Datavetenskap

Anders Jansson och Jonas Olsson

Besökssystemet

En databas för besökshantering

Examensarbete, C-nivå

Pubnum 2004:02

Besökssystemet

En databas för besökshantering

Anders Jansson och Jonas Olsson

Den här rapporten är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i den här rapporten, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Anders Jansson

Jonas Olsson

Handledare: Thijs Holleboom

Examinator: Stefan Lindskog

Sammanfattning

Vi har under höstterminen 2003 utfört ett examensarbete åt Proplate IT AB. Vår uppgift har varit att koda om ett redan befintligt besökssystem, men behålla dess funktionalitet. Besökssystemet uppfyller idag de flesta behov hos kunderna men har många brister i kod och struktur. Att göra en enkel ändring i det gamla systemet tog alldeles för lång tid och resultatet blev sällan bra. Det första vi gjorde när vi kom till Proplate IT i Karlskoga var att undersöka och förstå oss på det befintliga systemet. Vi gjorde också en tidsplan för hur vi skulle lägga upp vårat arbete. När vi hade samlat så mycket information att vi förstod oss på systemet bra gick vi igenom alla sidor och diskuterade fram vilka ändringar som skulle behöva göras. Vi hade fria händer vad det gällde designen av utseendet. Det hade vi eftersom de ville ha ett grundsystem som enkelt kan anpassas till individuella kunders behov. Då är det viktigaste att koden är lätt att uppdatera i efterhand. All text och samtliga knappar lades genomgående inom tabeller för att få en bra struktur. När vi kände oss nöjda med designen av gränssnittet gick vi vidare till databasdesignen. Vi visste vilka data som skulle komma att behöva lagras, så den övergripande designen gick fort. Det var svårt att få ner redundansen i databasen eftersom systemet tillåter användaren att mata in text nästan hur som helst. När vi hade fått till en databasdesign som företaget gav sitt godkännande till började vi med att koppla ihop gränssnittet med databasen. Här skulle vi få en fungerande koppling mot både Access och SQL Server. Det här var det som tog mest tid men det hade vi räknat med i vår tidsplanering. Arbetet har fortlöpt utan några större problem eller svårigheter. Vi har lagt ner lite mer tid än vad som krävs av kursplanen. Det här har bara varit roligt då vi har trivts bra på arbetsplatsen och arbetet har varit givande.

The visitor system

A database for handling visitors

Abstract

During autumn 2003 we performed a bachelor project for Proplate IT AB. Our task was to remake an already existing system that handles visitors. The goal was to keep the old systems functionality but rewrite all code. The old system fulfills a requirement on today's market but lacks in code structure and is therefore hard to maintain. The first thing we did when we arrived at Proplate was to examine the old system. We also made a time plan on how we should accomplish the job. When we had gathered enough information about the system and understood how it worked, we discussed all the necessary changes that needed to be done. Concerning the user interface we could do as we liked because the main concern wasn't to have a nice looking user interface but to have an easy system for future modifications. When we felt satisfied with the user interface we began to work on the database. We knew which data that should be stored in the database so the design part went fairly quick. It was hard to keep a low redundancy in the database because the system allows the user to enter data in an uncontrolled way. After we had the database design that the company gave their approval of, we went on to work on the connection between the user interface and the database. This part of the job was the most time consuming since we needed the system to work on both Access and SQL Server. During the bachelor project, no major problems or difficulties have arisen. The time we have put into the project is perhaps a little bit more then the time required by the syllabus. We had a great time at Proplate.

Innehållsförteckning

Figurförteckning

1 Inledning

Här beskriver vi lite mer ingående om företaget vi har utfört arbetet åt och vi beskriver också i stora drag vad uppgiften gått ut på. Uppgiften har vi delat upp i mindre delmoment. De här delmomenten kommer vi att ta upp och beskriva mer ingående längre fram.

1.1 Bakgrund

1.1.1 Proplate IT AB

Proplate IT AB ingår i Proplate-gruppen som består av följande företag:

- Proplate AB - moderbolag
- Proplate IT AB - Dotterbolag till Proplate AB
- Smidesbolaget Oxelösund AB - Dotterbolag till Proplate AB
- Projektledarna i karlstad AB - Intressebolag till Proplate AB

Proplate IT´s affärsidé är att vara den kompletta IT-partnern för företag och organisationer i region Östra Värmland där Proplate IT ska vara det professionella förstahandsvalet som leverantör av teknisk IT-support och hårdvara.

1.2 Examensarbetet

Proplate IT AB har övertagit förvaltningen av ett system – "Besökssytemet" – som utvecklades av en personalman på Bofors Defence AB. Systemet uppfyller väl ett behov på marknaden. Funktionaliteten i systemet är bra men har mycket i övrigt att önska vad det gäller strukturering av kod och databasmodell. Det är idag svårt att utöka systemet med nya funktioner. Koden och databasmodellen är rörig och ostrukturerad. Systemet är webbaserat och programmerat mot Access databas.

Vi har fått till uppgift att skriva om hela besökssystemet och behålla dess funktionalitet. Systemet ska också kunna köras mot SQL server. Det är också ett krav att koden ska vara lätt att uppdatera och bygga ut i framtiden.

1.2.1 Vår Uppgift

Systemanalys av det gamla systemet

- Förstå sig på det redan befintliga systemet.
- Upptäcka de detaljer som kan förbättras.

Design av gränssnitt

- Dra riktlinjer för hur det nya systemet ska se ut.
- Bygga skalen för varje enskild sida efter våra riktlinjer.

Utforma databassystemet

- Rita databasmodell
- Implementera databasmodellen i SQL server och Microsoft Access.
- Undersökning/testning/inläsning
- Koppla logiken från gränssnittet till databasen.

2 Analys av det gamla systemet

Här beskriver vi arbetet med hur vi analyserade och kom fram till de förändringar vi ville göra till det nya besökssystemet. Vi utgick ifrån vår kravspecifikation och testade sedan det gamla systemet för att i detalj se hur det fungerade. Där efter tog vi fram vad vi ville förändra.

2.1 Kravspecifikation

Vi har fått till uppgift att skriva om hela besökssystemet och behålla dess funktionalitet. Ingen gammal kod eller databas kommer att tas i bruk, utan allt ska göras om från början. Koden som ska skrivas ska vara enkel och utan onödiga extra detaljer. Det här görs för att det i framtiden ska vara lättare att lägga till och ta bort funktionalitet ur systemet. Systemet har tidigare endast körts mot Access databas men ska i den nya versionen kunna köras både mot Access och SQL Server databas.

I frågan om designen av gränssnittet i systemet har vi fått fria händer. Det slutliga systemet som säljs till kund kommer att modifieras efter deras behov och därmed få ett unikt utseende. Därför ska vi fokusera på att producera välskriven kod och att skapa en bra databas för att systemet ska bli lättare att uppdatera i framtiden. Naturligtvis kommer vi även att göra vårt bästa för att få till en ren och snygg design.

2.2 Utvärdering av det gamla systemet

Vi undersökte det gamla systemet i detalj och lärde oss hur det fungerade. När man tog en närmare titt på den bakomliggande koden såg man att indenteringen var väldigt bristfällig. Det fanns ingen logik bakom döpningen av variabler vad vi kunde se. En annan sak var avsaknaden av att återanvända kod. Likadan kod kunde återkomma på sida efter sida vilket är totalt onödigt. Det fanns till exempel en ”redigera besök” sida som var en kopia av sidan ”besöksanmälan”. Det medför ett väldigt extraarbete om man ska behöva ändra på varje sida så fort man vill göra en förändring. Vi tog också en titt på databasen och den led lite av samma problem när det gällde namnsättning och lagring av data. Det var till exempel många saker som lagrades dubbelt. När vi nu visste mer om systemet började vi också få en klarare uppfattning om vad vi tyckte behövde förändras i gränssnittet.

Vår uppfattning var att det skulle behövas ett mer stilrent och enkelt gränssnitt utan en massa onödiga finesser. Vi ville också att det skulle vara mer logiskt och lättanvänt. Alla sidorna borde också ha ett likadant utseende.

När vi funderade på vad som behövdes ändras i gränssnittet i det gamla systemet ställde vi oss genomgående följande frågor:

- Vad vill jag göra i det här formuläret?
- Förstår jag vad jag ska göra i det här formuläret?
- I vilken ordning vill jag fylla i uppgifterna?
- Är det här en logisk följd av inmatningsfälten?
- Är designen genomgående stilren, enkel och följer de övriga sidornas design?

2.3 Övergripande funktionalitet

Systemet används av de anställda på företaget. Vilka sidor de anställda har tillgång till beror på deras behörighet. Det finns tre nivåer för dem som har rätt att använda systemet:

Nivå 1: Normalanvändaren som har tillgång till ”nytt besök” (2.4.4) och ”rullande information” (2.4.2).

Nivå 2: Receptionisten som har tillgång till ”Reception” (2.4.3) och de sidor som ingår på nivå ett.

Nivå 3: Säkerhetschefen som har tillgång till ”Godkänn besök” (2.4.5) och ”Historik” (2.4.6) och de sidor som ingår på nivå två och nivå tre.

Normalanvändaren använder systemet till att anmäla kommande besökare. Receptionisten använder systemet för att se vilka besökare som ska komma. Säkerhetschefen använder systemet till att godkänna kommande besökare och har även tillgång till all historik.

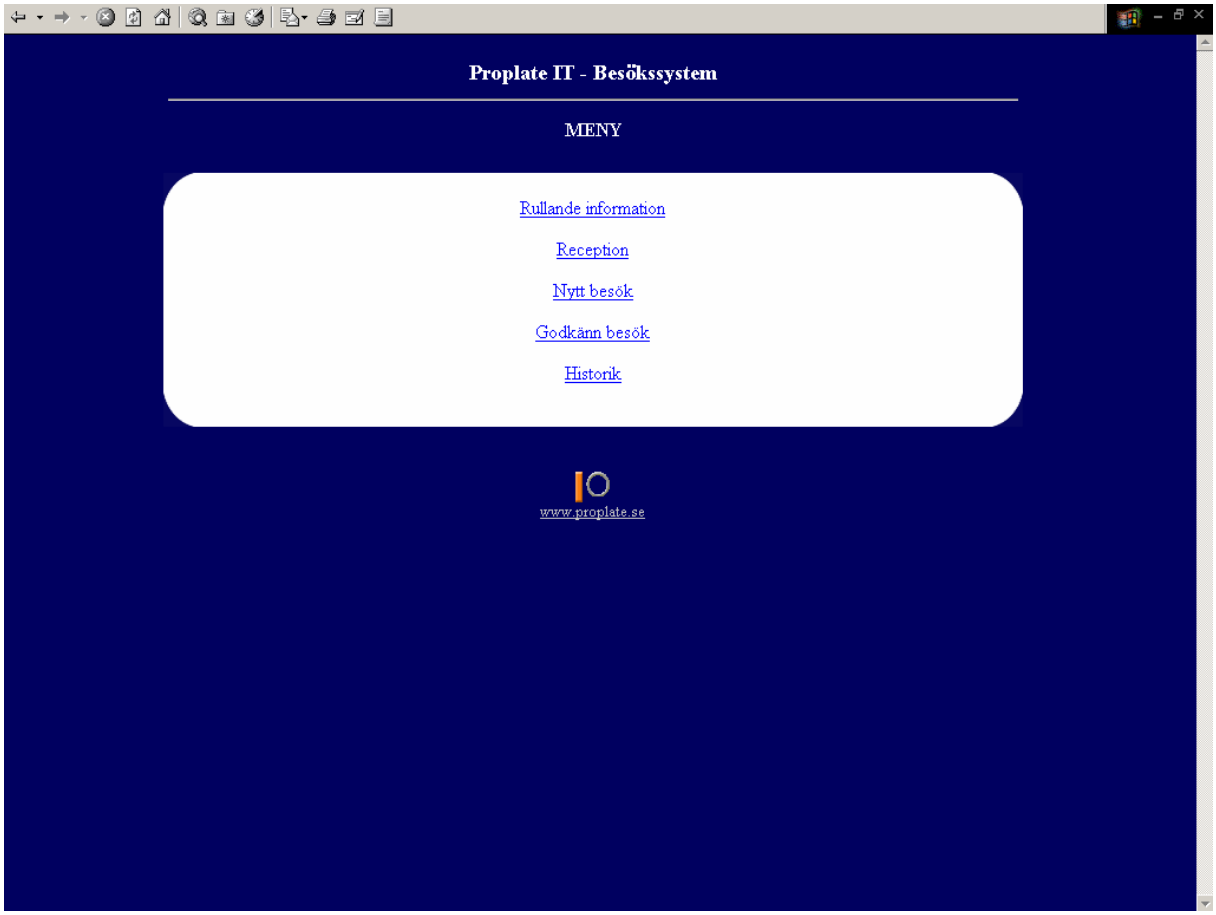
2.4 Gränssnitten

I våra undersökningar av det gamla systemet upptäckte vi en del detaljer vi ville förändra. Det resulterade i att vi för varje sida hade en hel del anteckningar om vad vi ville förändra. De här anteckningarna är nyttiga inför det kommande arbetet när vi ska skapa gränssnitten till det nya systemet. Här följer en beskrivning av de olika sidorna i det gamla besökssystemet och vad vi vill förändra.

2.4.1 Meny

Den här menyn (Figur 2-1) användes inte alls i det gamla systemet förutom vid demonstration. Användarna lade istället till de sidor de hade behörighet till under ”favoriter” i Internet Explorer (direktlänk till varje sida).

Figur 2-1 Meny



2.4.2 Rullande information

Den här sidan (Figur 2-2) användes för att välkomna dagens besökare. Besökarna rullar nedifrån och upp. Det kommer inte att ske några förändringar av den här sidan. Önskemålet är att den här sidan ska behållas och fungera tillsammans med det nya systemet.

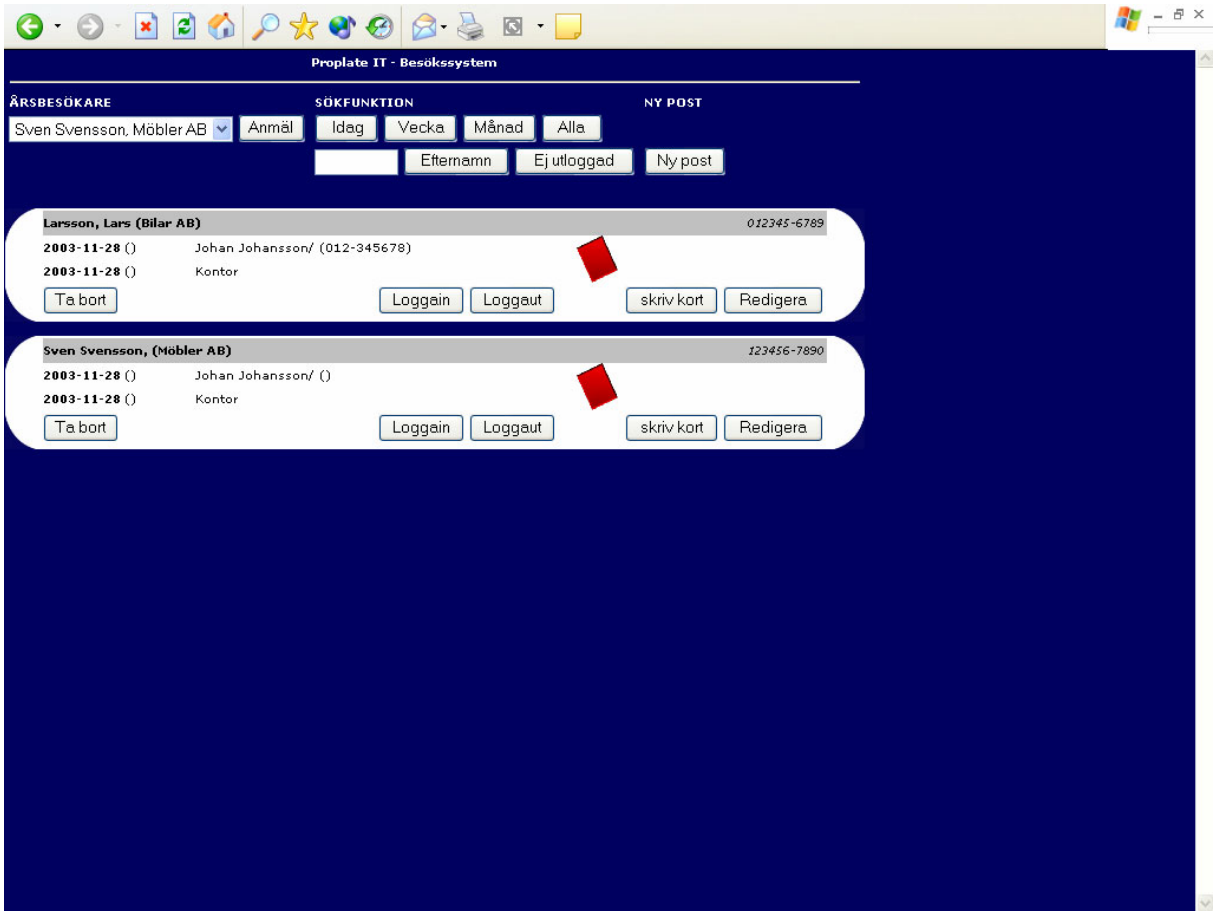
Figur 2-2 Rullande information

2.4.3 Reception

Den här sidan (Figur 2-3) används av receptionisten. Här kan receptionisten logga in och logga ut besökare. Med att logga in och ut besökare menar vi att tiderna för deras ankomst och avresa bokförs i databasen. Receptionisten kan också skriva ut besökskort och anmäla årsbesökare. Årsbesökare är besökare som blivit godkända av säkerhetschefen att komma på besök under ett helt år, till exempel år 2003. Det går även att ta bort eller redigera besök och använda sorteringsfunktioner för att visa besöken i en viss ordning. ”Ny post” knappen (i Figur 2-3) tar användaren till sidan för att lägga till en ny användare (2.4.4).

Några saker som vi anser är mindre bra är att när man loggat in en besökare kan man senare logga in samma besökare igen utan att först logga ut honom. Samma sak gäller också för att logga ut en besökare. Redigera knappen tar användaren till en helt ny sida för ändring av data, sidan för att lägga till nya besökare återanvänds alltså inte. För att göra en sökning med olika kriterier (Idag, Vecka, Månad m.m.) trycker man på motsvarande knapp. Det anser vi tar upp onödigt utrymme och det ska ändras till vår lösning.

Figur 2-3 Reception



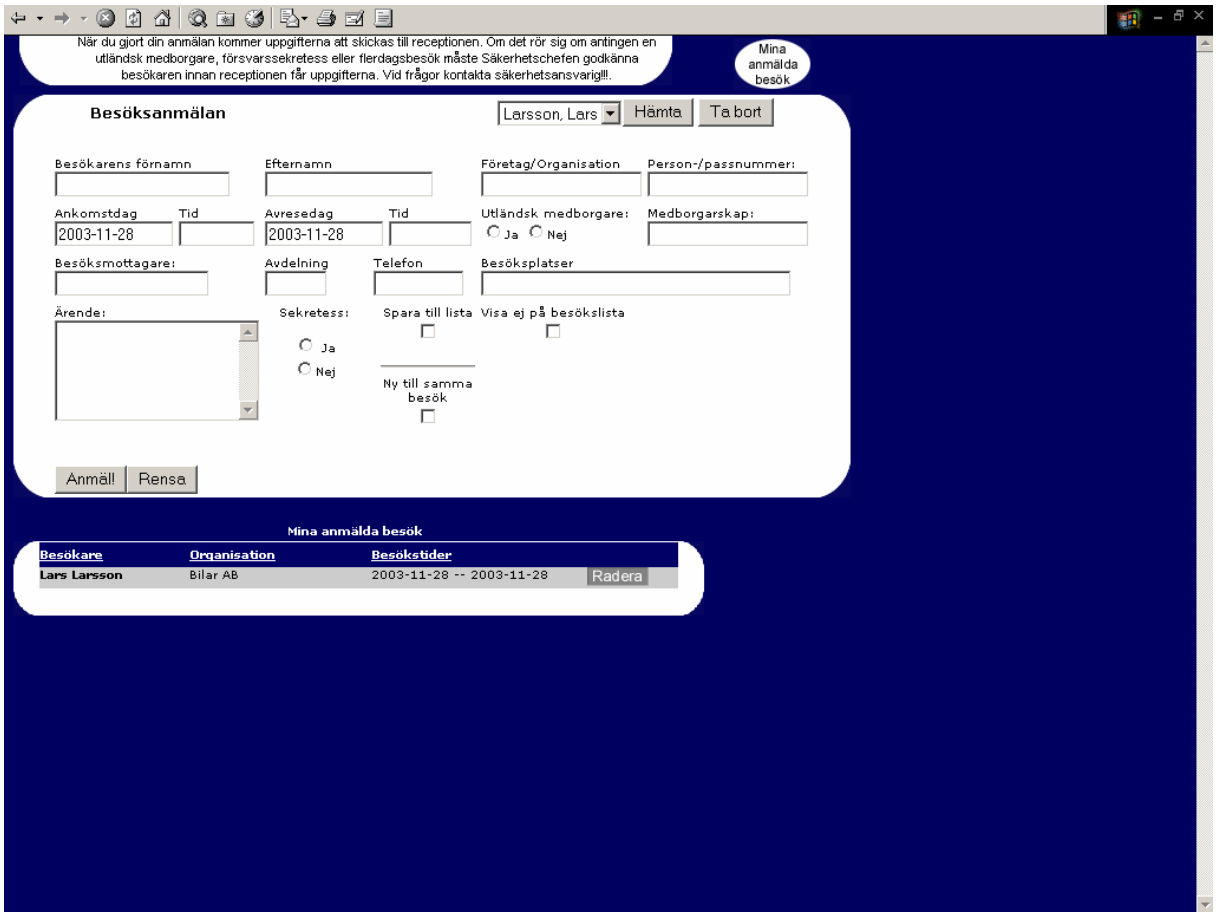
2.4.3.1 Mål för förändringar

- Inga runda hörn i presentationen av besökare. Hörnen är små GIF-bilder som endast tynger ned sidan och gör koden mer svårläst.
- Ändra position på knappar och utskriftsfält i presentationen av besökare.
- Sökfunktionen bör göras som en rullningslist där alla sökord är samlade. Innan har det varit enskilda knappar för Idag, Vecka, Månad, Alla och Ej Utloggad.
- Alla knappar ska ha samma storlek.
- Sökfunktion och presentation ska få liknande utseende (tabell med ram).
- Hjälptext ska visas i form av en "pop-upp ruta" om man håller muspekaren över exempelvis knappen logga in.
- Receptionisten ska inte kunna logga in en besökare som redan är inloggad och det ska inte gå att logga ut en besökare som redan är utloggad.

2.4.4 Nytt besök

Alla anställda med behörighet kan anmäla att en person ska komma på besök till företaget genom att fylla i en besöksanmälan (Figur 2-4). När man håller musen över ett inmatningsfält skrivs en hjälptext ut längst upp på sidan. Vid användning av kryssrutan ”spara till lista” sparas ifylld besökare till användarens personliga lista över besökare. Den här besökaren kan sedan återladdas dvs. att vissa inmatningsfält automatiskt fylls i. ”Ny till samma besök” fyller automatiskt i vissa fält när föregående besökare har sparats till databasen. Om en besökare har samma ankomstdag som avresedag och är svensk medborgare blir besöket automatisk godkänt. Alla andra fall måste säkerhetschefen godkänna. Den ovala cirkeln längst upp till höger sätter fokus på listan över användarens anmälda besökare. Alla besökare får röd bricka vilket innebär att besökaren inte får röra sig själv i lokalerna. Säkerhetschefen kan senare ändra den här brickan till grön vilket gör att besökaren får röra sig fritt. Vi upptäckte ett märkligt fel i systemet när man matade in personnumret på ett speciellt sätt. Besökaren fick då grön bricka. Besökarens förnamn, efternamn, personnummer, utländsk medborgare, ärende och sekretess är obligatoriska fält och kontrolleras av JavaScript vid sändning av formuläret.

Figur 2-4 Besöksanmälan



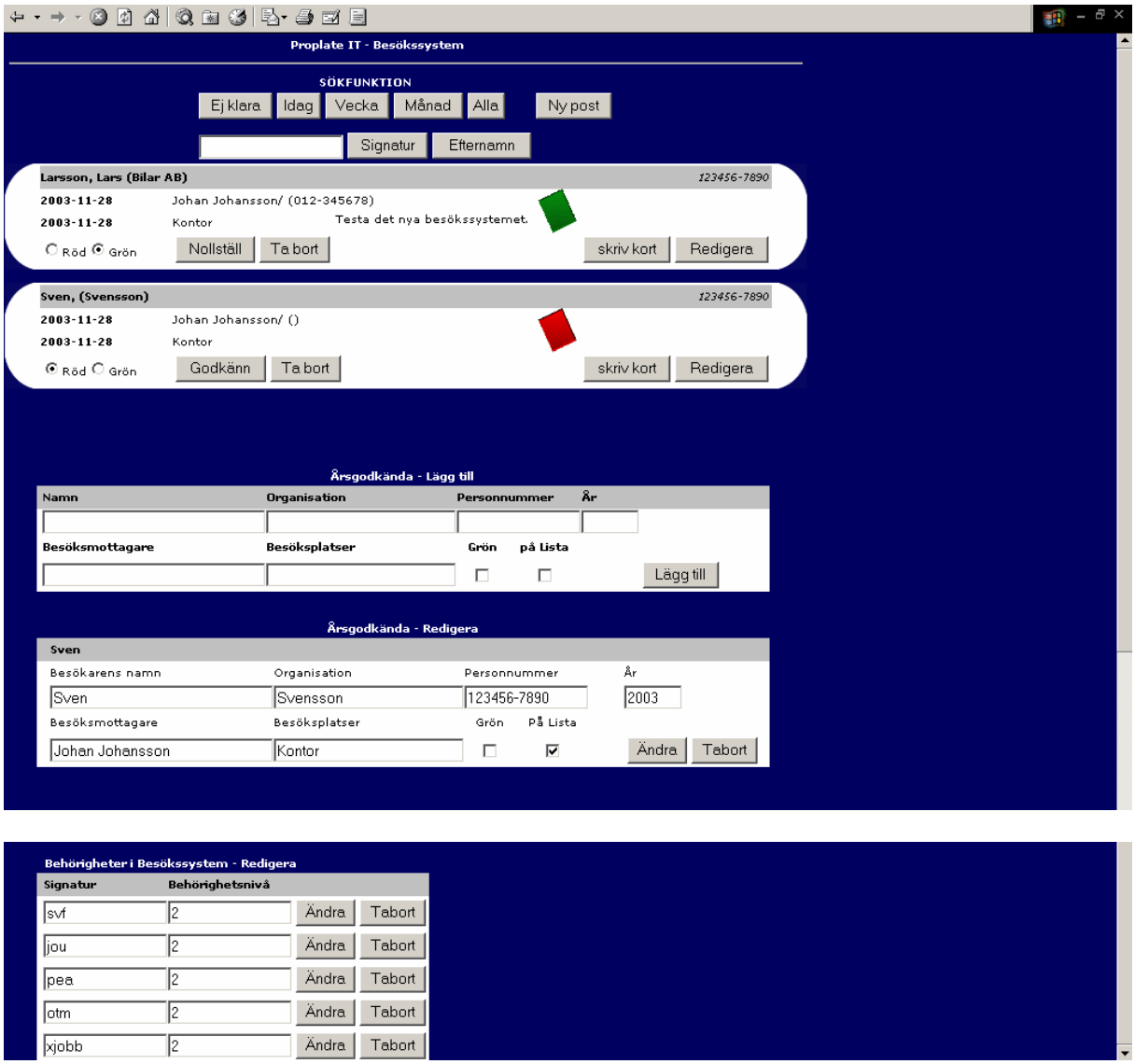
2.4.4.1 Mål för förändringar

- Ändra designen av formuläret till vad vi anser vara mer logiskt och mindre rörigt. Fälten ska uppdelas efter besökarens uppgifter och mottagarens uppgifter.
- Hjälptext ska visas i form av en liten ”popup” ruta när man håller muspekaren över textboxar eller knappar.
- Inga runda hörn ska finnas på formuläret eller presentationen av anmäld besökare.
- Den ovala cirkeln ”Mina anmälda besök” tas bort eftersom den egentligen saknar funktionalitet och ändå inte syns att det är en knapp.
- Storleken på knapparna skall vara lika i respektive formulär/tabell.

2.4.5 Godkänna besök

På den här sidan (Figur 2-5) har säkerhetschefen möjlighet att godkänna besök, tilldela behörighet, redigera/ta bort besök, skriva ut besökskort, lägga till/redigera årsbesökare, tilldela och administrera behörigheter i systemet. Här sker sökningen på samma sätt som i receptionen dvs. med hjälp av knappar. Alla årsbesökare visas på samma sida med ifylld information, det här medför att varje årsbesökare tar upp stor plats på sidan. Godkänn knappen godkänner besöket. Nollställ återställer besöket och det blir inte längre godkänt.

Figur 2-5 Godkänn besök



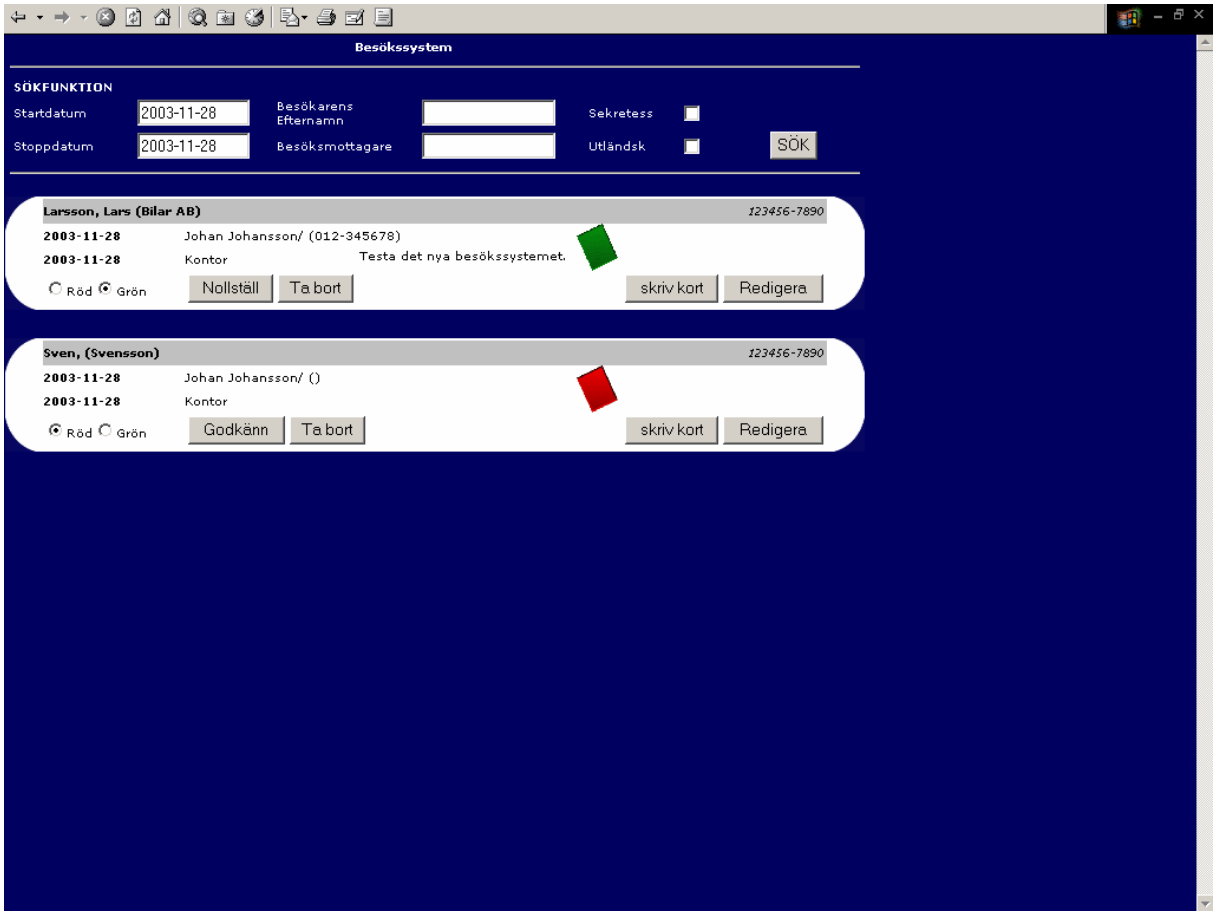
2.4.5.1 Mål för förändringar

- Dela upp sidan i separata sidor.
- Sökfunktionens knappar ska ersättas av en rullningslist för att reducera antalet knappar.
- Alla knappar ska ha samma storlek. Sökfunktion och presentation ska få liknande utseende (tabell med ram).
- Inga runda hörn ska finnas i presentationen av besökare.
- Behålla liknande utseende som övriga sidor till skillnad mot den här sidan som helt plötsligt har slutat med de runda hörnen.
- Hjälptext ska visas om man håller muspekaren över exempelvis knappen ”redigera”.
- Presentationen av de befintliga årsbesökarna ska göras smidigare. Istället för att alla besökare listas under varandra med all info redigerbar, så trycker man på redigera så fylls ett inmatningsformulär i med den redan inlagda besökarens uppgifter (sparar utrymme).

2.4.6 Historik

Den här sidan (Figur 2-6) används av säkerhetschefen för att söka efter alla besök som finns i databasen. Man kan söka med kriterierna startdatum, stoppdatum, besökarens efternamn, besöksmottagare, sekretess och utländsk. Den här sidan är också tänkt att säkerhetschefen ska kunna använda för att godkänna och redigera besök.

Figur 2-6 Historik



2.4.6.1 Mål för förändringar

- Inga runda hörn ska finnas i presentationen av besökare.
- Ändra position på knappar, checkrutor och utskriftsfält i presentationen av besökare.
- Sökfunktion och presentation ska få liknande utseende (tabell med ram).
- Ändra positionen på sökfält och knappar i sökfunktionen.
- Alla knappar ska ha samma storlek.

2.5 Problem

Den gamla koden är alldeles för rörig och svårläst för att ha någon nytta av. För att förstå sig på funktionaliteten var vi tvungna att testa de fall som kunde inträffa på samtliga sidor. Det vi gjorde var att vi anmälde besökare med olika kriterier och kontrollerade hur de uppträdde i systemet. Det vill säga vilken behörighet de fick och vilka sidor de visades på osv. Det här var en viktig del då all vår logik som vi kom att implementera baserades på det här testet. Oftast var det inte helt självklart hur sidorna var tänkta att fungera. Vår handledare var vid ett tillfälle tvungen att kontakta upphovsmannen till den gamla koden.

3 Design av de nya gränssnitten

Här tar vi upp hur vi har skapat de nya gränssnitten. Vi beskriver de tekniker vi använt oss av och vi visar också upp de nya gränssnitten och beskriver deras funktionalitet. Det här gör vi för att man klart och tydligt ska kunna se hur de nya gränssnitten skiljer sig ifrån de gamla. Strukturen över hur gränssnitten är uppbyggda tas också upp.

3.1 Tekniker

För att skapa gränssnitten krävdes det att vi friskade upp våra kunskaper i HTML (”Hyper Text Markup Language”) och läste in oss på hur olika saker som till exempel CSS (”Cascading Style Sheets”) fungerade. De kunskaperna fick vi genom att söka på Internet. Vi ska här beskriva de tekniker som vi använt oss av när vi skapat gränssnittet till den nya versionen av besökssytemet. Själva gränssnittet är uppbyggt i HTML med CSS och för att kontrollera att korrekt data inmatats använder vi JavaScript.

3.1.1 HTML

"HTML” är en förkortning som står för "Hyper Text Markup Language", vilket kan översättas ungefärligt till "Markeringsspråk för hypertexter". En hypertext är en text som innehåller en länk som leder dig till ett nytt dokument när du klickar på den. HTML är det markeringsspråk man använder när man gör hemsidor. HTML började dyka upp i slutet av 80-talet och har utvecklats i ett antal versioner sedan dess. 1993 kom en första standard, HTML 1.0, som skapades för den tidens enda webbläsare Mosaic. Programmet skapades av bl.a Marc Andreessen, som även låg bakom webbläsaren Netscape. I slutet av 1995 kom den första moderna versionen av HTML, som fick beteckningen HTML 2.0. Den utarbetades av IETF (the Internet Engineering Task Force). Nästa version, HTML 3.2, gjordes av W3C (the World Wide Web Consortium) och blev standard i början på 1997. Men det dröjde inte länge innan behoven på nytt vuxit och man behövde en ny revidering. Så i november 1997 presenterade W3C HTML 4.0. Den här versionen är mer eller mindre den slutgiltiga versionen, även om små revideringar har kommit. HTML 4.01 presenterades i slutet av 1999. Varje version innebär att ett antal nya element och funktioner lagts in i HTML-språket.

På grund av den snabba utvecklingen på Internet så skulle det behöva komma nya versioner minst en gång i halvåret, det är dock inte praktiskt möjligt och därför utvecklas inte html längre. Det finns andra möjligheter som man kan använda om det finns behov för saker som inte html klarar.

3.1.1.1 Exempel på HTML

```
<HTML>
<HEAD>
<TITLE> Exempel </TITLE>
</HEAD>
<BODY>
        <B> Den här texten blir fet </B>
</BODY>
</HTML>
```

Om man låter en webbläsare tolka ovanstående rad blir resultatet:

Den här texten blir fet

Webbläsaren tolkar alltså raden ”<B> Den här texten blir fet </B>” som att den ska göra den tjock. Det här gör den eftersom vi märkt ut texten med följande tags ”<B> och </B>”.

3.1.1.2 Formulär

För att kunna ta hand om användarens inmatade värden använde vi oss av formulär. Alla formulärobjekt i ett formulär omsluts av HTML-elementet ”<FORM>...</FORM>”. Elementet har två viktiga parametrar nämligen method och action. Formulärdata skickas till webbservern med protokollet http. Det finns två möjliga metoder (methods): GET eller POST. Med attributet action anger du webbadressen till det script som ska ta emot data från formuläret till exempel ett script skrivit i ASP (”Active Server Pages”).

När användaren fyllt i fälten så klickar han/hon på ”skicka” knappen. Webbläsaren läser då av alla ifyllda värden som skickas till det mottagande scriptet som specificerats av parametern action. För varje ifyllt eller valt objekt i det ifyllda formuläret skickas ett *namn* och

motsvarande värde till scriptet. Namn och värde kommer från attributen hos formulärobjekten name och value. Name talar om namnet på formulärobjektet och value talar om värdet.

På script sidan (vi använder ASP se 5.1.2) så kan man ta emot data genom att anropa Request.Form("name") som returnerar värdet som formulärobjeket innehåller.
Exempel på hur det fungerar finns i den bifogade koden i Bilaga B.

3.1.2 CSS Cascading Style Sheets

Cascading Style Sheets eller CSS som det förkortas är en lite nyare teknik när det gäller skapandet av webbsidor. CSS koden infogas i HTML-koden och fungerar alltså inte fristående. Med den här tekniken kan vi uppnå en del effekter som tidigare var omöjligt och det underlättar också väldigt mycket när det gäller underhållandet av kod. CSS dök upp för första gången i december 1996 och har sedan dess blivit allt vanligare, nu används version CSS2. Lägsta krav är Internet Explorer 4 eller Netscape Navigator 4. En fördel är att använda sig av både Internet Explorer och Netscape Navigator när man testar sina webbsidor. Det är nämligen inte alltid sant att de producerar samma resultat.

Eftersom det idag är så vanligt med webbsidor och att de ständigt växer i storlek behövs det nya tekniker för att lättare kunna underhålla koden på ett enkelt och smidigt sätt. Det är där CSS kommer in. Tidigare när man till exempel skulle ändra bakgrundsfärg på alla sidorna så hade man varit tvungen att ändra i html koden på varje enskild sida. När du använder Cascading Style Sheets så räcker det att göra ändringen på ett enda ställe. Det sparar mycket arbete och framförallt en massa tid. Den enda nackdelen som finns är dock att vissa webbläsare kan tolka koden lite annorlunda men det har alltid varit ett problem när man kodar webbsidor.

Det vanligaste sättet att använda CSS på är att man skapar ett fristående dokument med all CSS kod. Dokumentet döps till lämpligt namn med ändelsen ”.css”. Det här dokumentet kan sedan användas för att styra utseendet på samtliga webbsidor genom att i HEAD sektionen i HTML koden för varje webbsida referera till det här dokumentet.

3.1.2.1 Exempel på innehåll i en CSS fil

Här har vi skapat en klass myText med några attribut i filen stylesheet.css.

```
.myText
{
    FONT-SIZE: 8pt;
    COLOR: black;
    FONT-FAMILY: Arial
}
```

Exempel på en webbsida som refererar till vårt stylesheet och som skriver en liten text inne i en tabell.

```
<HTML>
<HEAD>
<LINK rel="stylesheet" type="text/css" href="stylesheet.css">
<TITLE>Exempel på stylesheet</TITLE>
</HEAD>
<BODY>
<TABLE>
<TR>
        <TD class="myText">
        Här skrivs vår text och den är av typen myText som
        beskrivs i stylesheet.css
        </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

3.1.3 Java Script

JavasScript är ett scriptspråk för webbsidor. JavaScript bäddas in i HTML. Även om namnen är väldigt lika så är Java och JavaScript inte samma sak. Det är två olika programmeringstekniker. Java är ett programspråk. JavaScript är ett scriptspråk. För att JavaScript ska fungera krävs det att webbläsaren har stöd för JavaScript och att det är aktiverat. Den första webbläsaren som hade stöd för JavaScript var Netscape Navigator 2.0.

Ett exempel med JavaScript:

```
<html>
<head>

<script language="JavaScript">
     function klicka_knapp()
      {
      alert("Det här är ett exempel");
      }
 </script>

</head>
<body>
  <input type="button" name="Button1" value="Klicka
  "onclick="klicka_knapp()">
</body>
</html>
```

På sidan visas en knapp när du klickar på knappen anropas funktionen klicka_knapp() som i sin tur ”poppar up” en alert ruta med texten ”Det här är ett exempel”.

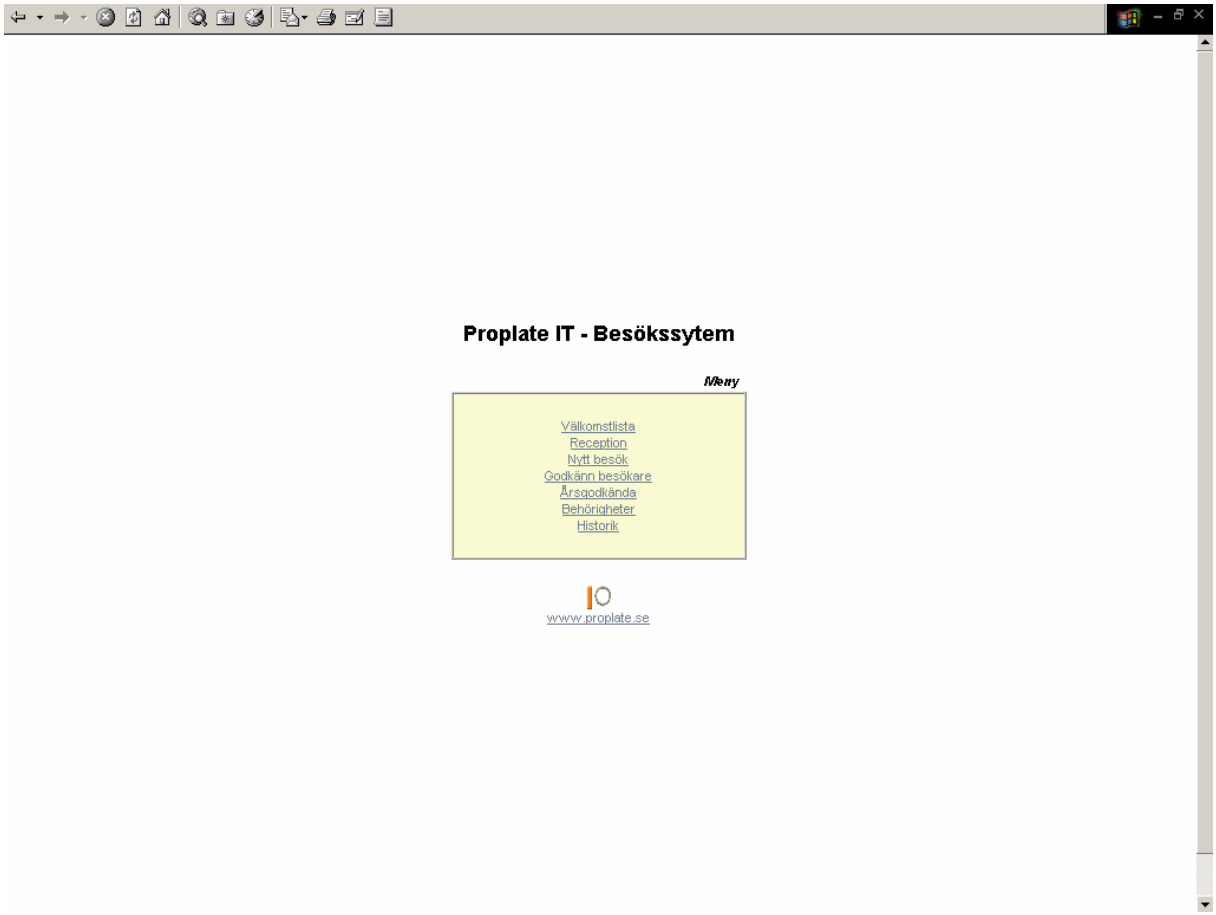
3.2 Gränssnitten

De gränssnitt som vi skapat skiljer sig en del från det föregående systemet, men det finns givetvis stora likheter. De ska utföra samma uppgift så det vore konstigt om inte likheterna fanns där. Vi kommer här att gå igenom varje gränssnitt var för sig och beskriva deras funktionalitet, samt påpeka de väsentliga skillnader som gjorts. Vi har försökt att undvika onödiga finesser och har satsat på en rak och enkel design. Det medför att det kommer bli enklare för framtida uppgraderingar.

3.2.1 Meny

Det är fortfarande lite oklart hur användarna av systemet kommer att använda de olika sidorna. Antingen så lägger dom till de sidor de har behörighet till under ”favoriter” i Internet Explorer eller så kommer den här menyn(Figur 3-1) att användas.

Figur 3-1 Meny



3.2.2 Välkomstlista

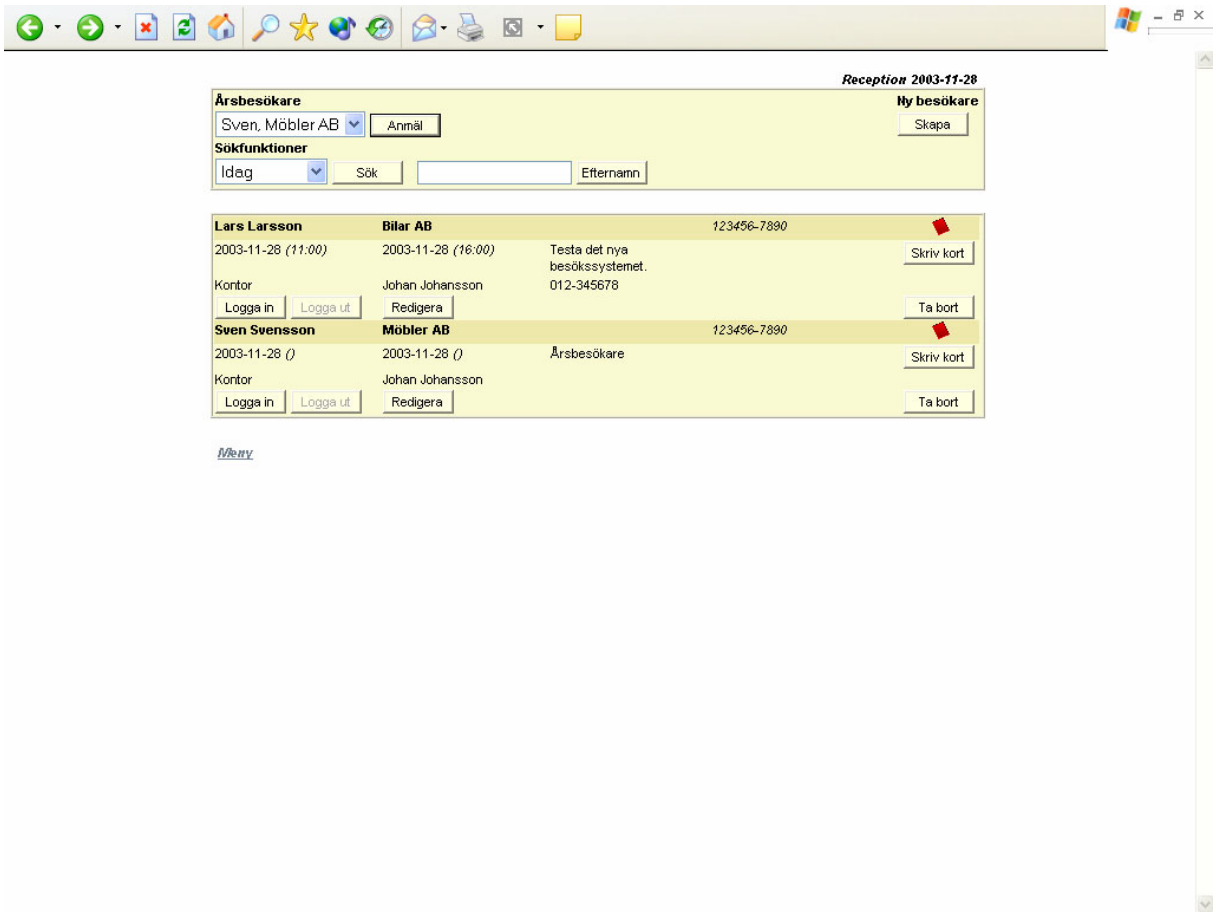
Den här sidan (Figur 3-2) fungerar exakt som den gamla (2.4.2).

Figur 3-2 Välkomstlista

3.2.3 Reception

Funktionaliteten på den här sidan (Figur 3-3) är den samma som tidigare (2.4.3). Vi har flyttat om sökfälten och sorterat om besökarens uppgifter för att öka tydligheten. Knapparna för olika sökkriterier ligger i en rullningslista. Receptionisten kan nu endast ”logga in” en besökare om besökaren är ”utloggad”. Det omvända gäller för att ”logga ut” en besökare.

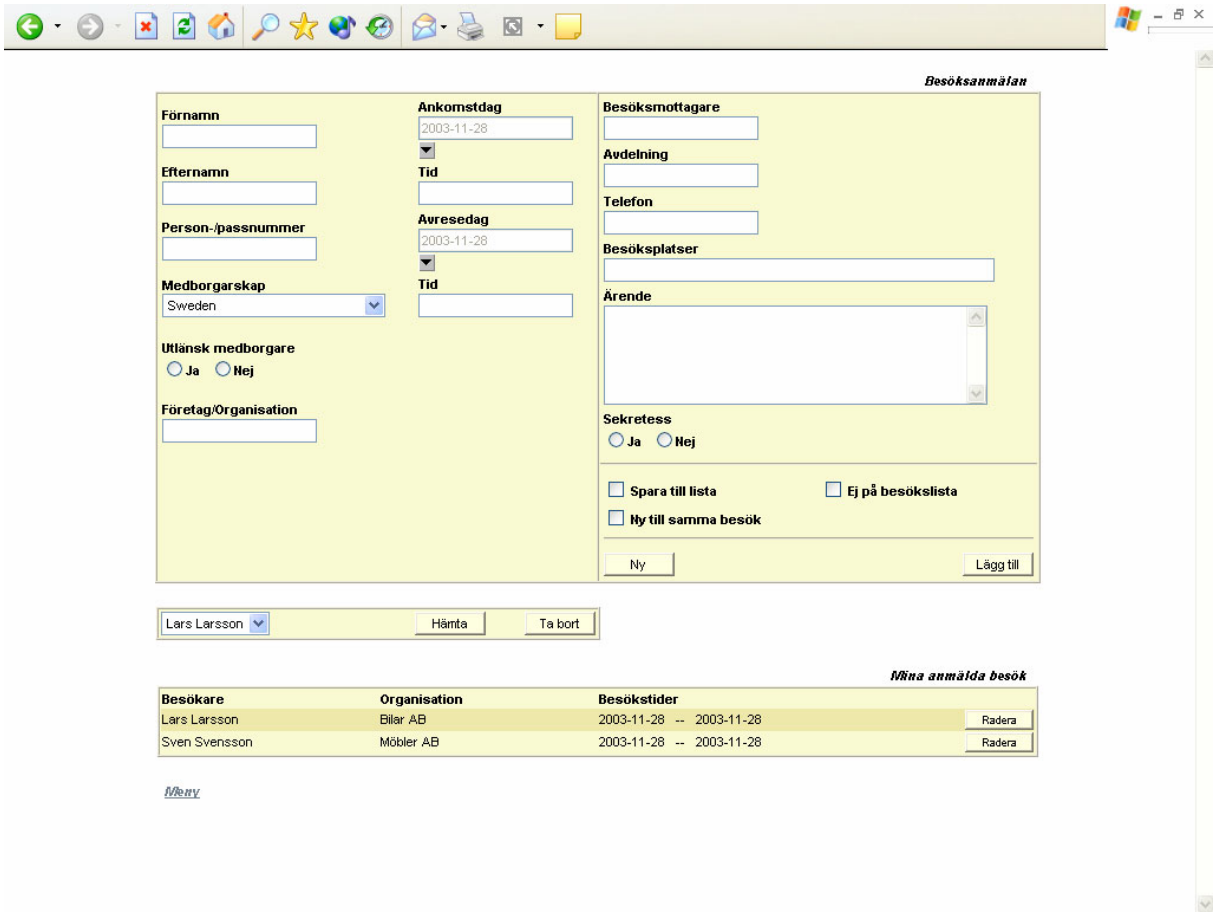
Figur 3-3 Reception



3.2.4 Nytt besök

Funktionaliteten på den här sidan (Figur 3-4) är densamma som tidigare (2.4.4). Fälten är sorterade och uppdelade efter besökare och besöksmottagare för att öka tydligheten. Datumfälten för ankomstdag och avresedag kan man inte längre skriva in text i. Istället trycker man på pilen bredvid fältet för att få upp en kalender där man väljer ett datum. Pilen för datumval är en bild som anropar en kalendersida som Proplate IT tillhandahöll. Samma fält som tidigare är obligatoriska för ifyllning. Det här kontrolleras även i det nya systemet av JavaScript. Sidan används även för redigering av besökare. Det vill säga att när exempelvis receptionisten vill redigera ett besök så återanvänds koden för den här sidan. På så sätt undviker vi att skapa en helt ny sida vars syfte enbart är att hantera redigering. Vid redigering laddas den berörda besökarens uppgifter in från databasen. Fältet för användarens egna sparade besökare (andra formuläret uppifrån i Figur 3-4), anmälda besök (tredje formuläret uppifrån i Figur 3-4), samt meny länk visas inte vid redigering. När redigeringen är slutförd skickas användaren tillbaka till den sida redigeringen startade från.

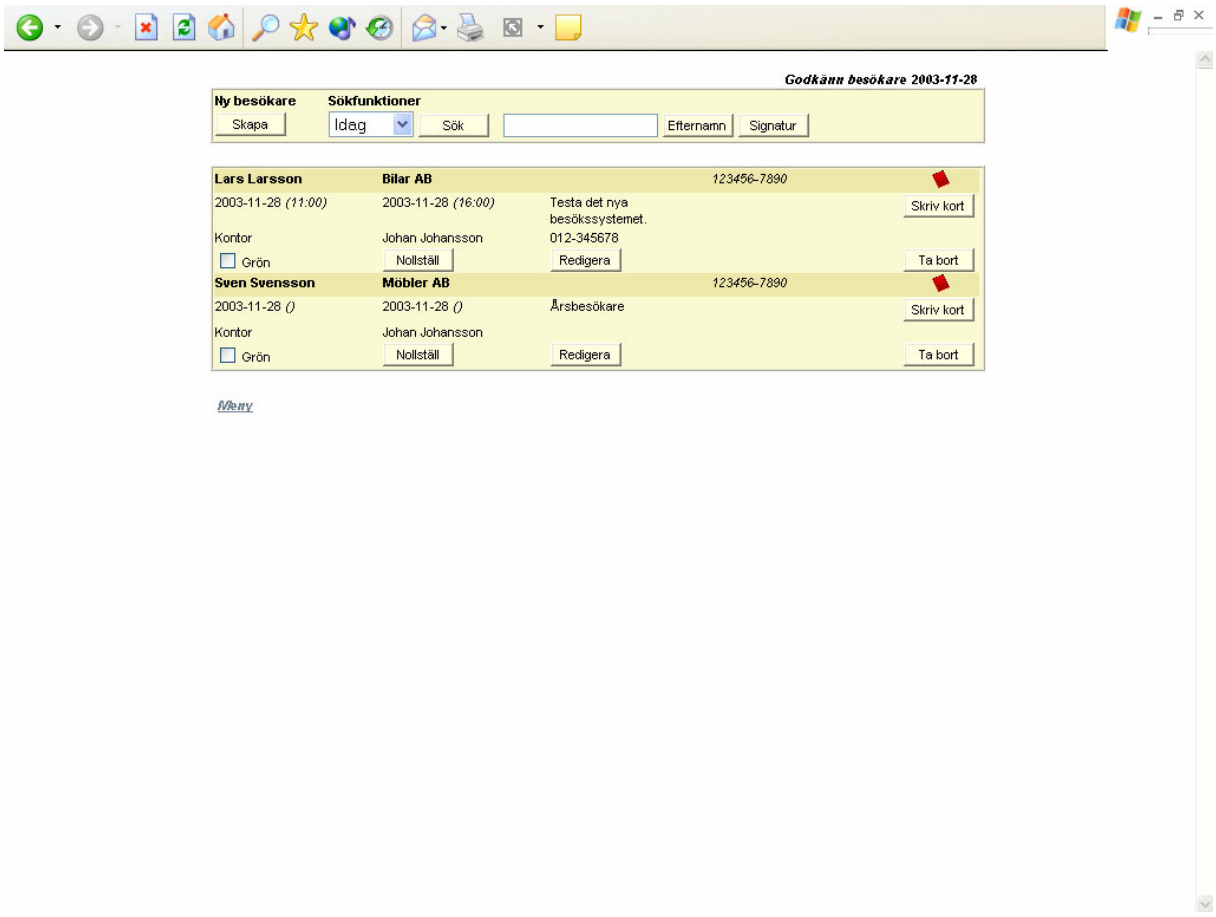
Figur 3-4 Besöksanmälan



3.2.5 Godkänn besökare

Det här är den första (Figur 3-5) av de tre sidor som vi skapat från sidan ”godkänn besök” (2.4.5) i det gamla systemet. Här kan behörig (oftast säkerhetschefen) söka efter olika besökare eller besök. Funktionaliteten är densamma som tidigare. När användaren trycker på redigera knappen så öppnas ”Nytt besök” sidan (3.2.4) där uppgifter automatiskt fylls i.

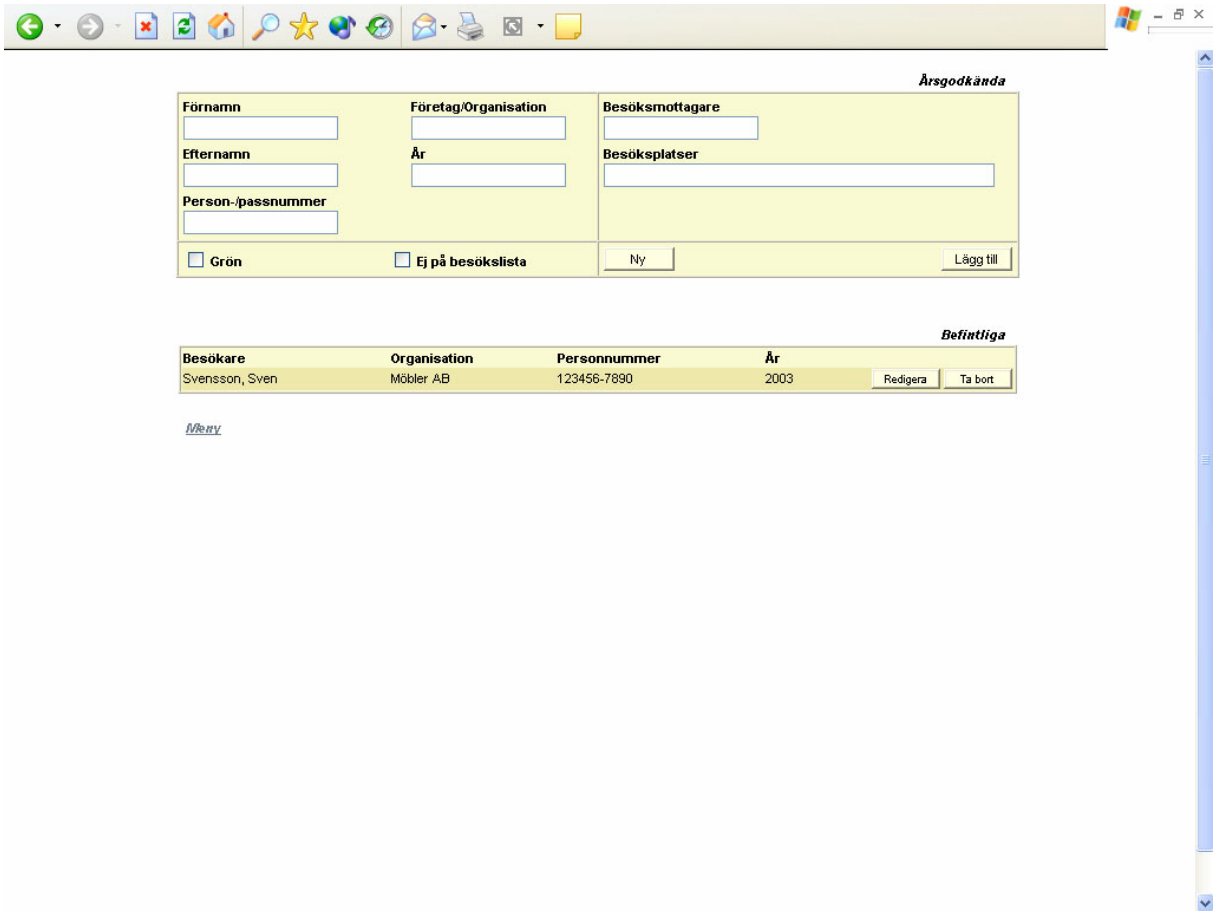
Figur 3-5 Godkänn besökare



3.2.6 Årsgodkända

Det här är den andra (Figur 3-6) av de tre sidorna som var en och samma sida i det gamla systemet (2.4.5). Funktionaliteten är i stort densamma. Det som skiljer är att de inlagda årsbesökarna nu visas i en lista. För att redigera en årsbesökare trycker man på ”redigera” knappen. Formuläret längst upp på sidan fylls då i automatiskt med årsbesökarens uppgifter. I det gamla systemet visades alla årsbesökare i redigerbar form direkt på sidan.

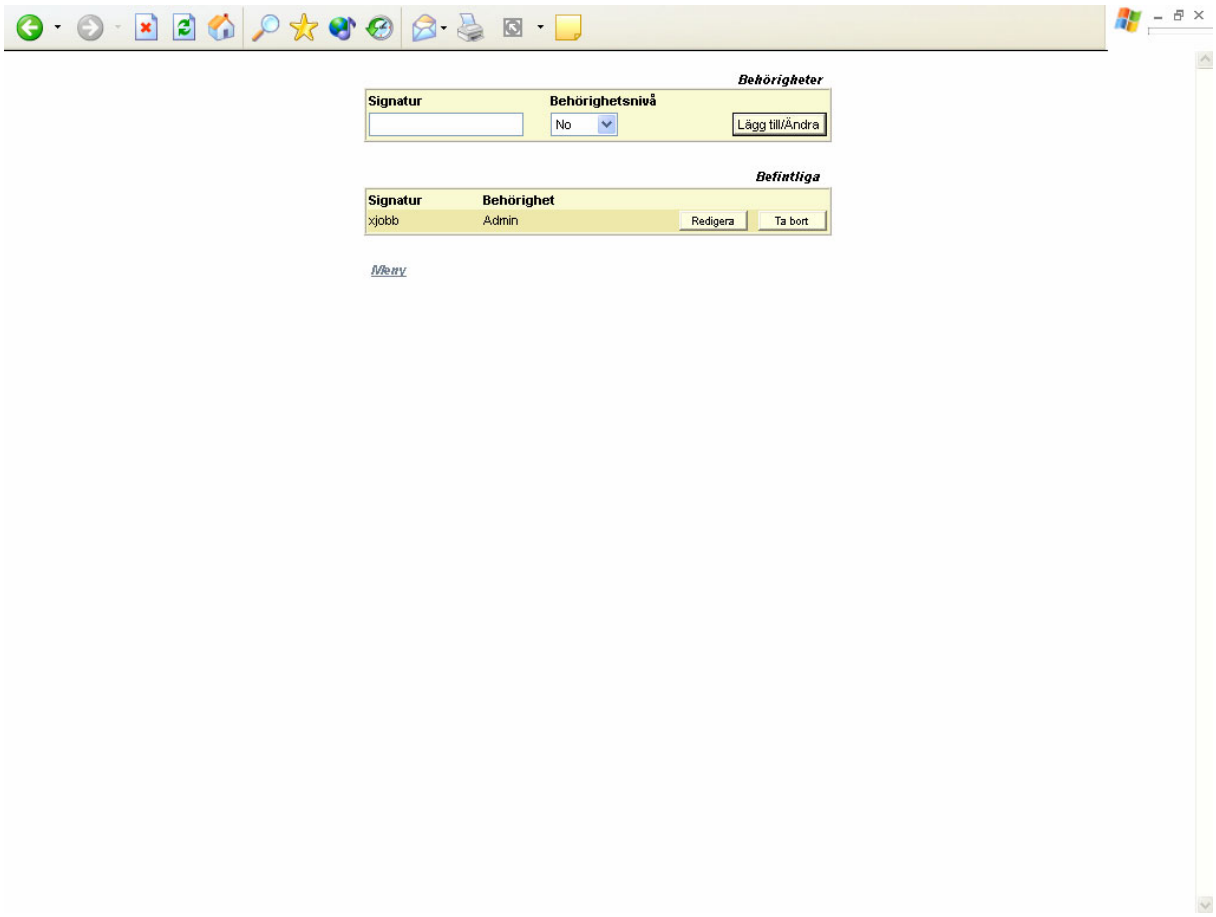
Figur 3-6 Årsgodkända



3.2.7 Behörigheter

Det här är den tredje (Figur 3-7) av de tre sidorna som vi skapat från sidan ”godkänn besök” (2.4.5) i det gamla systemet. Fungerar på ungefär samma sätt som tidigare. Den stora skillnaden är att de inlagda behörigheterna nu visas i en lista. När man vill redigera en användares behörighet trycker man på redigera knappen. Formuläret längst upp på sidan fylls då i automatiskt med användarens uppgifter. Den här lösningen sparar på utrymme eftersom användarna nu inte längre visas i direkt redigerbar form, vilket de gjorde i det gamla systemet.

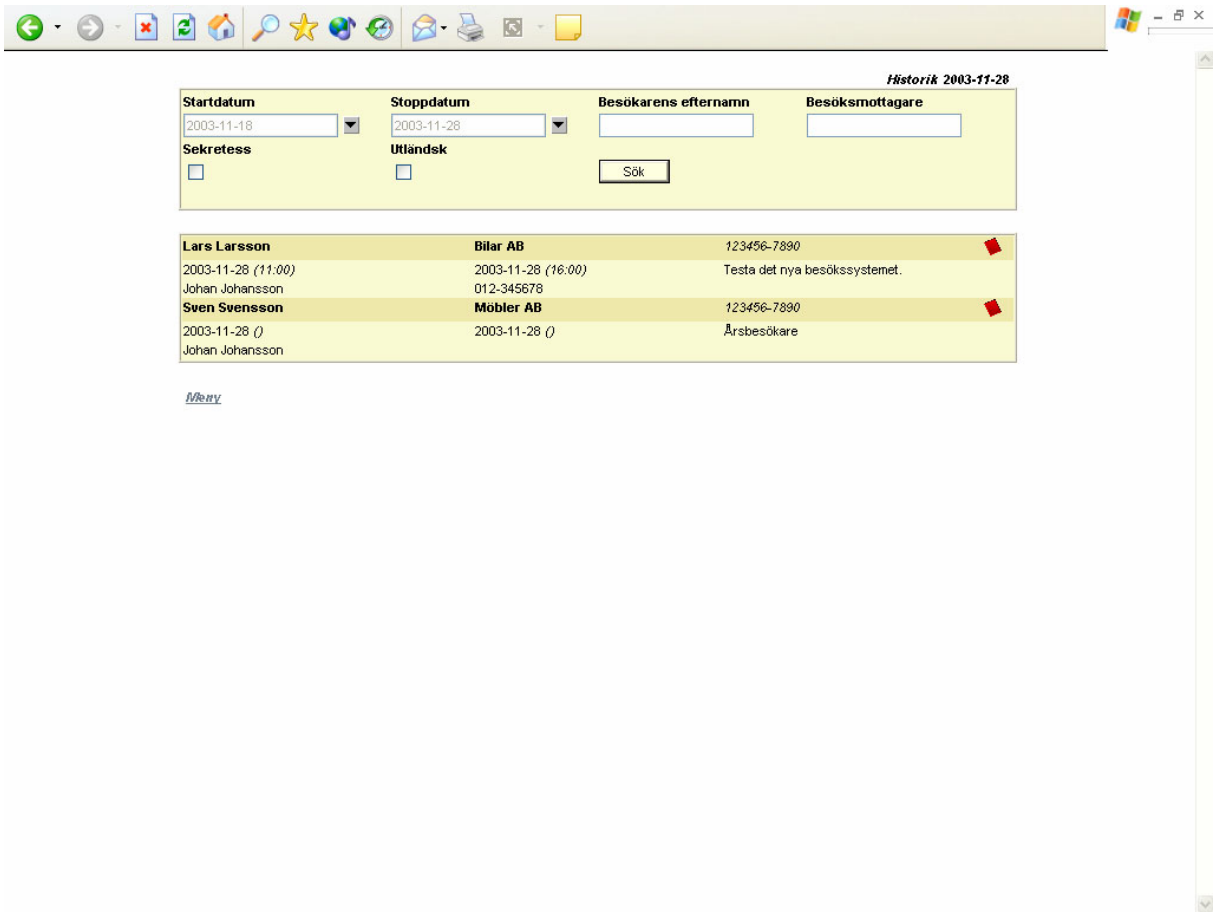
Figur 3-7 Behörigheter



3.2.8 Historik

I det gamla systemet påminner ”historik” sidan (2.4.6) väldigt mycket om ”godkänn besökare” sidan (2.4.5). Vi har valt att i det nya systemet inte ta med funktionalitet för att ändra färg på kortet, godkänna, ta bort, skriva kort och redigera. Historik sidan (Figur 3-8) ska vara till för att visa gamla anmälda besök och de ska man inte kunna redigera i efterhand.

Figur 3-8 Historik



3.3 Gränssnittets struktur

Gränssnitten är skrivna i HTML och vi har använd oss av tabeller för att bygga upp de olika inmatningsformulären. Formulär (Form) kommer att användas för att skicka data till en ASP sida som lägger till eller uppdaterar databasen. Namnsättningen av de olika formulärobjekten följer en standard som bestämts av företaget. Namn på inmatningsfälten i ett formulär ska exempelvis alltid börja med ”txt”. All kod och kommentering har gjorts på engelska. Exempel på hur det ser ut finns i bilaga B. På varje sida där vi ska ha någon form av lista som visar besökare, har vi lyft ut listan till en separat fil. Det gjordes för att det ska vara lättare att hitta den om man ska göra ändringar. Det vi gjort för att koden ska vara lättläst är att indentera den så att man klart och tydligt kan se vart olika saker börjar och slutar. Vi använder CSS för att underlätta uppdateringar av gränssnittets utseende.

4 Design av databas

Här kommer vi att beskriva den process som lett fram till vår slutgiltiga databasmodell. En mer detaljerad beskrivning av samtliga tabeller och kopplingen dem emellan kommer också att beskrivas.

4.1 Designprocessen

Det första vi gjorde var att skriva ner alla fält som skulle komma att behövas till exempel förnamn, efternamn och besöksmottagare. När vi gjort det så började vi fundera ut bra beskrivande namn på tabeller och fält. Då vi skulle behålla det gamla systemets funktionalitet uppstod det ett problem. Problemet uppstod därför att man i det gamla systemet inte behövde fylla i vissa fält. De fälten som var obligatoriska kunde innehålla vilka tecken som helst. Det blir därmed svårt att säkerställa om en besökare redan har varit inlagd vid ett tidigare tillfälle. Därför fick vi göra så att alla besök blir en ny post i databasen. Det i sin tur leder till att en viss redundans uppstår då samma person faktiskt kan lagras på mer än ett ställe i databasen. Det här kan inte undvikas eftersom systemet ska vara uppbyggt på det här sättet.

Efter vi konstaterat att alla besök ska lagras som en ny post i databasen så började vi med att skissa på en lösning. Vi gjorde ett enkelt ER-diagram som vi visade upp för vår handledare. Efter en del diskussioner fram och tillbaka med vår handledare enades vi om att en del ändringar behövdes göras. Anledningen till ändringarna var att databasen ska vara bättre anpassad för framtida behov. Den slutgiltiga lösningen som vi hade kommit fram till var att vi ska ha två huvudtabeller. Den ena (Visitors) ska innehålla allt om besökaren exempelvis namn och ärende. Den andra (Visits) ska innehålla allt om själva besöket, till exempel ankomstdag och tid för inloggning. Vi ska också skapa tre tabeller som tar hand om data för vilken typ av besökare det är. En tabell (VisitorType) som innehåller vilken typ av besökare det handlar om. I vårt fall har vi normalbesökare eller årsbesökare och vill man ha fler är det bara att lägga till dem i tabellen. Den tabellen ska ha en koppling till en andra tabell (VisitorProperty) som innehåller egenskaperna för besökstypen. Den tredje tabellen (PropertyValue) ska innehålla de aktuella värdena. Övriga tabeller i databasen blev Cards, Citizenships, Users, UserList och SecurityLevel. Vi beskriver tabellerna lite mer ingående längre fram.

Vi övergick också till att designa databasen direkt i designläget i sql server då vi nu hade en klar bild av hur databasen ska se ut. För att kunna ha en historik över vilka besökare som varit inloggade så valde vi att inte radera besöken fysiskt ur databasen. När de ska tas bort använder vi oss av ett fält (VisitActive) som sätts till 0 om besöket ska anses vara raderat.

När vi väl hade skapat alla tabeller så bestämde vi också vilka datatyper som alla fält skulle ha. Fältens längd och vilka fält som skulle vara obligatoriska bestämdes också. När vi gjort klart det så konverterade vi databasen från SQL Server till Access. Nu hade vi två identiska databaser en i Microsoft SQL Server och en i Microsoft Access.

4.2 Beskrivning av tabellerna

Här beskriver vi de olika tabellerna i databasen i stora drag.

Visits

I den här tabellen sparas all data som hör till ett besök. Varje besökare får en koppling till en och endast en post i den här tabellen. Poster i den här tabellen återanvänds inte. Varje gång en besökare läggs till i databasen skapas en helt ny post i tabellen. Det här görs för att allt som läggs in i databasen ska finnas kvar för visning senare (t.ex. historik sidan (3.2.8)). Inga data tas heller bort vid borttagning av ett besök. Istället sätts ett attribut fält (VisitActive) om till ett värde som motsvarar "borttagen".

Visitors

I den här tabellen sparas all data som hör till en besökare. Inga poster återanvänds här. Varje gång en besökare kommer på besök (även om besökaren har varit där tidigare) skapas en ny post i tabellen. Den största anledningen till det här är att vid anmälan av en ny besökare finns ingen koppling till de redan befintliga i databasen. Det finns nämligen inga krav på den som anmäler ett besök hur inmatningen av uppgifter ska skrivas. Därmed kan man inte utföra en matchning mot uppgifterna i databasen.

Users

I den här tabellen sparas alla användare (signatur) av systemet och deras rättighetsnivå. Nivåerna finns beskrivna i övergripande funktionalitet (2.3).

UserList

I den här tabellen sparas koppling mellan en användare av systemet och vilka besökare som användaren vill ha sparade till sin egen lista. Den här listan kan man använda på "nytt besök" sidan (3.2.4) för att snabbare anmäla en besökare.

SecurityLevels

I den här tabellen sparas beskrivningen på de säkerhetsnivåer som finns i systemet.

Citizenships

I den här tabellen lagras alla länder som är fördefinierade i systemet (av oss). Alla länder är inskrivna på engelska.

Cards

I den här tabellen lagras de olika korttyper som finns i systemet.

VisitorType

Den här tabellen är fördefinierad (av oss). Här lagras vilka olika typer av besökare det finns i systemet. I dagens läge finns endast två stycken olika besökare (normalbesökare och årsbesökare).

VisitorProperty

Den här tabellen är fördefinierad (av oss). Här lagras de olika egenskaper som en speciell besökstyp kan ha.

PropertyValue

I den här tabellen lagras värden för besökarens egenskaper.

5 Logik

Här beskriver vi hur vi gjort kopplingen mellan gränssnitten och databasen. Vi tar upp de tekniker vi använt oss av. Här tar vi även upp hur vi löste uppgiften med att systemet skulle gå att köra mot både Microsoft Access och Microsoft SQL Server. Hur frågor ställs mot databasen och hur vi löste säkerheten tas även det upp. Här beskriver vi också hur felhanteringen löstes och hur vi lade upp strukturen på sidorna.

5.1 Tekniker

För att vi skulle kunna koppla gränssnitten mot databasen krävdes att vi undersökte hur det skulle gå till. Det gjorde vi genom att vi sökte information på Internet och testade oss fram. Vi rådfrågade även personal på Proplate för att få deras synpunkter på det hela. Här beskriver vi lite olika tekniker vi har använt oss av samt hur vi löste problemet med att systemet ska gå att köra mot både Access databas och SQL Server.

5.1.1 SQL

SQL ("Structured Query Language") är ett språk som man använder vid kommunikation mot databaser. Det är det helt dominerande språket i databassystem. SQL utvecklades från första början av IBM, men efter ett tag så började många programvaruutvecklare göra egna varianter. Språket är standardiserat, men det förekommer i mängder med olika dialekter beroende på vilket företag det är som har utvecklat databashanteraren. För att ta ett exempel så har Microsoft sin egen variant i Access. Den första formella specifikationen för språket antogs av ANSI 1986 och av ISO 1987. 1992 sågs standarden över och man tog fram en ny specifikation, SQL-92. Det är den specifikationen som flertalet tillverkare följer.

SQL är både ett manipuleringsspråk (Data Manipulation Language) och ett definitionsspråk (Data Definition Language), det vill säga det går både att definiera relationer och samband mellan relationer samt hämta och modifiera data i relationerna.

En sats i SQL är uppbyggd av reserverade ord i språket och användarens egendefinierade ord. Språkets reserverade ord kan man inte ändra på. De måste stavas exakt och kan inte splittras upp på flera rader. De av användaren definierade orden, har konstruerats av användaren, i enlighet med speciella regler. De orden representerar namn på olika databasobjekt till exempel tabeller, kolumner, vyer, index och så vidare. I många dialekter av SQL så krävs det även att varje sats avslutas med ett semikolon.

Microsoft Access

Access är en databashanterare med ett grafiskt användargränssnitt tillverkat av Microsoft. Access är egentligen bara det grafiska verktyget. Själva databasmotorn heter Jet och distribueras med Micosoft Windows eller med tilläggskomponenterna MDAC (Microsoft Data Access Components)

Microsoft SQL Server

SQL Server är en databashanterare från Microsoft. Precis som de flesta andra är den av relationstyp och använder SQL som frågespråk. SQL-dialekten som används heter Transact-SQL (T-SQL).

5.1.2 ASP

ASP (”Active Server Pages”) är ett språk för att bygga dynamiska HTML sidor. En HTML sida innehåller bara statisk information som aldrig kan ändra sig förrän någon går in i filen och byter ut innehållet. I en ASP sida kan man med hjälp av variabler och funktioner bygga ett innehåll i en sida som kan ändra sig beroende på olika händelser. En ASP sida är en kombination av programkod och HTML. När en användare gör ett anrop mot en ASP sida körs först programkoden på servern. Därefter levererar den en ren HTML sida tillbaka till användaren. Användaren kan således inte se den bakomliggande logiken för sidan. Det behövs inget speciellt verktyg för att skriva ASP sidor, man kan använda sig av en vanlig text editor.

För att man ska kunna köra en ASP sida måste servern som sidan ligger på, ha stöd för ASP. Den vanligaste servern är ”Internet Information Server (IIS)” från Microsoft som bara kan användas på en Windows NT-server. Alternativet till att använda ASP är PHP (Personal Home Page eller Hypertext Preprocessor). ASP och PHP klarar av i princip samma saker. Att använda oss av PHP var aldrig något alternativ då vi fick order om att använda oss av ASP.

5.1.2.1 Exempel på ASP

```
<HTML>
<HEAD>
<TITLE>Exempel på ASP</TITLE>
</HEAD>
<BODY>
<TABLE>
<TR>
        <TD>
        Dagens datum är:<%=Date%><BR>Klockan är:<%=Time%>
        </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Där man vill skriva ASP kod i html sidan startar man med tagen ”<%”. Därefter följer utskriften som görs genom att man skriver ”=” innan uttrycket. Uttrycken i det här fallet är de inbyggda servervariablerna ”Date” och ”Time”. Sist så avslutas ASP tagen med ”%>”. När den här sidan anropas av användaren ersätter servern ”<%=Date%>” med dagens datum i HTML koden. Samma händer med ”<%Time%>” och aktuell tid skrivs in i HTML koden.

Det användaren ser är t.ex. endast det här:

```
Dagens datum är:2003-12-31
Klockan är:23:10:01
```

På det här sätt bygger man upp en dynamisk sida med innehåll som kan variera och ändra sig.

5.1.2.2 Request.QueryString

När vi kodat det nya systemet har vi ofta använt request.querystring, som ingår i ASP. Här beskriver vi lite mer detaljerat hur request.querystring fungerar.

Värdet av Requets.QueryString(parameter) är en array med alla värden som finns i servervariabeln QUERY_STRING. Request.Querystring hämtar värdet av variablerna som finns med i HTTP´s querystring. HTTP´s querystring specificeras av de värden som följer efter frågetecknet (?).

Många olika processer kan generera en querystring. Till exempel kan man generera en variabel med namn string, som får värdet ”det här är ett exempel” på följande sätt:

```
<A HREF="exempel?string=det här är ett exempel">string exempel</A>
```

Querystrings kan också genereras när man skickar ett formulär eller att man skriver direkt i adress fältet i sin webbläsare.

Syntax

Request.QueryString(*variable*)[(*index*)|.Count]

Parametrar

variable

Specificerar namnet på den variabeln i HTTP´s querystring som ska hämtas.

index

En valfri parameter som tillåter dig att få tag på ett av flera multipla värden för variable.

Det kan vara ett heltal från 1 till Request.QueryString(*variable*).Count.

Ett exempel

Följande sträng skickas:

```
http://www.besökssystem.se/test/name.asp?name=Sven
```

Det skulle resultera i följande QUERY_STRING värde:

```
name=Sven
```

Nu kan vi använda informationen i ett ASP script:

```
Hej, <%=Request.QueryString("name")%>.
```

Resultat:

Hej, Sven.

5.1.2.3 Access och SQL Server

För att på ett enkelt sätt kunna ändra vilken sorts databas systemet ska köras mot skapades för det första två olika databaser, en i Access och en i SQL Server. Dessa två databaser måste vara identiska till utseende (tabeller, datatyper m.m.) för att vi ska kunna använda oss av samma kod. Vårt första försök till att få systemet att fungera mot båda databaserna var att skapa lagrade procedurer i varje databas. En lagrad procedur är egentligen en SQL fråga som kan exekveras utifrån med vissa inparametrar (om man så vill). Lagrade procedurer snabbar upp exekvering av frågor mot databasen eftersom dom är kompilerade. Det blir också en något högre säkerhetsgrad i systemet eftersom frågorna endast återfinnes i själva databasen. Vi upptäckte snart att vi inte på ett enkelt sätt kunde exekvera frågorna i Access databasen och i SQL databasen på samma sätt från ASP. För att få det att fungera skulle vi vara tvungna att sätta en variabel i koden som har information om vilken databas systemet körs emot. Vid varje tillfälle där en fråga exekveras skulle man behöva kontrollera variabeln för att se vilken databas man ska ställa frågan mot. Det här leder till mycket dubbel kodning och blir inte entydigt. Om man vill ha fram annan information ur en fråga och ändrar i t.ex Access måste man också gå in i SQL server för att ändra frågan. Det här blir omständligt. Vi sökte därför istället en lösning då vi kunde exekvera samma fråga mot de olika databaserna. Den lösning vi fann var att skapa en koppling mot databasen via ADO (ActiveX Data Objects). ADO gör det möjligt att koppla upp sig och utföra ändringar och uppdateringar mot en databas. När en fungerande koppling har upprättats skapas ett ”recordset” också det via ADO. Ett ”recordset” är en variabel som innehåller svarsinformation från databasen. Via det här ”recordsetet” kan man nu exekvera frågor mot vald databas. Frågorna ligger lagrade i variabler direkt på sidorna och är skrivna på ett sådant sätt att dom kan köras mot både Access och SQL Server. Uppkopplingen mot databaserna sker alltså nästan på exakt samma sätt, det är endast databasens namn, användarnamn och lösenord som skiljer dom åt. Vi har valt att separera de två databaskopplingarna i två filer (”database_sql.inc” och ”database_access.inc”). Det gjordes för att förenkla administrationen. Alla sidor inkluderar filen ”database.inc” och i den finns informationen om vilken databas sidorna ska ansluta sig till. Vill man byta och köra mot en annan databas skriver man över filen ”database.inc” med den nya (t.ex ”database_access.inc”).

Den lösning som vi valde var den som vi fick att fungera först. På grund av tidsbrist gjordes inga större undersökningar om det fanns andra metoder/sätt att lösa det på. Fördelen med att kunna köra mot både Access och SQL server är att det nu är möjligt för fler företag

att kunna använda systemet. När systemet installeras på ett företag så bestäms det vilken databas som skall användas beroende på om företaget använder SQL server eller Access. När det gäller underhåll av databasen så blir det inga problem eftersom man inte kör mot både SQL server och Access samtidigt. Skulle man däremot villa byta så är kravet att databaserna är identiska.

5.1.2.4 Frågor mot databas

Här behandlar vi kort hur man ställer frågor mot en databas och behandlar svaret man får tillbaka.

Lägg till

När vi lägger till en ny post i databasen måste vi göra det generellt (eftersom det ska fungera mot båda Access och SQL Server). För att etablera ett ”recordset” gentemot databasen måste först en fråga exekveras. Resultatet man får tillbaka kan sedan ändras lokalt och därefter gör man en uppdatering som också sker i databasen. För att lägga till en helt ny post vill vi därför inte få något resultat tillbaka från vår fråga. Därför ställs frågan enligt nedan:

```
recordset.open ”SELECT * FROM <table> WHERE  <attribute ID> = -1”
```

Förklaring:

recordset= en recordset variabel

<table>= tabellens namn

<attribute ID>= identifierare för en specific rad

Eftersom det aldrig finns ett ID i databasen med värdet -1 kommer den här frågan alltid returnera NULL

Nu kan vi skapa en ny rad i tabellen <table> genom att skriva:

```
recordset.AddNew
```

”recordset” får nu tilldelat alla attribut som finns i tabellen <table>. Vi kan nu lägga till de data vi vill ha i de olika attributen i tabellen.

```
recordset.fields(”<attribute>”)=<value>
```

Vi avslutar det hela genom att göra en uppdatering av ”recordset” och därmed uppdateras även databasen.

recordset.update

recordset.close

Uppdatera

Det här sker ungefär på samma sätt som med ”lägg till” fast vi returnerar en post från databasen som redan finns.

```
recordset.open ”SELECT * FROM <table> WHERE <attribute ID> = <existing ID>”
```

Resterande sker på exakt samma sätt.

5.1.3 Säkerhet

För att användare inte ska kunna komma åt sidor de inte ska ha behörighet till så har vi satt en variabel på varje sida som innehåller sidans säkerhetsnivå. Nivån på variabeln jämförs med användarens säkerhetsnivå som finns lagrad i databasen. Användarens namn får vi tag på genom att anropa funktionen GetUser. Resultatet av GetUser funktionen använder vi sedan för att kunna hämta säkerhetsnivån ur databasen med hjälp av en SQL fråga. Sedan görs en jämförelse av nivåerna. Skulle användarens nivå vara för låg så skickas man vidare till ”error.asp” sidan där ett meddelande kommer upp att användaren har blivit nekad åtkomst till sidan.

5.1.4 Felhantering

På varje sida i systemet tar vi hand om eventuella fel som kan uppstå. Det finns generellt tre stycken olika fel som kan uppstå på en ASP sida.

"Compile-time errors"

De här beror oftast på fel i koden vilket förhindrar ASP från att ”kompilera” (sidan kan inte genereras). T.ex. om man glömmer att avsluta en "Do" loop eller en "If" sats.

"Runtime errors"

De här felen uppkommer då man försöker exekvera ASP sidan. T.ex. om man försöker tilldela en variabel ett otillåtet värde.

"Logic errors"

Logiska fel är svårare att upptäcka. Problemet ligger någonstans i strukturen av koden. Det kan till exempel vara ett fel i algoritmen. Det krävs testkörning för att upptäcka de här.

Om ett fel inträffar som vi inte har kunnat förutse ska inte systemet ”gå ner”. Med att systemet ”går ner” menas att en konstig sida (med i många fall oförståelig text) visas för användaren. För att komma runt det har vi använt oss av följande kod som inledning på varje sida.

```
<%on error resume next%>
```

Den här koden innebär att om webbläsaren påträffar ett fel hoppar den över den del av koden där felet är och går vidare. Säkerheten av systemet påverkas inte av det här eftersom det utförs på servern innan användaren kan ta del av sidan. Men det här innebär också att användaren av systemet inte blir informerad om att något gått fel. Det kan vara lika frustrerande som att se en oförståelig sida med text. Därför måste man ta hand om fel som uppstår och ge användaren ett meddelande om att något gått fel. På de ställen man kan misstänka att det kan bli fel lägger man till en kontroll för att upptäcka dem.

I filen "visitor_approved_list.inc" finner vi den här koden för att kontrollera om ett fel har uppstått:

```
*fråga som ställs mot databasen
<%if err.number=0 then%>
*Kod som läser in data från databasen
<%else%>
        Response.Redirect("error.asp?err=20&pageref=year_approved_
list.asp")%>
<%end if%>
```

"err.number" tilldelas en felkod då något har gått fel. Man kan inte kontrollera alla möjliga fel som kan uppstå, men har ett fel inträffat innan den här tidpunkten i koden är det troligen ett databasspecifikt fel. Om "err.number" fortfarande är "0" vid tidpunkten för kontrollen är allt som det ska. Om "err.number" är något annat än "0" skickas användaren vidare till en ny sida som vi har valt att kalla "error.asp". Den här sidan tar emot querystringen "err" som innehåller en felkod som vi har hittat på och "pageref" som innehåller namnet på den sida som felet uppstod på. "error.asp" tar sedan hand om dessa parametrar och skriver ut ett felmedelande.

Vi har valt att använda oss av följande felkoder och felmedelanden.

```
10 Error opening database.
20 Error in recordset.
30 Error writing to database.
40 UserName not found in database.
50 User denied.
60 No records found.
70 Error reading from database.
```

5.2 Sidornas struktur

Varje sida har delats upp på följande sätt. Vi har vår huvudsida (gränssnittet), den som användaren kommer att se. Sedan har vi en sida som har hand om att lägga till data och uppdatera data i databasen. Vi har en sida som tar hand om att ta bort data ur databasen. När sidorna utfört uppdatering eller borttagning så omdirigeras man direkt till huvudsidan. Vi valde den här lösningen för att slippa få väldigt komplexa sidor. I bilaga B så finner ni koden för en av våra sidor (year_approved). Anledningen till att samtliga sidors kod ej är med i bilagan är att företaget inte ville att all kod skulle publiceras. Eftersom alla sidor är uppbyggda på precis samma sätt så får man ändå en klar bild över hur systemet är uppbyggt.

Strukturen är alltså följande:

<namn>.asp - Huvudsidan

<namn>_data.asp - Sidan för att lägga till och uppdatera databasen.

<namn>_remove.asp - Sidan för att ta bort besökare ur databasen

<namn>_list.inc - Innehåller koden som innehåller listan med befintliga besökare

Utöver ovanstående finns det några generella filer som beskrivs lite närmare längre fram. De är följande:

GetUser.inc, Database.inc, Recordset.inc, Security_check.inc, End_of_page.inc, Error.asp

Huvudsidan <namn>.asp

Efter ”On error resume next” som används för felhanteringen på varje sida inkluderar vi funktionen ”GetUser” som hämtar namnet på användaren. Vi valde att lägga funktionen ”GetUser” i en fil som inkluderas eftersom det vore onödigt att skriva samma kod på varje sida. Efter det följer två inkluderingar till ”database.inc” och ”recordset.inc”. I ”database.inc” sker kopplingen mot antingen SQL Server eller Access. I ”recordset.inc” skapas vårt recordset. Nästa sak som följer är säkerhetsnivån. På varje sida har vi en variabel där man sätter aktuell sidas säkerhetsnivå. Vi inkluderar här även filen ”security_check.inc” som innehåller kod för att hämta användarens säkerhetsnivå ur databasen och jämföra mot sidans säkerhetsnivå. Vi valde den här lösningen på grund av enkelheten. Det gick väldigt snabbt att implementera den. Går vi vidare så kommer vi till sidans inställningar ”Page settings” där bland annat bredden för tabeller anges. De här inställningarna kan variera lite beroende på sida, exempelvis alla sidors tabeller är inte lika breda.

Nästa del är den där vi avgör vad sidan ska göra beroende på vad som skickats med i querystring. Vi har här en ”select case” sats som beroende på vilket action som skickas med i querystring avgör om data ska laddas från databasen. I vissa fall ska fält fyllas i och i andra fall ska de lämnas tomma. Anledningen till att huvudsidan får sköta olika uppgifter är för att undvika att skapa en ny likadan sida för varje enskild sak som ska utföras. Vi öppnar helt enkelt huvudsidan igen och skickar med en variabel ”action” i querystring och beroende på värdet så får sidan utföra en viss uppgift. Efter det så kommer JavaScriptet, funktionerna används för att kontrollera inmatningar eller att öppna en sida och skicka med värden i querystringen. Sen följer vår HTML och ASP kod för gränssnittet som vi redan beskrivit i tidigare kapitel. Formulär används för att skicka data till de ASP sidor som har till uppgift att uppdatera databasen. (Vi använder ibland även funktioner i vårt JavaScript för samma ändamål.) Vi använder ASP för at kunna utnyttja variabler och göra felkontroller och ställa frågor mot databasen. Efter HTML koden inkluderar vi filen end_of_page.inc som stänger vår koppling mot databasen.

6 Testning av systemet

Här beskriver vi hur testningen gått till. Utöver den kontinuerliga testningen som vi själva gjort under resans gång så har även vår handledare testat det färdiga systemet. Vi tar också upp resultatet av den slutgiltiga testningen.

6.1 Testerna

Vi har genomgående under hela utvecklingen av systemet testat de olika sidorna. Vi har inte följt någon speciell mall för hur testningen ska gå till väga. Vår uppgift var att få samma logik i det nya systemet som det gamla och därför jämförde vi också vårt system med det gamla. När vi testade en sida gjorde vi samma inmatningar i det nya som det gamla systemet och jämförde resultaten. Vi körde testerna i det nya systemet både mot Access och SQL Server. När resultaten blev de samma var testningen lyckad.

Vår handledare på Proplate IT gjorde den slutliga testningen när vi var färdiga med systemet. De synpunkter han kom med var endast små grafiska fel och några önskemål om förändringar. Vi rättade snabbt och enkelt till det mesta. Några saker gick inte att rätta till på grund av begränsningar i Internet Explorer. Ett exempel på detta är tabuleringsordningen. Vid användandet av ”radiobuttons” (Figur 3-4 Besöksanmälan, för att välja ja eller nej) går det inte att tabulera sig mellan valen. Radiobuttons som hänger ihop har endast ett tabuleringsvärde.

7 Slutsats

Arbetet har fortlöpt utan några större problem. Samarbetet mellan de inblandade parterna har fungerat utmärkt. De första dagarna gick åt till att bekanta sig med varandra och lägga upp en plan för hur arbetet skulle utföras. Vår planering höll igenom hela projektet. Vi känner oss nöjda med vår arbetsinsats eftersom vi har uppnått de mål som ställdes av Proplate. Det var också väldigt stimulerande att få erfarenhet av att jobba ute på ett företag. Det sista som inträffade på Proplate var att vår handledare testkörde systemet och kom med synpunkter som han tyckte behövde ändras. Här fick vi prov på hur enkelt det var att göra förändringar i koden på begäran. Den genomgående lika designen och upplägget av koden gör att man känner igen sig på varje sida. Har man väl satt sig in i hur en sida är uppbyggd förstår man även de andra. I och med detta kände vi att vårt mål med arbetat var nått. Systemet fungerar nu både mot Access och SQL Server och det är enkelt att läsa och uppdatera kod.

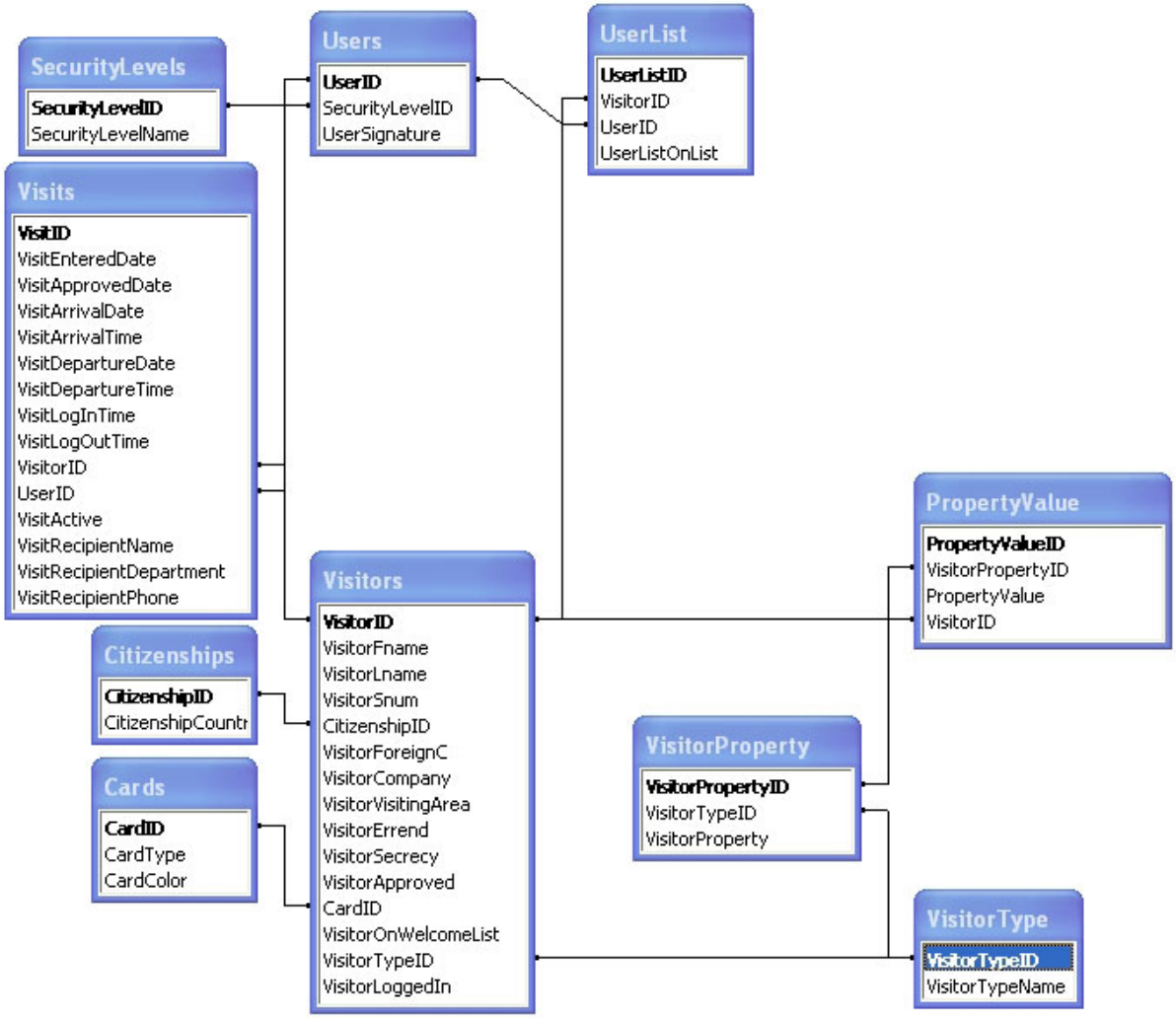
Referenser

[1] www.planet-source-code.com, 2003-10-01

[2] www.w3c.org, 2003-10-01

[3] www.devshed.com, 2003-10-01

A Databasdesign



B Kod

Koden för vårat besökssystem är genomgående lika och generell för samtliga sidor. Vi kommer här att presentera en vald del av den koden vi har skrivit. Varför vi inte publicerar all kod är för att Proplate IT troligen kommer ta det här systemet i bruk. Det är också onödigt att presentera all kod då den ser ganska lika ut för varje sida.

```
<%@ Language=VBScript %>
<%  'error.asp
    'Last updated: 2003-11-13
    'Name: Anders och Jonas

    'Page settings
    '-------------------------------------------------------------------------------
    dim border, tbwidth1,cmdwidth,cmdheight
    'border setting
    border=0
    'width for table
    tbwidth1=300
    'width for buttons
    cmdwidth=60
    cmdheight=20
    '-------------------------------------------------------------------------------%>

<%'HTML and ASP code
    '-------------------------------------------------------------------------------%>
    <HTML>
    <HEAD>
    <LINK rel="stylesheet" type="text/css" href="stylesheet.css">
    <TITLE>Error</TITLE>
    </HEAD>
    <BODY>
        <%'Outer table.%>
        <TABLE border="<%= border %>" width="100%" height="100%">
            <TR>
                <TD valign="middle"  align=center>
                    <%'Pageinfo text.%>
                    <TABLE width="<%= tbwidth1 %>">
                        <TR>
                            <TD width="100%" class="pageinfo" align="right">
                                ERROR  <%=request.querystring("err")%>   in   <%=Request.QueryString("pageref")%>
                            </TD>
                        </TR>
                    </TABLE>
                    <%'Frame around form (outer line).%>
                    <TABLE width="<%= tbwidth1 %>" class="table" border="1" cellspacing="0">
                        <TR>
                            <TD class="text" width="100%" valign="middle" align="center">
                                <%select case request.querystring("err")
                                        case 10%>
                                        Error opening database.
                                    <%case 20%>
                                        Error in recordset.
                                    <%case 30%>
                                        Error writing to database.
                                    <%case 40%>
                                        UserName not found in database.
                                    <%case 50%>
                                        User <%=request.querystring("user")%> denied.
                                    <%case 60%>
                                        No records found.
                                    <%case 70%>
                                        Error reading from database.
                                <%end select%>
                                    <br>Please contact your system administrator.
                            </TD>
                        </TR>
                    </TABLE>
                    <%'Link %>
                <BR>
                <TABLE width="<%= tbwidth %>">
                    <TR>
                        <TD class="pageinfo" align="left">
                            <A href="menu.asp">Meny</A>
                        </TD>
                    </TR>
                </TABLE>
                </TD>
            </TR>
        </TABLE>
    <P> </P>
    </BODY>
    </HTML>
<%'-------------------------------------------------------------------------------%>


<% 'getuser.inc %>
<%
Function GetUser

    Dim strUserName

    strUserName = Request.ServerVariables("LOGON_USER")
    strUserName = Right(strUserName, Len(strUserName) - inStr(1, strUserName, "\"))

    GetUser = strUserName

End Function
%>
<%'-------------------------------------------------------------------------------%>


<% 'recordset.inc %>
<%
dim rs
set rs=server.CreateObject("ADODB.Recordset")
rs.ActiveConnection=DatabaseConnection
rs.CursorType=2
rs.CursorLocation=3
rs.LockType=3
%>
<%'-------------------------------------------------------------------------------%>


<% 'security_check.inc %>
<%
rs.Open "select SecurityLevelID from Users where UserSignature=" & "'" & GetUser & "'"

if rs.fields("SecurityLevelID") < PageSecurityLevel then
    rs.Close
    Response.Redirect("error.asp?err=50&pageref=security&user=" & GetUser)
end if
rs.Close
%>
<%'-------------------------------------------------------------------------------%>


<%@ Language=VBScript%>
<%  'year_approved.asp
    'Last updated: 2003-12-05
    'Name: Anders Jansson, Jonas Olsson%>
<%on error resume next%>

<%'Function for retrieving user name.
    '-------------------------------------------------------------------------------%>
    <!--#include file="getuser.inc"-->
<%'-------------------------------------------------------------------------------%>

<%'Make database connection
    '-------------------------------------------------------------------------------%>
    <!--#include file="database.inc"-->
  <!--#include file="recordset.inc"-->  <%'Create recordset%>
<%'-------------------------------------------------------------------------------%>

<%'Check if user is authorized to see page.
    '-------------------------------------------------------------------------------
    dim PageSecurityLevel
    PageSecurityLevel=3%>
    <!--#include file="security_check.inc"-->
<%'-------------------------------------------------------------------------------%>

<%'Page settings.
    '-------------------------------------------------------------------------------
    dim border, tbwidth1, tbwidth2, cmdwidth, cmdheight
    'border settings
    border=0
    'width for table
    tbwidth1=352
    tbwidth2=tbwidth1*2
    'width for buttons
    cmdwidth=60
    cmdheight=20
    '-------------------------------------------------------------------------------%>
```

```
<%'Decide what to do (based on querystring).
    '--------------------------------------------------------------------------------
    dim FirstName, LastName, Snum, Company, RecipientName, VisitingArea, Year
    dim Card, OnList, AddUpdateText, AddUpdateFunc, VisitorID, Action

    'Text on submit button.
    AddUpdateText="Lägg till"
    VisitorID=-1
    OnList=1

    'Decide what action to take
    Action=clng(request.querystring("action"))
    select case Action

        'No action
        case 0

        'Load visitor from "My List"
        case 1
                VisitorID=request.querystring("visitorID")
                AddUpdateText="Ändra"
                rs.Open "select * from Visitors where VisitorID=" & VisitorID & ";"
                FirstName=rs.Fields("VisitorFname")
                LastName=rs.Fields("VisitorLname")
                Snum=rs.Fields("VisitorSnum")
                Company=rs.Fields("VisitorCompany")
                VisitingArea=rs.Fields("VisitorVisitingArea")
                Card=rs.Fields("CardID")
                OnList=rs.Fields("VisitorOnWelcomeList")
                rs.Close
                rs.Open "select * from Visits where VisitorID=" & VisitorID & ";"
                RecipientName=rs.Fields("VisitRecipientName")
                rs.Close
                rs.Open "select * from PropertyValue where VisitorID=" & VisitorID & " and VisitorPropertyID=2" & ";"
                Year=rs.Fields("PropertyValue")
                rs.Close
    end select
    '--------------------------------------------------------------------------------
%>

<%'HTML and ASP code for page.
    '--------------------------------------------------------------------------------%>
<HTML>
<HEAD>
<%'Javascript validator for page.
    '--------------------------------------------------------------------------------%>
    <script language="javascript">
    <!--#include file="year_approved_check.inc"-->
    function LoadYearApproved(idnumber)
    {
            window.location="year_approved.asp?visitorID="  + idnumber + "&action=1";
    }

    function DeleteYearApproved(idnumber, name)
    {
            del = confirm("Är du säker på att du vill ta bort " + name + "?");
            if(del)
                window.location="year_approved_remove.asp?visitorID="  + idnumber;
    }

    function NewYearApproved()
    {
            window.location="year_approved.asp";
    }

    </script>
<%'--------------------------------------------------------------------------------%>
<LINK rel="stylesheet" type="text/css" href="stylesheet.css">
<TITLE>Årsgodkända</TITLE>
</HEAD>
<BODY>
<%'Outer table.%>
<TABLE border="<%= border %>" width="100%" height="100%">
    <TR>
        <TD valign="top"  align=center>
            <%'Pageinfo text.%>
            <FORM action="year_approved_data.asp?visitorID=<%=VisitorID%>" onsubmit="return ValidateForm(this)" method="POST" name="YearApproved">
            <TABLE width="<%=tbwidth2%>">
                <TR>
                    <TD class="pageinfo" align="left">
                        * = obligatoriska fält
                    </TD>
                    <TD class="pageinfo" align="right">
                        Årsgodkända
                    </TD>
                </TR>
            </TABLE>
            <%'Frame around form (outer line).%>
            <TABLE class="table" border="1" cellspacing="0">
                <TR>
                    <TD valign="top">
                        <%'Year visitor input.%>
                        <TABLE width="<%=tbwidth1%>" border="<%=border%>">
                            <TR>
                                <TD width="45%" class="headline">
                                    * Förnamn
                                    <BR>
                                    <INPUT class="textbox" value="<%=FirstName%>" name="txtFirstName" tabIndex=1 title="Ange besökarens förnamn." maxlength="20">
                                </TD>
                                <TD width="10%">
                                     
                                </TD>
                                <TD width="45%" class="headline">
                                    Företag/Organisation
                                    <BR>
                                    <INPUT class="textbox" value="<%=Company%>" name=txtCompany tabIndex=7 title="Ange från vilket företag/organisation besökeren kommer ifrån." maxlength
                                        ="50">
                                </TD>
                            </TR>
                            <TR>
                                <TD width="45%" class="headline">
                                    * Efternamn
                                    <BR>
                                    <INPUT class="textbox" value="<%=LastName%>" name="txtLastName" tabIndex=2 title="Ange besökarens efternamn." maxlength="20">
                                </TD>
                                <TD width="10%">
                                     
                                </TD>
                                <TD width="45%" class="headline">
                                    * År
                                    <BR>
                                    <INPUT class="textbox" value="<%=Year%>" name="txtYear" tabIndex=9 title="Ange för vilket år besökaren är godkänd." maxlength="4">
                                </TD>
                            </TR>
                            <TR>
                                <TD width="45%" class="headline">
                                    Person-/passnummer
                                    <BR>
                                    <INPUT class="textbox" value="<%=Snum%>" name=txtSnum tabIndex=3 title="Ange besökarens person-/passnummer. Ex: 123456-7890" maxlength=12>
                                </TD>
                            </TR>
                        </TABLE>
                    </TD>
                    <TD  valign="top">
                        <%'Reciever input.%>
                        <TABLE width="<%=tbwidth1%>" border="<%=border%>">
                            <TR>
                                <TD class="headline">
                                    * Besöksmottagare
                                    <BR>
                                    <INPUT class="textbox" value="<%=RecipientName%>" name="txtRecipientName" tabIndex=12 title="Ange vem som är besöksmottagare." maxlength=50>
                                </TD>
                            </TR>
                            <TR>
                                <TD class="headline">
                                    Besöksplatser
                                    <BR>
                                    <INPUT size="60" value="<%=VisitingArea%>" class="textbox" name="txtVisitingArea" tabIndex=15 title="Ange besöksplatser för besökaren." maxlength=50>
                                </TD>
                            </TR>
                        </TABLE>
                    </TD>
                <TR>
                    <TD valign="bottom">
                        <%'Checkboxes.%>
                        <TABLE width="<%=tbwidth1%>" border="<%=border%>">
                            <TR>
```

```
                                <TD width="50%" class="headline">
                                <%if Card = 2 then%>
                                    <INPUT checked type="checkbox" name="chkCard" tabIndex=19" title="Markera om årsbesökaren ska ha grön bricka.">
                                <%else%>
                                    <INPUT type="checkbox" name="chkCard" tabIndex=19" title="Markera om årsbesökaren ska ha grön bricka.">
                                <%end if%>
                                Grön bricka
                                </TD>
                                <TD width="50%" class="headline">
                                <%if OnList = 0 then%>
                                    <INPUT checked type="checkbox" name="chkOnYList" tabIndex=20 title="Markera om årsbesökaren inte skall visas på välkomstlista.">
                                <%else%>
                                    <INPUT type="checkbox" name="chkOnYList" tabIndex=20 title="Markera om årsbesökaren inte skall visas på välkomstlista.">
                                <%end if%>
                                    Ej på välkomstlista
                                </TD>
                            </TR>
                        </TABLE>
                    </TD>
                    <TD>
                        <%'Buttons.%>
                        <TABLE width="<%=tbwidth1%>" border="<%=border%>">
                            <TR>
                                <TD width="50%">
                                    <INPUT class="button" onclick="NewYearApproved()" type="button" value="Ny" name="cmdNew" tabIndex=22 title="Rensa formuläret.">
                                </TD>
                                <TD width="50%" align=right>
                                    <INPUT class="button" type="submit" value="<%=AddUpdateText%>" name="cmdAddUpdate" tabIndex=23 title="<%=AddUpdateText%>">
                                </TD>
                            </TR>
                        </TABLE>
                    </TD>
                </TR>
            </TABLE>
            </FORM>
            <BR>
            <%'Pageinfo text.%>
            <TABLE width="<%=tbwidth2%>">
                <TR>
                    <TD class="pageinfo" align="right">
                        Befintliga
                    </TD>
                </TR>
            </TABLE>
            <%'Include list with year approved people.%>
            <!--#include file="year_approved_list.inc"-->
            <%'Link to Menu%>
            <BR>
            <TABLE width="<%=tbwidth2%>">
                <TR>
                    <TD class="pageinfo" align="left">
                        <A href="menu.asp">Meny</A>
                    </TD>
                </TR>
            </TABLE>
        </TD>
    </TR>
</TABLE>
</BODY>
</HTML>
<%'--------------------------------------------------------------------------------%>

<%'End of page. Close everything.
    '-------------------------------------------------------------------------------------%>
    <!--#include file="end_of_page.inc"-->
<%'-------------------------------------------------------------------------------------%>


<% 'year_approved_check.inc %>
function ValidateForm(form)
{
    if(form.txtFirstName.value=="")
    {
        alert('Du måste ange ett förnamn.');
        form.txtFirstName.focus();
        return false;
    }

    if(form.txtLastName.value=="")
    {
        alert('Du måste ange ett efternamn.');
        form.txtLastName.focus();
        return false;
    }

  if(form.txtYear.value=="")
    {
        alert('Du måste ange ett år.');
        form.txtYear.focus();
        return false;
    }

    if(form.txtRecipientName.value=="")
    {
        alert('Du måste ange en besöksmottagare.');
        form.txtRecipientName.focus();
        return false;
    }

    return true;
}
<%'--------------------------------------------------------------------------------%>


<%@ Language=VBScript %>
<%  'year_approved_data.asp
    'Last updated: 2003-11-14
    'Name: Anders o Jonas%>
<%on error resume next%>

<%'Function for retrieving user name.
    '-------------------------------------------------------------------------------------%>
    <!--#include file="getuser.inc"-->
<%'-------------------------------------------------------------------------------------%>

<%'Make database connection
    '-------------------------------------------------------------------------------------%>
    <!--#include file="database.inc"-->
    <!--#include file="recordset.inc"--><%'Create recordset%>
<%'-------------------------------------------------------------------------------------%>

<%'General
    '--------------------------------------------------------------------------------
    dim YADLastVisitorID, YADUserName, YADUserNameID, VisitorID
  VisitorID=clng(request.querystring("visitorID"))
    '-------------------------------------------------------------------------------------%>

<%'Get userid from table Users
    '--------------------------------------------------------------------------------
    YADUserName=GetUser
    if YADUserName="" then
        Response.Redirect("error.asp?err=40&pageref=year_approved_data.asp")
    end if

    rs.Open "select UserID from Users where UserSignature='" & YADUserName & "';"
    YADUserNameID=rs.Fields("UserID")
    rs.Close
    '--------------------------------------------------------------------------------    %>

<%'Add data to table "Visitors"
    '--------------------------------------------------------------------------------
    rs.Open "select * from Visitors where VisitorID=" & VisitorID & ";"
    if VisitorID=-1 then rs.AddNew

    rs.Fields("VisitorFname")=Request.Form("txtFirstName")
    rs.Fields("VisitorLname")=Request.Form("txtLastName")
    rs.Fields("VisitorSnum")=Request.Form("txtSNum")
    rs.Fields("VisitorCompany")=Request.Form("txtCompany")
    rs.Fields("VisitorApproved")=1
    rs.Fields("CitizenshipID")=1
    rs.Fields("VisitorErrend")="Årsbesökare"
    rs.Fields("VisitorVisitingArea")=Request.Form("txtVisitingArea")

    if Request.Form("chkCard")="" then
        rs.Fields("CardID")=1
    else
        rs.Fields("CardID")=2
```

```
    end if

    if Request.Form("chkOnYList")="" then
        rs.Fields("VisitorOnWelcomeList")=1
    else
        rs.Fields("VisitorOnWelcomeList")=0
    end if

    'Year visitor (2)
    rs.Fields("VisitorTypeID")=2
    rs.Update

    'Get ID of last visitor.
    rs.MoveLast
    YADLastVisitorID=rs.Fields("VisitorID")
    rs.Close
    '--------------------------------------------------------------------------------%>

<%'Add data to table "Visits"
    '--------------------------------------------------------------------------------

    if VisitorID=-1 then
        rs.Open "select * from Visits where VisitID=-1;"
        rs.AddNew
        rs.Fields("VisitorID")=YADLastVisitorID
        rs.Fields("VisitActive")=0
    else
        rs.Open "select * from Visits where VisitorID=" & VisitorID
    end if

    rs.Fields("UserID")=YADUserNameID
    rs.Fields("VisitRecipientName")=Request.Form("txtRecipientName")

    rs.Update
    rs.Close
    '--------------------------------------------------------------------------------%>

<%'Add data to table "PropertyValue"
    '--------------------------------------------------------------------------------

    if VisitorID = -1 then
        rs.Open "select * from PropertyValue where PropertyValueID=-1;"

        rs.AddNew

        rs.Fields("VisitorPropertyID")=1

        if Request.Form("chkOnYList")="" then
            rs.Fields("PropertyValue")=0
        else
            rs.Fields("PropertyValue")=1
        end if

        rs.Fields("VisitorID")=YADLastVisitorID

        rs.Update
        rs.AddNew

        rs.Fields("VisitorPropertyID")=2
        rs.Fields("PropertyValue")=Request.Form("txtYear")
        rs.Fields("VisitorID")=YADLastVisitorID

        rs.Update
        rs.AddNew

        rs.Fields("VisitorPropertyID")=3
        rs.Fields("PropertyValue")=1
        rs.Fields("VisitorID")=YADLastVisitorID

        rs.Update

        rs.Close
    else
        rs.Open "select * from PropertyValue where VisitorID=" & VisitorID & " and VisitorPropertyID=2;"

        rs.Fields("PropertyValue")=Request.Form("txtYear")
        rs.Update
        rs.close
    end if
    '--------------------------------------------------------------------------------%>

<%'Redirect when done.
    '--------------------------------------------------------------------------------
    if err.number<>0 then
         Response.Redirect("error.asp?err=30&pageref=year_approved_data.asp")
    else
          Response.Redirect("year_approved.asp?visitorID=" & "&action=0")
    end if
    '--------------------------------------------------------------------------------%>

<%'End of page. Close everything.
    '--------------------------------------------------------------------------------%>
    <!--#include file="end_of_page.inc"-->
<%'--------------------------------------------------------------------------------%>

<% 'year_approved_list.inc %>
<%  'This page require:
    'Database connection
    'Recordset
%>

<%dim YALSQL, YALrowalt
YALrowalt=0

YALSQL= "SELECT Pv2.PropertyValue, Visitors.VisitorID, Visitors.VisitorLoggedIn, Visitors.VisitorFname, Visitors.VisitorLname, Visitors.VisitorCompany, Visitors.VisitorSnum" &_
               " FROM VisitorProperty AS Vp, PropertyValue AS Pv2, VisitorProperty AS Vp2, PropertyValue AS Pv INNER JOIN Visitors ON" &_
               " Pv.VisitorID = Visitors.VisitorID WHERE (((Pv.VisitorPropertyID)=[Vp].[VisitorPropertyID]) AND ((Vp2.VisitorPropertyID)=[Pv2].[VisitorPropertyID]))" &_
               " AND ((Pv2.VisitorID)=[Pv].[VisitorID]) AND ((Vp.VisitorProperty)='YearActive') AND ((Pv.PropertyValue)='1') AND ((Vp2.VisitorProperty)='YearValid')) ORDER BY Visitors.
                   VisitorLname;"

rs.Open YALSQL %>
<TABLE class="table" border="1" cellspacing="0">
    <TR>
        <TD>
            <TABLE width="<%= tbwidth2 %>" border="<%= border %>" cellspacing="0">
                <TR>
                    <TD width="25%" class="headline">
                        Besökare
                    </TD>
                    <TD width="20%" class="headline">
                        Organisation
                    </TD>
                    <TD width="25%" class="headline">
                        Personnummer
                    </TD>
                    <TD width="10%" class="headline">
                        År
                    </TD>
                    <TD width="20%" class="headline">
                         
                    </TD>
                </TR>

                <%if err.number=0 then
                    do while not rs.EOF
                        if YALrowalt=0 then%>
                <TR>
                    <TD class="rowalt3">
                            <%=rs.fields("VisitorFname")%> <%=rs.fields("VisitorLname")%>
                    </TD>
                    <TD class="rowalt3">
                        <%=rs.fields("VisitorCompany")%>
                    </TD>
                    <TD class="rowalt3">
                        <%=rs.fields("VisitorSnum")%>
                    </TD>
                    <TD class="rowalt3">
                    <%=rs.fields("PropertyValue")%>
                    </TD>
                    <TD align="right" class="rowalt3">
                        <INPUT class="minibutton" onclick="LoadYearApproved('<%=rs.fields("VisitorID")%>')" type="button" value="Redigera" name="Edit">
                        <% if rs.Fields("VisitorLoggedIn")=1 then %>
                            <INPUT class="minibutton" onclick="DeleteYearApproved('<%=rs.fields("VisitorID")%>','<%=rs.fields("VisitorFname")%>' + ' ' +'<%=rs.fields("VisitorLname")%>')"
                                type="button" value="Ta bort" name="Delete" disabled title="Du kan ej ta bort en person som är inloggad.">
                        <%else%>
                            <INPUT class="minibutton" onclick="DeleteYearApproved('<%=rs.fields("VisitorID")%>','<%=rs.fields("VisitorFname")%>' + ' ' +'<%=rs.fields("VisitorLname")%>')"
                                type="button" value="Ta bort" name="Delete">
```

```
                        <%end if%>
                    </TD>
                </TR>

                <%  YALrowalt=1
                else%>
                <TR>
                    <TD class="rowalt1">
                        <%=rs.fields("VisitorFname")%> <%=rs.fields("VisitorLname")%>
                    </TD>
                    <TD class="rowalt1">
                        <%=rs.fields("VisitorCompany")%>
                    </TD>
                    <TD class="rowalt1">
                        <%=rs.fields("VisitorSnum")%>
                    </TD>
                    <TD class="rowalt1">
                        <%=rs.fields("PropertyValue")%>
                    </TD>
                    <TD align="right" class="rowalt1">
                        <INPUT class="minibutton" onclick="LoadYearApproved('<%=rs.fields("VisitorID")%>')" type="button" value="Redigera" name="Edit">
                        <INPUT class="minibutton" onclick="DeleteYearApproved('<%=rs.fields("VisitorID")%>','<%=rs.fields("VisitorFname")%>' + ' ' +'<%=rs.fields("VisitorLname")%>')" type
                            ="button" value="Ta bort" name="Delete">
                    </TD>
                </TR>

                <%YALrowalt=0
                end if
                rs.movenext
                loop
                rs.close
                else
                rs.close
                Response.Redirect("error.asp?err=20&pageref=year_approved_list.asp")%>
                <%end if%>
            </TABLE>
        </TD>
    </TR>
</TABLE>

<%'End of page. Close everything.
    '--------------------------------------------------------------------------------
    set rs=nothing
    cn.Close
    set cn=nothing
    '--------------------------------------------------------------------------------%>


<%@ Language=VBScript %>
<%  'year_approved_remove.asp
    'Last updated: 2003-11-14
    'Name: Anders o Jonas%>
<%on error resume next%>

<%'Function for retrieving user name.
    '--------------------------------------------------------------------------------%>
    <!--#include file="getuser.inc"-->
<%'--------------------------------------------------------------------------------%>

<%'Make database connection
    '--------------------------------------------------------------------------------%>
    <!--#include file="database.inc"-->
    <!--#include file="recordset.inc"--><%'Create recordset%>
<%'--------------------------------------------------------------------------------%>

<%'General
    '--------------------------------------------------------------------------------
    dim YARLastVisitorID, YARUserName, YARUserNameID

    '--------------------------------------------------------------------------------%>

<%'Get userid from table Users
    '--------------------------------------------------------------------------------
    YARUserName=GetUser
    if YARUserName="" then
        Response.Redirect("error.asp?err=40&pageref=year_approved_data.asp")
    end if

    rs.Open "select UserID from Users where UserSignature='" & YARUserName & "';"
    YARUserNameID=rs.Fields("UserID")
    rs.Close
    '--------------------------------------------------------------------------------    %>

<%'Add data to table "PropertyValue"
    '--------------------------------------------------------------------------------
    YARSQL="select * from PropertyValue where VisitorID=" & request.querystring("visitorID") & "and VisitorPropertyID=3" & ";"
    rs.Open YARSQL
    rs.Fields("PropertyValue")=0
    rs.Update
    rs.Close
    '--------------------------------------------------------------------------------%>

<%'Redirect when done.
    '--------------------------------------------------------------------------------
    if err.number<>0 then
        Response.Redirect("error.asp?err=30&pageref=year_approved_remove.asp")
    else
        Response.Redirect("year_approved.asp")
    end if
    '--------------------------------------------------------------------------------%>

<%'End of page. Close everything.
    '--------------------------------------------------------------------------------%>
    <!--#include file="end_of_page.inc"-->
<%'--------------------------------------------------------------------------------%>
```