan guide som förklarar hur man gör stiligare *table* -element (Top 10 CSS Table Designs, webbsida 2011). För att få CSS att fungera på rätt sätt dvs. varannan rad skulle ha en viss färg, så gjordes en variabel som i en IF -sats kollar om räknaren för tabellraderna är jämn eller ojämn och byter namnet på CSS klassen som skall användas.



Figur 27.Administrativ sida för hantering av användare.

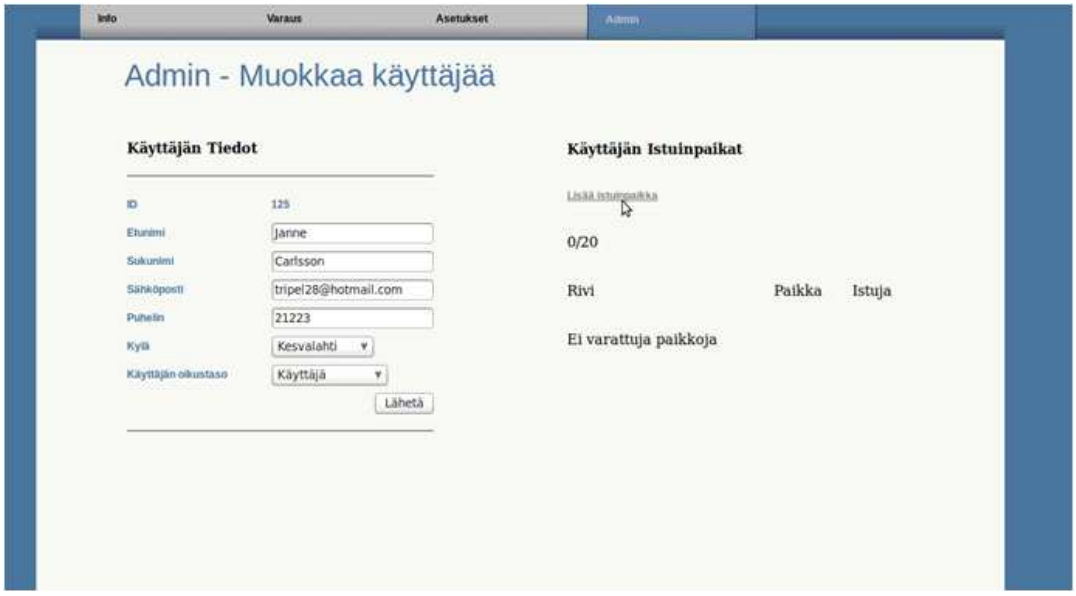
Det finns en enkel sökfunktion som man kan använda för att filtrera innehållet i tabellen, vilket gör det möjligt att snabbt hitta en specifik användare. Sökningen kan ske genom e-post, förnamn och efternamn. Man kan också välja att se alla användare, administratörer eller offline -användare.

id varje användare i tabellen finns det en knapp som tar bort användare och en annan som finns till för att redigera användaren. Vid bort tagning av användare så användes Javascript till att göra en pop-up ruta för att säkerställa valet om att ta bort användaren.

Här hittar man också en länk till att lägga till en offline -användare.

4.6.2 Redigering av användare

För att kunna hantera offline -användare så krävdes det något sätt att välja dem på och sedan också någon sorts metod för att reservera en sittplats åt dem. Detta blev också ett bra sätt att redigera andra användare, eftersom administratören kan lätt söka fram en användare och ser deras reserverade sittplatser, istället för att söka upp sittplatsen och sedan frigöra den i reserveringsvyläget.



Figur 28. Administrativ sida för redigering av användare.

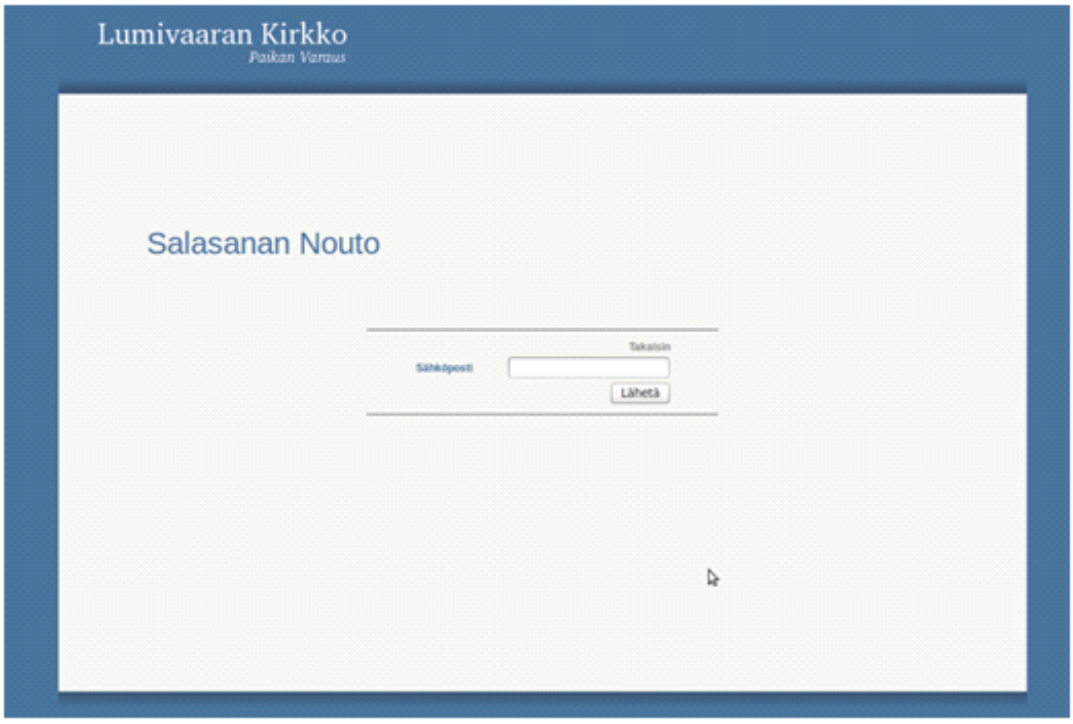
På redigerings sidan finns det en länk för att lägga till en sittplats, vilket startar en SESSION som används för att reservera en sittplats åt en användare. När man klickar på knappen tas man till översiktsvyläget var man kan välja sal eller läktare och när man reserverar en sittplats så reserveras den inte åt den inloggade administratören utan åt användaren som redigeras. Om man vill så kan man avbryta reservationsprocessen genom ett kryss bredvid namnet för personen man reserverar åt.

Användare har också tillgång till en personlig sida, där kan de se information om sitt eget användarkonto. Här har man möjlighet att se vilka sittplatser som är reserverade och man kan byta lösenord och telefonnummer.

Figur 29. Användarens personliga redigeringssida.

4.6.3 Lösenords återhämtning

Denna funktion hittar man via inloggnings sidan, den är till för att återhämta användarkonton om man har glömt bort sitt lösenord. För tillverkning av denna sida användes en guide som riktlinje för hur man skall göra en funktion för återhämtning av lösenord (Creating a PHP Password Reset Script, webbsida 2011).



Figur 30. Lösenords –återhämtnings –sida.

Funktionen kräver att användaren matar in sin email adress, om användaren med den adressen hittas i användarregistret så görs ett nytt lösenord som skickas med PHP-funktionen *mail* åt användaren och sparas i krypterad form i databasen. Användaren kan därefter logga in igen med det nya lösenordet och byta det mot ett eget lösenord.

4.6.4 Auto-Logout funktion

Den här funktionen är till för att automatiskt logga ut användare som inte har använt sidan på 10 minuter. Denna funktion användes (Session time out in PHP, webbsida-forum 2011), för att undvika använding av Javascript.

Den kräver att man startar en PHP SESSION när man loggar in, som håller reda på senaste klockslaget när användaren gjorde något på sidan. Varje gång användaren gör något så uppdateras klockslaget. När användaren inte har gjort något på 10 minuter så avslutas SESSIONEN och med hjälp av en meta -tag i HTML så uppdateras sidan automatiskt efter 10 minuter och vid det tillfället förflyttas den frånvarande användaren till inloggningssidan.

5 RESULTAT

Detta lärdomsprov har varit omfattande och resultatet blev ett fullständigt system för reservering av sittplatser gjort med PHP. Beställaren verkade också nöjd med resultatet och systemet skall tas i bruk.

Personligen så har jag lärt mig mycket om PHP -programmering och webbutveckling. Själv anser jag att jag har lyckats bra med att göra ett system som håller de krav som beställaren haft och jag har varit kreativ i utformningen av systemet.

KÄLLFÖRTECKNING

Trykta

Overgaard, Jörgen , Eriksson, Ulrika , Ek, Jesper 2004.
PHP 5 programmering. 1 upplagan. Sundbyberg. Pagina Förlags AB.
ISBN 91-636-0800-6.

Suehring, Steve , Converse, Tim , Park, Joyce 2009
PHP6 and MySQL 6 Bible.
ISBN 978-0-470-38450-3.

Schmitt, Christopher 2008
Professional CSS: Cascading Style Sheets for Web Design (2nd Edition)
ISBN 978-0-470-17708-2

Duckett, Jon 2010
HTML, XHTML, CSS and JavaScript
ISBN 978-0-470-54070-1

Elektroniska

XAMPP, webbsida 2011. Introduktion av XAMPP [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://www.apachefriends.org/en/xampp.html>

Apache Friends, webbsida 2011. Information om Apache Friends projektet [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://www.apachefriends.org/en/index.html>

Wampserver presentation, webbsida 2011. Wampservers webbsida [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://www.wampserver.com/en/presentation.php>

Gedit, webbsida 2011. Gedits webbsida [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL: http://projects.gnome.org/gedit/>

Notepad plus plus, webbsida 2011. Notedpad++ webbsida [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://notepad-plus-plus.org/>

Gimp Intro, webbsida 2011. Introduktion av Gimp [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://www.gimp.org/about/introduction.html>

Test Mail Server Tool, webbsida 2011. Test Mail Server Tools webbplats [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://www.toolheap.com/test-mail-server-tool/>

CSS DropDown Menu Tutorial, webbsida 2011. Dropdown meny exempel [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://ago.tanfa.co.uk/css/examples/menu/tutorial-h.html/>

Basic PHP System: View/Edit/Delete/Add Records, webbsida 2011.Guide till uppbyggnad av ett enkelt PHP system [online].[hänvisning 3.6.2011].Tillgänglig i form av www-dokument:
<URL: http://www.killersites.com/community/index.php?/topic/1969-basic-php-system-vieweditdeleteadd-records/>

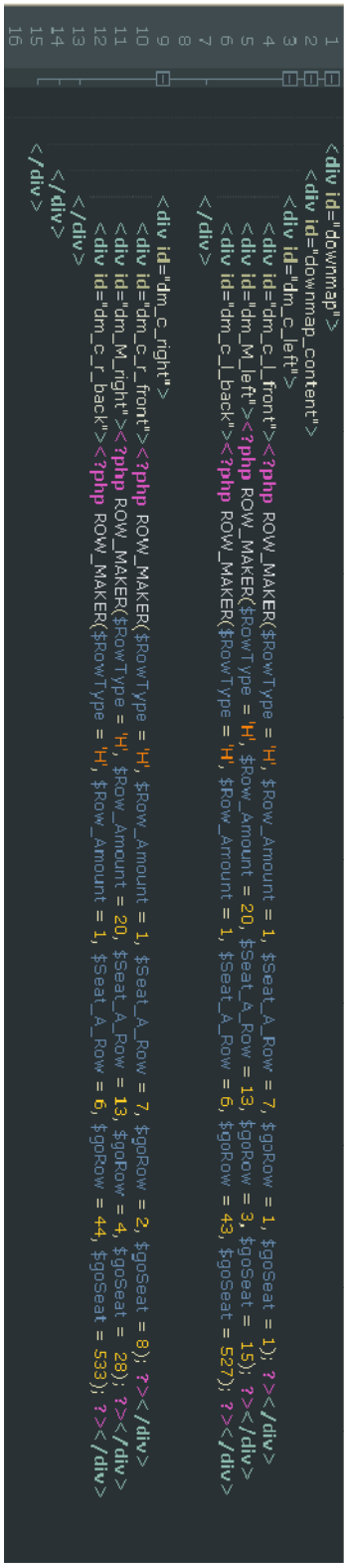
Top 10 CSS Table Designs, webbsida 2011. Guide till design av html elementet table [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://www.smashingmagazine.com/2008/08/13/top-10-css-table-designs/>

Creating a PHP Password Reset Script, webbsida 2011. Guide till lösenords återhämtning [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://www.danbriant.com/general/creating-php-password-reset-script/>

Session time out in PHP, webbsida-forum 2011. Automatisk utloggnings funktion [online].[hänvisning 3.6.2011]. Tillgänglig i form av www-dokument:
<URL:http://www.daniweb.com/web-development/php/threads/124500>

BILAGOR

Bilaga 1. Layout av kyrksalen i kodform



```
<div id="downmap">
    <div id="downmap_content">
        <div id="dm_c_left">
            <div id="dm_c_l_front"><?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 7, $goRow = 1, $goSeat = 1); ?></div>
            <div id="dm_M_left"><?php ROW_MAKER($RowType = 'H', $Row_Amount = 20, $Seat_A_Row = 13, $goRow = 3, $goSeat = 15); ?></div>
            <div id="dm_c_l_back"><?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 6, $goRow = 43, $goSeat = 527); ?></div>
        </div>
        <div id="dm_c_right">
            <div id="dm_c_r_front"><?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 7, $goRow = 2, $goSeat = 8); ?></div>
            <div id="dm_M_right"><?php ROW_MAKER($RowType = 'H', $Row_Amount = 20, $Seat_A_Row = 13, $goRow = 4, $goSeat = 28); ?></div>
            <div id="dm_c_r_back"><?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 6, $goRow = 44, $goSeat = 533); ?></div>
        </div>
    </div>
</div>
```

Bilaga 2. Layout av kyrkläktaren i kodform

```
<div id="upmap">
<div id="upmap_content">
<div id="um_c_left">
<div id="um_c_l_out">
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 9, $goRow = 45, $goSeat = 539); ?>
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 9, $goRow = 46, $goSeat = 548); ?>
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 12, $goRow = 47, $goSeat = 557); ?>
</div>
<div id="um_c_l_in">
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 48, $goSeat = 569); ?>
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 49, $goSeat = 577); ?>
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 50, $goSeat = 585); ?>
</div>
<div id="um_c_right">
<div id="um_c_r_in">
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 51, $goSeat = 593); ?>
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 52, $goSeat = 601); ?>
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 53, $goSeat = 609); ?>
</div>
<div id="um_c_r_out">
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 9, $goRow = 54, $goSeat = 617); ?>
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 9, $goRow = 55, $goSeat = 626); ?>
<?php ROW_MAKER($RowType = 'V', $Row_Amount = 1, $Seat_A_Row = 12, $goRow = 56, $goSeat = 635); ?>
</div>
</div>
<div id="um_c_middle">
<div id="um_c_m_left">
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 57, $goSeat = 647); ?>
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 59, $goSeat = 663); ?>
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 9, $goRow = 62, $goSeat = 683); ?>
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 10, $goRow = 64, $goSeat = 701); ?>
</div>
<div id="um_c_m_middle">
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 4, $goRow = 60, $goSeat = 671); ?>
</div>
<div id="um_c_m_right">
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 58, $goSeat = 655); ?>
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 8, $goRow = 61, $goSeat = 675); ?>
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 9, $goRow = 63, $goSeat = 692); ?>
<?php ROW_MAKER($RowType = 'H', $Row_Amount = 1, $Seat_A_Row = 10, $goRow = 65, $goSeat = 711); ?>
</div>
</div>
</div>
```