Computer Science

Danial Abdali

Bernes Maksumic

EuroBoss Information System (EBIS)

Bachelor's Project

2002:24

This report is submitted in partial fulfilment of the requirements for the Bachelor's degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Danial Abdali / Bernes Maksumic

Approved, 2002-06-05

Advisor: Donald Ross

Examiner: Donald Ross

Sammanfattning

Denna rapport är en C-uppsats på 10 poäng i ämnet Datavetenskap på Karlstads Universitet. Vår uppdragsgivare är EuroBossClass AB, ett relativt nystartat resebyrå företag med sin bas i Stockholm. Företaget genomgår en förändring och har planer att expandera sin verksamhet p.g.a. allt större kundefterfrågan.

Målet med detta arbete är att designa och implementera ett bokningssystem som ska fungera som en webbsida, där endast företagets personal och andra behöriga har tillgång till det. Systemet ska ha ett webb-baserat gränssnitt och ska vara lätt tillgängligt och gränssnittet ska vara tydligt. Dokumentationen här beskriver hur vårt arbete och utvecklingsprocessen för denna prototyp har förlöpt.

EuroBoss Information System (EBIS)

Abstract

This document is a Bachelor's thesis, 10 points in Computer Science at Karlstad University. This project has been undertaken at the request of EuroBossClass AB, a relatively new travel company based in Stockholm. The company is expanding due to the increasing demand from the customers.

The main goal of the project is to design and implement a booking system, which would work as a web site for this travel company. This system will be only available for staff working within the company and other authorised individuals. This dissertation describes the prototype system for the company.

Innehållsförteckning

Figurförteckning

Tabellförteckning

Bilaga förteckning

1 Introduktion

Uppgiften för vårt examensarbete är att konstruera ett webb-baserat bokningssystem och denna dokumentation beskriver hur vårt arbete och utvecklingsprocessen har förlöpt. Vi inleder med att presentera bakgrunden till arbetet och de krav som vår uppdragsgivare EuroBossClass AB har ställt, samt eventuella justeringar och ändringar under arbetets gång. Därefter ska vi klargöra vilka program som vi har använt oss av och en kort beskrivning av hur dessa komponenter fungerar ihop. Dessutom ska vi redovisa för andra alternativa program/komponenter som finns att tillgå. Vidare beskrivs de olika stegen i utvecklingsarbetet i form av skapandet av databas, användargränssnitt och implementering. Vi avslutar vår presentation med en slutsummering, där vi diskuterar kring de slutsatser och erfarenheter som har kommit fram under arbetets gång och eventuella problem som har uppstått under denna tid.

Rapporten är strukturerad som följande:

- **Kapitel 1** Bakgrundsfakta om vår uppdragsgivare EuroBossClass AB
- **Kapitel 2** Resonemang kring valet av komponenter och verktyg, tillsammans med en kort beskrivning av hur de olika komponenterna fungerar och några alternativ till dem.
- **Kapitel 3** Här beskrivs de olika stegen i utvecklingsarbetet och funderingar kring skapandet av databasen.
- **Kapitel 4** Beskriver hur arbetet med användargränssnittet och implementering av systemet har fortlöpt.
- **Kapitel 5** Innehåller värdering om det genomförda arbetet och funderingar kring projektet.
- **Kapitel 6** Innehåller slutsatser som kommit fram under arbetets gång.

1.1 Bakgrund

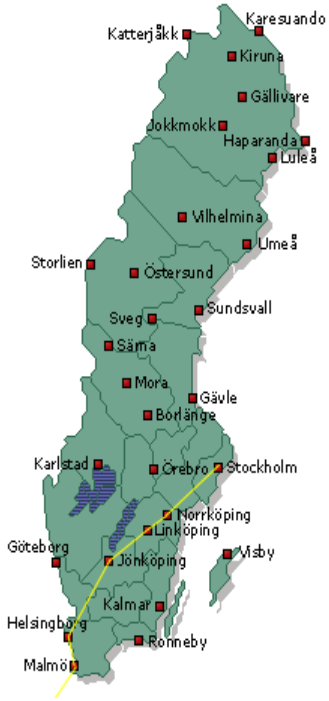
1.1.1 Företaget EuroBossClass AB

EuroBossClass AB är ett resebyrå företag som skapades för 5 år sedan och som erbjuder bussturer från Sverige till Bosnien-Hercegovina. De har för tillfället en buss som kör reguljärt fram och tillbaka mellan de två länderna.

Figur 1.1 Färd väg

Bilden nedan illustrerar de olika städer inom Sverige från vilka passagerare kan påbörja eller avsluta sin resa:

- Stockholm
- Södertälje
- Nyköping
- Norrköping
- Linköping
- Jönköping
- Värnamo
- Ljungby
- Helsingborg
- Landskrona
- Malmö



Figur 1.2 Karta över destinationspunkter inom Sverige

Samma möjlighet ges självfallet till passagerare som befinner sig i Bosnien-Hercegovina. EuroBossClass är det enda resebyrån för tillfället inom Sverige som kan erbjuda sådana turer för ett rimligt pris.

Nedan följer en karta på de olika städer inom Bosnien-Hercegovina som företaget erbjuder turer till och från:

• Bihac • Bosanski Petrovac • Kljuc • Jajce • Mrkonjic • Donji Vakuf • Travnik • Zenica • Sarajevo	

Figur 1.3 Karta över destinationspunkter inom Bosnien-Hercegovina

Eftersom efterfrågan på billiga resor är så stor så tänker företaget expandera och har planer att skaffa sig ytterligare en buss för att bemöta kundernas krav och på så sätt utvidga sin verksamhet ännu mera. Detta måste vi ta hänsyn till när vi konstruerar deras bokningssystem.

Det huvudsakliga problemet idag är att företagets personal saknar ett enhetligt system för reservation av biljetter. Kontoret i Sarajevo respektive Stockholm måste hela tiden faxa fram och tillbaka uppdaterad information om de senaste biljett beställningarna. Detta är mycket tidskrävande och ineffektivt. Med en ökad kundkrets så har företaget allt större behov av ett enhetligt bokningssystem. Systemet ska vara lätttillgängligt och gränssnittet ska vara tydligt så att personalen lätt kan guida sig genom de olika fönstren.

1.2 Syfte

Syftet med vårt arbete är att designa och implementera ett bokningssystem som ska fungera som en webbsida, där endast företagets personal och andra behöriga har tillgång till. Kundregistret finns för närvarande som en Access databas, vilket vi har tillgång till. Den innehåller information om bussresor samt kunder.

Från databasen ska information om kunderna hämtas (om kunden redan existerar i databasen) och sedan ska platsen bokas för denna person vid en viss avgångstid.

1.3 Förutsättningar och krav

Här beskrivs de krav som vår uppdragsgivare EuroBossClass AB har ställt på oss, samt förutsättningarna under vilket programmet skall fungera. En del av kraven är förhandlingsbara, då vi själva har givits fria händer att arbeta på bästa möjliga sätt medan andra är fastställda och kan ej bestämmas av oss.

1.3.1 Förutsättningar

I inledningsskedet av arbetet var det svårt att få någon klar bild av hur vårt system skulle se ut. Men med hjälp av de direktioner och krav som EuroBossClass presenterade blev bilden klarare och vi fick en bättre insikt i hur arbetet skulle utformas.

1.3.2 Krav

Vägen till att få fram en kravspecifikation som båda parterna, alltså vi som uppdragstagare och EuroBossClass som uppdragsgivare var lång och besvärlig. Vi kände att de krav som ställdes på oss var orimliga och ej realistiska. Vi hade gång på gång förklarat för EuroBossClass att vi hade för lite tid för att kunna både skapa ett fungerande system som är tillförlitligt och ett system som skulle vara färdigutvecklat. Tyvärr så hade de i början mycket svårt att förstå vår resonemang och situationen som befann vi oss på, men efter en tid av diskussioner, förhandlingar och överenskommelser kunde vi enas om en kravspecifikation som både var rimligt och genomförbar.

De krav som vår uppdragsgivare har ställt på oss är klara och fastställda i det mån att de vill ha ett system som är enkelt att utöka med nya funktioner. Vi måste designa vårt

bokningssystem utifrån den befintliga databasen som företaget hade redan skapat sedan förut. En av de viktigaste kraven var att vi skulle bibehålla information som de redan hade i databasen och inte förändra dess innehåll. Anledningen till detta var att även kontoret i Bosnien-Hercegovina skulle använda sig av samma databas.

Det fanns också önskemål från vår uppdragsgivare att vi skulle utrusta vårt system med funktioner för hantering av paketleverans och priser. De hade en tanke om att vi skulle dela in de befintliga busstationerna i zoner och utifrån detta räkna ut priser för dessa zoner. Andra önskemål är att man skall kunna skapa nya färdvägar, dvs helt nya resmål (till exempel till Oslo) med nya busstationer (till exempel Örebro, Karlstad, Årjäng). Man skall även kunna ändra och avboka en biljett för en vald kund.

Dessa krav är inte primära i utformningen av vårt system, utan vi har som mål att bygga systemet på så sätt att det uppfyller de primära målen som är uppsatta från början och om det ges tid och utrymme kommer vi att försöka hjälpa EuroBossClass med vidareutveckling av system efter att vi har slutfört denna rapport.

När det gäller val av program har vi givits fria händer att välja från de diverse tekniker som finns att tillgå. Självfallet har vi jämfört de olika verktygen och valt den som passar vår projekt bäst. Vi kommer att beskriva vår val av program senare i rapporten (**se avsnitt 2.1 Resonemang**) och även argumentera för varför vi anser att den passar just vårt arbete.

1.3.3 Information om vad systemet skall innehålla

Resultatet av våra möten och de diskussioner som fördes på de möten kom att ligga som grund för hur vårt system skall se ut och vilka funktioner som systemet skall innehålla.

Systemet skall alltså bestå av menyer innehållande all information som är relevant för reservation av resa för resenär.

Nedan följer kraven som EuroBossClass har ställt:

Primär mål:

- Det ska på ett enkelt sätt gå att registrera en ny avgång för bussen.
- Genom att välja den registrerade avgången ska man på ett enkelt sätt kunna lägga till passagerare för denna avgång.

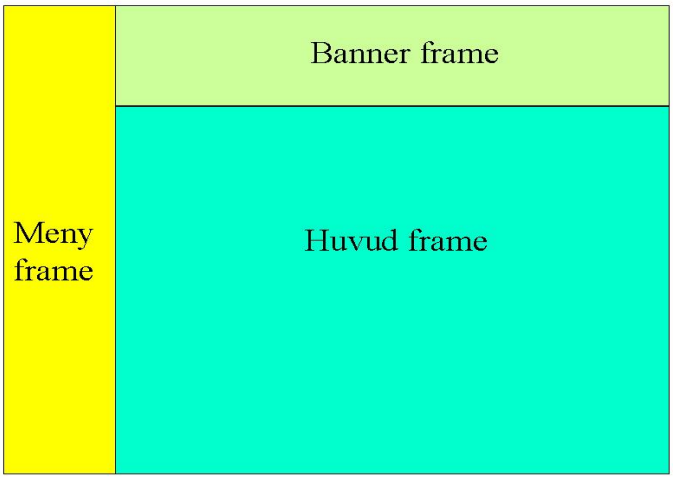
- Om passageraren finns med i databasen sedan tidigare, dvs om passageraren har rest med EuroBoss förut skall man kunna hitta honom/henne i databasen med en sök funktion.
- Finns passageraren inte med i databasen ska man kunna registrera honom/henne som en ny passagerare i databasen.
- Det ska gå att boka en sittplats i bussen åt en vald kund. En redan upptagen plats måste vi markera på något sätt och ”spärra” så att det inte går att boka en sittplats två gånger.
- Det ska på ett enkelt sätt gå att registrera en eller flera nya bussar.

Sekundär mål:

- Det ska även gå att skapa nya färdvägar som innehåller nya busstationer.
- Ändring och avbokning av biljetter för specifik kund ska tillåtas.
- Hantering av paketleverans. (i mån av tid)
- Programmet ska även kunna räkna ut priser för resa inom de olika zonerna. (i mån av tid)

1.3.4 Gränssnittskrav

Vårt systemet ska ha ett webb-baserat gränssnitt. Systemet ska vara lätttillgängligt och gränssnittet ska vara tydligt. Alla fönster ska ha beskrivande text som hjälper användaren. Användargränssnittet ska vara grafiskt med ett huvudfönster indelad i tre frames (banner, menyframe och huvudframe). Gränssnittet kommer att vara konstruerad på så sätt att alla nya fönster som öppnas skall öppnas i den dynamiska *Huvud frame*, medan *Meny frame* och *Banner frame* är oförändrade. Uppdelningen av huvudfönstret kommer att förbli detsamma genom hela programmet.



Figur 1.4 Uppdelning av huvudfönstret i frames.

Användaren ska kunna på ett smidigt sätt navigera sig genom de olika fönstren genom att klicka på knapparna. När det gäller utseende, färg och ramindelning (frames) har vi fått fria händer. Enda kravet är att vi ska skapa en så homogen miljö så möjligt. Med detta menar de att användaren ska kunna känna en viss trygghet och harmoni när de navigerar sig genom de olika meny valen som finns att tillgå.

1.3.5 Avgränsning

För närvarande finns det mycket som tyder på att företaget kommer att ställa fler krav på systemet än de som är fastställda i nuläget. Vi som har tagit oss an detta arbete kommer säkerligen inte kunna uppfylla alla kraven och därför har vi förklarat för dem att en eventuell vidareutveckling av systemet är endast möjligt för oss efter att vi har skapad grunden för vårt system. Hantering av paketleverans och prissättning grundad på zoner, är några av de önskemål som har förts fram av vår uppdragsgivare. Dessa krav koncentrerar vi oss inte på i första hand, då vi vill att projektet ska förbli hanterbart.

Vi vill alltså i första hand koncentrera oss på att göra ett bokningssystem som klarar av att söka eller lägga till en ny passagerare till databasen. Systemet ska dessutom klara av bokning av en plats åt passageraren vid en angiven avgångstid. Man ska också på ett enkelt sätt kunna registrera en ny avgång (alltså datum för en ny resa) och nya bussar (ifall företaget köper fler bussar).

1.4 Design aspekter

Ett av de första problemen var att utforma en entitet/relation modell (E/R-modell). Det var ett mycket intressant designfråga att veta vilka entiteter med respektive attribut man ska ta med, hur man ska koppla ihop de olika tabellerna och bestämma relationerna mellan tabellerna. En annan aspekt gäller organisation, avgränsning och uppdelning av arbetet.

Andra designfrågor som dök upp tidigt i projektet var hur vi skulle lägga upp våran kod på nätet, eftersom ASP koden exekveras där.

Om man skulle lägga den på en webbhotell så måste man konstant skicka den modifierade koden till servern via något (FTP) File Transfer Protocol program. Detta skulle göra felsökningen mycket svårare och dessutom ta mycket tid.

Då kom vi på att vi kunde använda våran egen dator som server och exekvera koden på den. Vi installerade PWS (Personal Web Server) från Microsoft och skapade en DSN (Data Source Name) på samma dator. På så sätt kunde vi nu exekvera programmet på en ”virtuell server”, vilket sparade mycket tid åt oss och dessutom gjorde felsökningen mycket lättare.

1.5 Summering

I ovanstående kapitel har vi presenterad fakta om vår uppdragsgivare EuroBossClass AB och anledningen till varför de är i behov av ett enhetligt system för hantering av bokning av biljetter. Syftet med vårt arbete är alltså att konstruera och implementera ett bokningssystem som ska fungera som en hemsida, där endast företagets personal och andra behöriga har tillgång till.

Dessutom har vi format fram en kravspecifikation med hjälp av EuroBossClass, där vårt arbete är delat in i två separata delar. Den primära delen, vilket är den delen av projektet som vi kommer att lägga all energi och uppmärksamhet på består av de grundläggande funktionerna för hantering och registrering av nya kunder och reservation av biljetter för en specifik kund. Den andra delen av arbetet som består av ytterligare funktioner för hantering av paketleverans, priser och andra funktioner som vi kommer att ta i tu med i mån av tid.

De kriterier som EuroBossClass har på systemets användargränssnitt är att den ska vara webb-baserat. Systemet ska vara lätt tillgängligt och gränssnittet ska vara tydligt. Alla fönster ska ha beskrivande text som hjälper användaren. Användaren, vilket är anställda på företaget ska kunna på ett smidigt sätt navigera sig genom de olika fönstren genom att klicka på knapparna som finns att tillgå.

Andra design aspekter som diskuterades under de möten som vi hade angående vårt gränssitt var utseende, färg och ramindelningen på sidorna. När det gäller ramindelning har vi som krav att använda oss av tre frames, bestående av ett *Huvud* frame, *Banner* frame och ett *Meny* frame.

De vill även ha ett användargränssnitt, där användaren ska kunna känna en viss trygghet och harmoni när de navigerar sig genom de olika meny valen som finns att tillgå.

2 Bakgrundsfakta

Det här kapitlet beskriver resonemanget bakom val av teknik och programmeringsspråk för tillämpning av vår dynamiska databas och några alternativa lösningar till den teknik som vi slutligen har valt för vårt arbete.

När man ska göra en dynamisk hemsida så har man en rad programmeringsspråk att välja mellan, fast ASP och CGI är de vanligaste och mest förekommande [1].

2.1 Resonemang

Det system som vi ska designa och implementera ska fungera som en dynamisk (dokument som förändras över tiden) databas för samling och presentation av inmatad information. I samråd med vår uppdragsgivare, EuroBossClass kom vi fram till att vi skulle använda oss av ASP-tekniken (se avsnitt 2.1.1) som för tillfället har på ett revolutionerande sätt gjort det möjligt för många privatpersoner och annat expertis att skapa riktigt avancerade webbplatser på lättast möjliga sätt.

ASP är inte i sig ett renodlat programmeringsspråk utan en lös samling objekt som kan kommunicera med servern och användarens webbläsare. Själva koden som styr objekten har vi valt att skriva i skript-språket VBScript, men man kan lika gärna använda Java-Script eller Jscript [3].

2.1.1 Vad är ASP?

ASP, som står för Active Server Pages, är en teknologi utvecklad av Microsoft för att kunna baka in programmeringskod, så kallad ASP-kod, i HTML-koden.

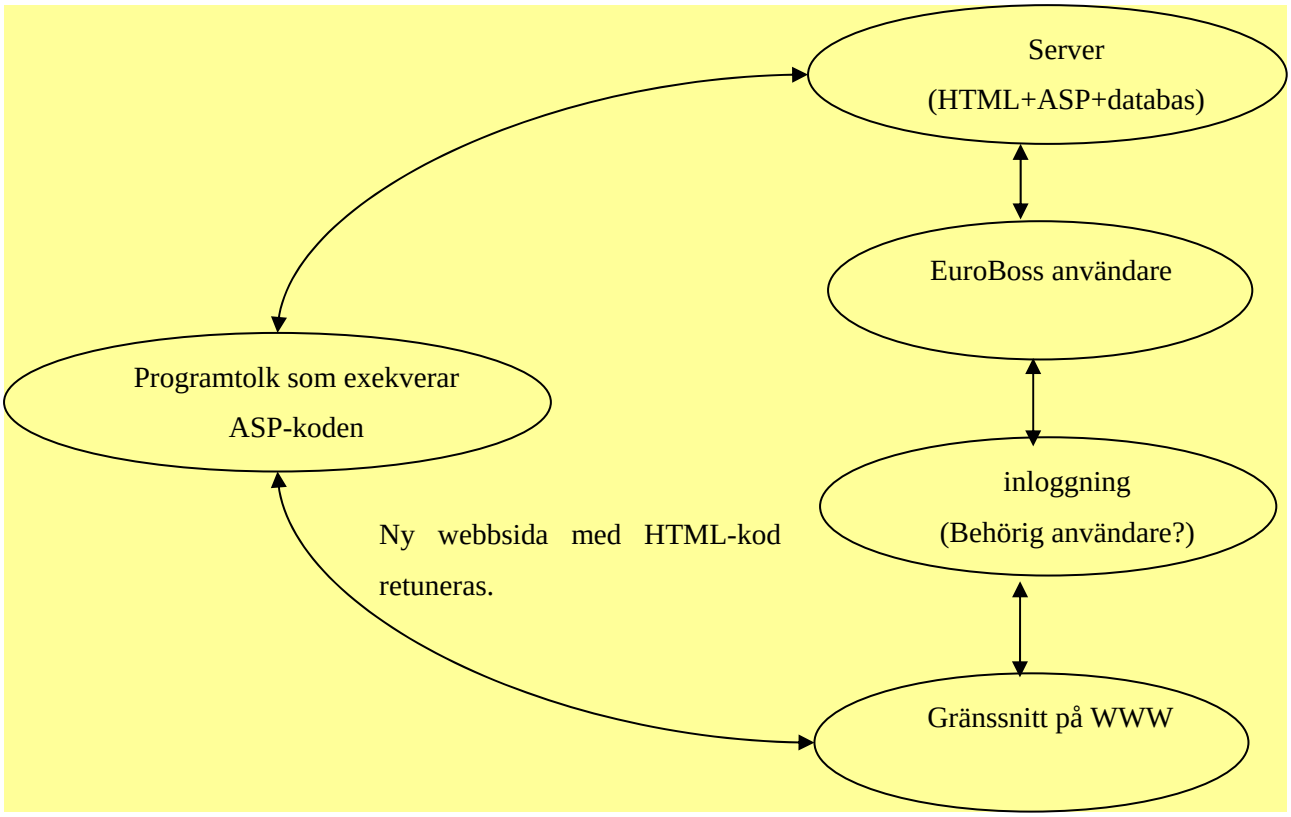
Poängen med detta är att man kan gör webbsidan dynamisk, det vill säga när olika användare vid olika tidpunkter laddar ner webbsidan kommer olika resultat att visas.

Ett enkelt exempel på en dynamisk webbsida är en webbsida som returnerar den aktuella tiden och datumet. Om man gör en sådan webbsida i HTML så kommer till exempel klockan bara att stämma en gång per dygn. Skillnaden med ASP är du kan lägga in ett kommando i HTML-koden som skriver om HTML-koden i just det ögonblicket webbsidan laddas ner, så att den aktuella tiden skrivs in. På det viset blir webbsidan dynamisk.

2.1.2 Hur fungerar ASP?

Som vi nämnde ovan så bakas alltså ASP-koden in i våran HTML-kod. För att markera att man har bakat in ASP-kod i HTML-koden så ska filen ha filtillägget ASP istället för HTML.

När sedan ASP webbsidan anropas via www, så är det inte ASP koden som sidan är kodad med som skickas tillbaka till användarens webbläsare. ASP-koden skickas först till en programtolk på webbservern som exekverar koden i samma ögonblick som webbsidan anropades. Beroende på resultatet av användarens krav så skriver programtolken en helt ny webbsida i ren HTML-kod utan ASP-kod, och det är just denna HTML-kod som skickas tillbaka till användaren. På så sätt kommer aldrig användaren att se ASP-koden, utan han ser den HTML-kod som programolken gjorde om som resultatet på användarens begäran [1].



Figur 2.1 Illustration på hur våran ASP kod kommer att exekveras på servern.

Viktigt att upprepa är att ASP är inte ett renodlad programmeringsspråk utan en lös samling objekt som kan kommunicera med servern och användarens webbläsare. ASP är uppbyggd av sex så kallade objekt, alla för olika användningsområden. Objekten innehåller i sig ett stort antal metoder och egenskaper, och det är dessa man arbetar med.

Här är en sammanställning över objekten:

• Application	Används för att dela information mellan flera olika användare.
• Request	Används för all kommunikation från användaren.
• Response	Används för all kommunikation till användaren.
• Server	Används för att importera externa objekt.
• Session knuten	Används för att temporärt spara information till en viss användare.
• ObjectContext	Används enbart för interaktion med Microsoft Transaction Center.

Tabell 2.1 Sammanställning över objekten

Själva programmeringskoden, alltså den som styr objekten, kan man skriva i valfritt skriptspråk.

Microsoft rekommenderar det egna skriptspårket, men man kan också använda Java-Script, Jscript och till och med Perl, fast om man använder Perl så krävs det att den ansvariga för servern installerar ett speciellt tilläggsprogram för Perl.

VBScript är standardspråket (default) för ASP, och så länge man använder VBScript så behöver man egentligen inte ange vilket språk ASP-koden är skriven i.

2.1.3 Vad är VBScript?

VBScript är en förenklad skriptversionen av programmeringsspråket Visual Basic, utvecklad av Microsoft. De flesta av Microsofts program som till exempel Officepaketet förstår VBScript och det är just VBScript som används i bakgrunden då man själv skapar makron i till exempel Microsoft Word och Microsoft Word.

Precis som Officepaketet så förstår även Explorer VBScript. Det gör att man kan lägga in små programslingor direkt i HTML-koden i en så kallad SCRIPT-container.

Exempel:

```
<SCRIPT LANGUAGE=”VBScript”>
<!--
[Här ligger VBScript-koden]
-->
</SCRIPT>
```

I syntaxen ovan specifierar man med attributet LANGUAGE, vilket programmeringsspråk, skriptet är skrivet i. Här kan man bland annat skriva:

```
<SCRIPT LANGUAGE=”JavaScript”>
<SCRIPT LANGUAGE=”JScript”>
```

Beroende på vilket programmeringsspråk man vill använda, JavaScript eller Jscript.

Ett bra regel är att man lägger så mycket av SCRIPT-containern i HEAD-containern som möjligt. Det för att vara säker på att webbläsaren har hunnit ladda ner skriptet innan den hinner anropas [2].

2.1.4 Skillnaden mellan VBScript och ASP med VBScript

Den stora skillnaden mellan ett renodlat VBScript och ASP, förutom syntaxen, är att ASP-koden exekveras på webbservern medan VBScriptet skickas med HTML-koden till användaren och exekveras lokalt av användarens webbläsare.

VBScript är utvecklat av Microsoft och det är enbart program och webbservrar utvecklade av Microsoft som förstår VBScript. Netscape den andra stora webbläsaren förstår sig inte på VBScript, vilket är ett aktivt ställningstagande från deras sida. Detta innebär att renodlat VBScript direkt inbakad i HTML.kod har ett mycket begränsat värde [6].

2.2 Alternativa lösningar

I kapitlet beskrivs alternativ lösning till ASP som finns att tillgå och som fungerar lika bra som ASP.

2.2.1 Vad är CGI?

CGI står för Common Gateway Interface och är ett "gränssnitt" eller en "gränsyta". Det ser till att information som har matats in på något sätt (t.ex. ett formulär) kan föras från servern till serverprogram som tillhandahåller gemensamma servicefunktioner i ett datornät, och tvärtom.

CGI gör det möjligt för webb-läsare att kommunicera med scripts på servern. Information från och till webbsidan överförs, bearbetas av CGI-scriptet och presenteras för webbläsaren i form av en HTML-sida. Alltså är tekniken mycket lik ASP-tekniken.

CGI handlar precis som ASP om att göra en webbsida dynamisk. Det gör den genom att låta webbsidan kommunicera med CGI-program som ligger på webbservern.

2.2.2 Hur fungerar CGI?

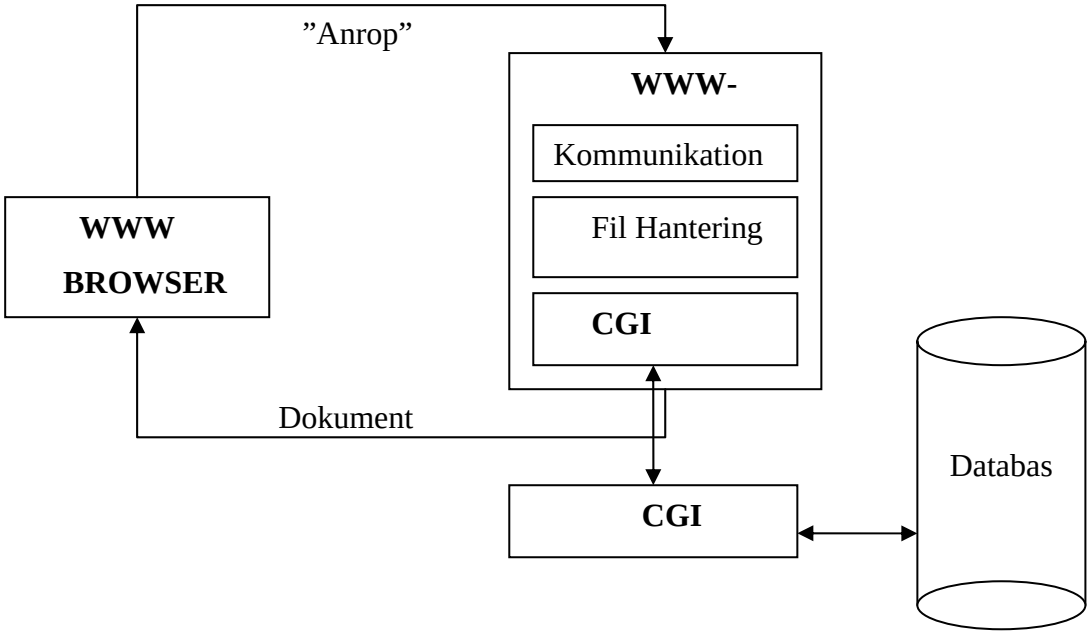
CGI fungerar enligt följande:

1. Man begär från sin webbläsare att få köra ett visst program kallat CGI-program på servern. Med begäran kan man skicka egen information som skall behandlas av CGI-programmet.

2. Servern tar emot användarens begäran, startar CGI-programmet, och skickar vidare den information man sänt till detta.

3. CGI-programmet utför sin uppgift, och genererar utdata, som skickas in i servern igen.

4. Servern skickar vidare den nya informationen tillbaka till webbläsaren som en HTMLsida.



Figur 2.2 Principen för hur CGI-tekniken fungerar

CGI-scripts är skrivna i något programmeringsspråk. Vilket språk man skall använda beror på vilken plattform man använder och vilken serverprogramvara man kör. På PC-plattformar brukar man använda Visual Basic, C eller C++. På en Apple-maskin går det ofta bra med AppleScript, och i Unix-miljö kan man välja i princip vad som helst som går att skapa körbara program ifrån, t.ex. C, C++, PERL eller shell-script. Generellt kan CGI-program skrivas på C, C++, Ada, Fortran, Unix shell, Visual Basic och AppleScript.

2.2.3 ASP eller CGI?

CGI är en betydligt äldre teknik än ASP vilket inte innebär att det är en sämre teknik. CGI metoden låter ett program på en webbserver kommunicera med en webbläsare i ren ASCII-kod. Det gör alltså att CGI är inte i sig ett programmeringsspråk. Så länge den överenskomna metoden följs kan CGI-programmet vara skrivet i vilket språk som helst. Det i särklass vanligaste språket för ett CGI-program är Perl.

ASP och CGI är två teknologierna som är nära besläktade. Dessa två metoder har nästan exakt samma användningsområde, med var sina fördelar och begränsningar.

I princip allt som går att åstadkomma med ASP går även att åstadkomma med CGI och tvärtom. Likheter hittar vi också vid exekveringen, då CGI-programmet precis som ASP exekveras på servern och en ny webbsida skapas och skickas tillbaka till webbläsaren, på basis av programexekveringen.

Den enda stora skillnaden mellan dessa två teknologier är att i CGI använder vi ett separat program skrivet i ett vanligt programmeringsspråk, medan i ASP ligger programsatserna inbakade i HTML-koden. Detta gör att ASP är lite lättare att förstå sig på och mer användarvänligt då man inte behöver skapa ett separat program.

2.3 Summering

I detta kapitel har vi presenterat två olika typer av tekniker som finns att tillgå för tillämpning av en dynamisk databas och även redovisat för ASP, vilket är den teknik som vi har med samråd med vår uppdragsgivare bestämt att använda i vårt arbete.

ASP står för Active Server Pages, är en teknologi utvecklad av Microsoft för att kunna baka in programmeringskod, så kallad ASP-kod, i HTML-koden. ASP fungerar på så sätt att ASP-koden skickas först till en programtolk på webbservern som exekverar koden i samma ögonblick som webbsidan anropas. Beroende på resultatet av användarens krav så skriver programtolken en helt ny webbsida i ren HTML-kod utan ASP-kod, och det är just detta HTML-kod som skickas tillbaka till användaren.

Det programmeringsspråk som vi använder är VBScript, men som nämnd ovan kan man skriva koden i valfritt skriptspråk.

VBScript är en förenklad skriptversionen av programmeringsspråket Visual Basic, utvecklad av Microsoft. De flesta av Microsofts program som till exempel Officepaketet förstår VBScript och det är just VBScript som används i bakgrunden då man själv skapar makron i till exempel Microsoft Word och Microsoft Word.

3 Design av databasen

Detta kapitel beskriver hur vi gick tillväga vid skapandet av vår databas och utformningen av ER-modellen som är nödvändig för det framtida arbetet och konstruktion av vårt system. Vi har erhållit en befintlig databas av EuroBossClass AB, innehållande uppgifter om deras kunder. Databasen i sig är alltså inte fullständigt utan behöver vidareutveckling. Den innehåller alltså endast uppgifter om de redan befintliga kunderna som finns registrerade i databasen. Med uppgifter menar vi namn, adress, stad, telefon, E-mail etc. om en specifik kund. Resten av kapitlet innehåller en mer detaljerad beskrivning av de olika entiterna som finns i vår datamodell med tillhörande attribut. Där kommer det att framgå vilka delar av databasen som är skapade för denna projekt och den existerande kundregistret.

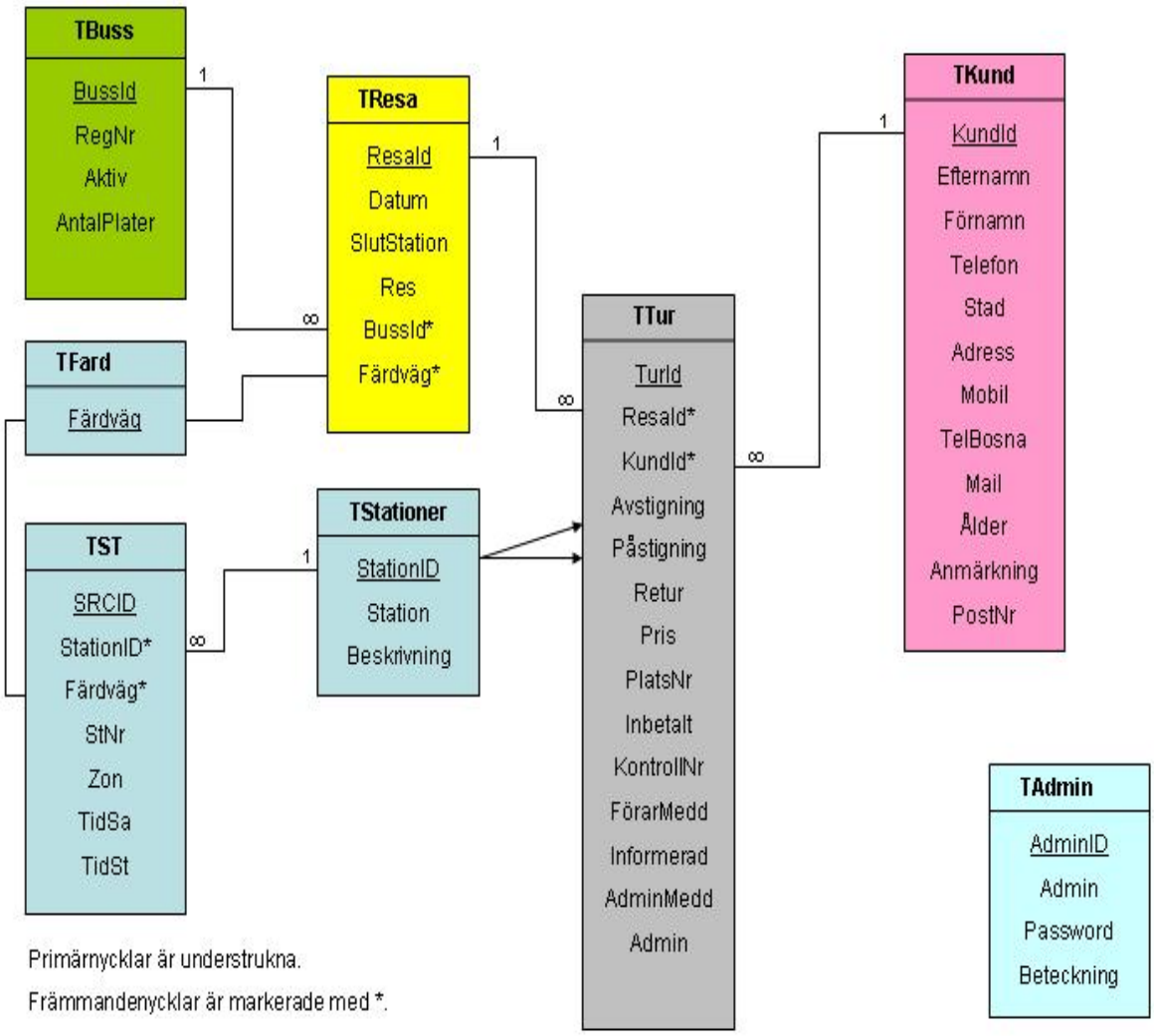
Enligt de kraven som är ställda, måste databasen och systemet var skapad så generell som möjligt för att lätt kunna byggas ut vid senare behov.

3.1 Arbetets gång

Arbetet med databasen gjordes på så sätt att först skapades tabellerna med hjälp av datamodellen. Sedan använde vi oss av den befintliga kundregistret för att testa vår databas. För att kunna kommunicera med databasen behöver man använda SQL, vilket betyder Structured Query Language. Detta språk använder vi bland annat för att tala om för databasen vad det är som skall hämtas, läggas till, tas bort eller ändras.

3.2 ER-diagram (Entitet/relation)

Databasen är skapade på så sätt att alla entiteter med tillhörande attribut är definierade. Alla relationer som är utskrivna är 1:N utom relationen mellan TResa och TFard som har relationen 1:1.



Figur 3.1 ER-diagram

3.3 Förklaringar till ER-modellen

Viktigt att uppmärksamma för bedömning av tabellerna att primärnycklarna är understrukna och främmandenycklarna är markerade med *.

3.3.1 TTur

Tabellen TTur innehåller relevant information om en tur för en viss passagerare. Med en tur menar vi då att en resa består av flera turer och därmed kommer varje tur att få sitt eget unika nummer. En resa i sig identifieras också med ett specifikt nummer.

Tabell	**Beskrivning**
Ttur	**Innehåller info om en viss tur.**
TurID	Den unika numret som en tur erhåller.
ResaId*	Identifierar en resa.
KundId*	Den unika numret som en kund får.
Avstigning	Numret på station där passagerare stiger av.
Påstigning	Numret på station där passagerare stiger på.
Retur	Tur och retur biljett.
Pris	Pris för en resa.
PlatsNr	Sittplatsen i bussen för en passagerare.
Inbetalt	Pengarna som kunden betalat in.
KontrollNr	Euroboss interna kontroll nummer.
FörarMedd	Meddelande till föraren.
Informerad	Föraren informerad om meddelandet.
AdminMedd	Medellande till Administratören. (T.ex. om någon kund måsta ha en speciell plats.)
Admin	Administratörens beteckning.

Tabell 3.1 TTur

3.3.2 TResa

Tabellen TResa innehåller information om en viss resa såsom datum för resan, resans slutstation, antal återstående platser som går att reservera. Vi har även attributet BussId (för närvarande för en buss) ifall företaget expanderar och köper in nya bussar. Färdvägen är också för närvarande bara en. Mer förklaring om den kommer i TFard tabellen.

Tabell	**Beskrivning**
TResa	**Innehåller info om en viss resa.**
ResaId	Identifierar en resa.
Datum	Datum för en viss resa.
SlutStation	Slutstation för en resa.
Res	Antal återsående platser i bussen.
BussId	Numret på bussen som kör. (om flera)
Färdväg*	För närvarande bara en färväg(Bihac). I framtiden ska det gå att skapa flera färdvägar innehållande nya stationer.

Tabell 3.3.2 TResa

3.3.3 TKund

Tidigare konstaterade vi att vi hade erhållit en befintligt databas från vår uppdragsgivare. Denna databas bestod av egentligen tabellen TKund som innehåller all viktig information om en specifik kund. Med en kund menar vi en resenär som har bokat en resa. Vi har tagit med även mobiltelefon, E-mail och Telefon i Bosnien för att lättare kunna ta kontakt men en resenär om några ändringar i resplanen skulle uppstå.

Tabell	**Beskrivning**
Tkund	**Innehåller info om en specifik kund.**
KundId	Den unika numret som en kund erhåller.
Efternamn	Kundens efternamn.
Förnamn	Kundens förnamn.
Telefon	Kundens telefon .
Stad	Staden kunden bor i.

Adress	Kundens adress.
Mobil	Kundens mobiltelefon.
TelBosna	Kundens telefon i Bosnien.
Mail	Kundens email.
Ålder	Kundens ålder.
Anmärkning	Övrig information.
PostNr	Post numret till kunden.

Tabell 3.3 TKund

3.3.4 TBuss

Tabellen innehåller information om en specifik buss. För närvarande innebär det att företaget bara har tillgång till en buss, men vi kommer att vid implementeringen ta hänsyn till att företaget kanske kommer inom kort att inköpa en eller flera bussar. Då kommer de att på ett smidigt sätt kunna lägga till och registrera de nya bussarna.

Tabell	**Beskrivning**
TBuss	**Innehåller info om buss(för tillfället bara en buss)**
BussId	Identifierar en buss
RegNr	Registernumret på en buss
Aktiv	Huruvida om bussen är bokad att köra el. ej.
AntalPlatser	Totala antalet sittplatser på bussen.

Tabell 3.4 TBuss

3.3.5 TStationer

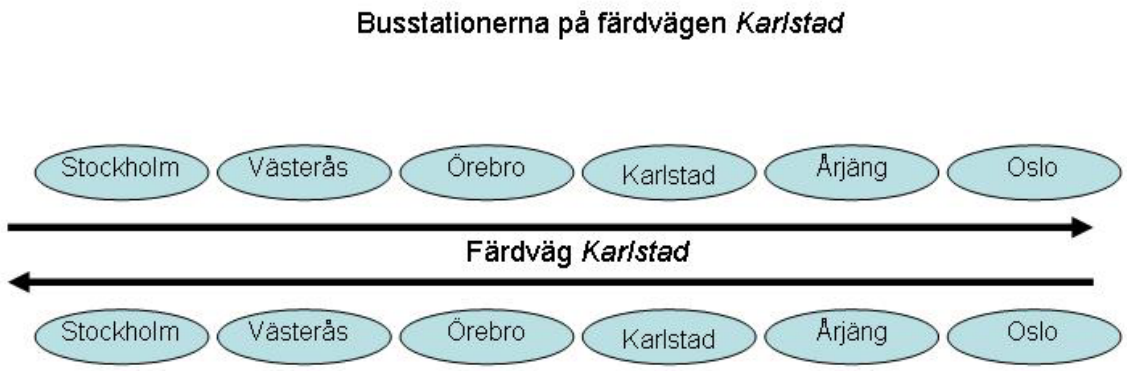
Tabellen TStationer innehåller information om de olika busstationerna. Till exempel så har Stockholm StationID = 1, Södertälje StationID = 2 osv...
Attributet *Station* är namnet på stationen t.ex. Stockholm, Södertälje.
Beskrivning är beskrivningen om var stationen finns. T.ex. Jönköping A6 IKEA.

Tabell	Beskrivning
TStationer	**Innehåller info om stationerna.**
StationID	Identifierings nummer för en station.
Station	Namn på stationen. Ex.Stockholm
Beskrivning	Beskrivning av stationen. Ex. IKEA vid Hydro-pumpen.

Tabell 3.5 TStationer

3.3.6 TFard

Med tabellen Tfard är det tänkt att man på ett enkelt sätt ska kunna skapa en ny färdväg och utifrån den nya färdvägen ska nya stationer skapas. Just nu kallar vi färdvägen för Bihac eftersom det är den staden i Bosnien bussen åker igenom. I framtiden ska man även kunna skapa nya färdvägar och utifrån dem helt nya busstationer. Som ett exempel kan vi ta att företaget vill arrangera resor till Oslo. Då kommer vi att kalla den nya färdvägen för (ex. Karlstad) och för färdvägen Karlstad kommer nya busstationer att skapas (Stockholm-Västerås-Örebro-Karlstad-Årjäng-Oslo).



Figur 3.2 Färdväg ”Karlstad”

Tabell	Beskrivning
TFard	**Innehåller info om färdvägar.**
Färdväg	Identifierar en ny färdväg för en resa.

Tabell 3.6 TFard

3.3.7 TST

Med hjälp av TST tabellen skall man kunna efter att man har skapat en ny färdväg och ordna de nya stationerna i en speciell ordning. Man ska alltså kunna lista upp de olika busstationerna man tänker besöka vid en ny resa och dessutom veta ankomsttiden vid nästa station. Det ska även gå att boka av (alltså inte besöka) en ”tidigare skapad” station om man inte vill besöka denna. (Se dock avsnittet Avgränsning 1.3.5)

Tabell	**Beskrivning**
TST	**Innehåller info om en viss resa.**
SRCID	Identifierare
StationId*	Namn på stationen. Ex.Stockholm
Färdväg*	Identifierar en färdväg för en resa.
StNr	Ordningsnumret på en station.
Zon	Identifierar en zon.
TidSa	Avgångstid från en station då bussen ska till Sarajevo.
TidSt	Avgångstid från en station då bussen ska till Stockholm.

Tabell 3.7 TST

3.3.8 TAdmin

Med sidans administratörer menar vi behörig personal inom EuroBoss. Tabellen TAdmin används för att endast behöriga ska ha tillgång till systemet genom att skriva ett specifikt inloggningsnamn och lösenord. Om inloggningsnamnet och lösenordet inte stämmer visas ett felmeddelande och man ges möjlighet att logga in på nytt.

Tabell	**Beskrivning**
TAdmin	**Innehåller info om sidans administratörer.**
AdminID	Identifieringsnummer för en administratör.
Admin	Administratörens namn eller nick.
Password	Administratörens password.
Beteckning	Beteckning.

Tabell 3.8 TAdmin

3.4 Summering

I detta kapitel har vi beskrivit hur vi har gått tillväga vid skapandet av vår databas och utformningen av ER-modellen som är nödvändig som är av stor vikt för arbetet och konstruktion av vårt system. Vi hade från EuroBossClass erhållit en befintligt databas innehållande information om deras redan registrerade kunder. Med information menar vi namn, adress, tel, stad, E-mail etc. om en specifik kund. Databasen i sig var långt ifrån komplett och behövdes att utvecklas mycket mera.

Det fösta steget i arbetet var att få en klar bild över hur databasen skulle se ut och vilka tabeller, entiteter med tillhörande attribut som behövdes för att kunna klara av de krav som ställdes på våran databas. Genom att göra detta kunde vi skapa en ER-modell som gav oss en översiktligt bild över de diverse attribut och entiteter som behövdes för arbetet. Det andra steget i utvecklingsarbetet var nu att bestämma primärnycklar och främmandenycklarna i denna ER-modell. Det är mycket viktigt att man från början får fram de attribut som är unika i varje tabellen för att sedan kunna bestämma relationerna mellan tabellerna och koppla ihop tabellerna på ett så smidigt och smart sätt som möjligt.

Nu hade vi äntligen en databas som motsvarade de förväntningar som vi och vår uppdragsgivare hade från början. Viktigt att nämna är att vi hade under hela konstruktionstiden kontakt med EuroBossClass och de var delaktiga i skapandet av databasen.

4 Användargränssnitt och Implementation

I detta kapitel beskrivs utvecklingsarbetet för utformning av vårt systems gränssnitt samt Implementation. Vi hänvisar till bilagan **A Filöversikt,** där det framgår klart och tydligt hur filerna exekveras när programmet körs vid de olika menyvalen.

4.1 Funderingar kring Användargränsnittet

De generella kraven som vår uppdragsgivare ställde i kravspecifikationen är att systemet ska vara lätttillgängligt och gränssnittet ska vara tydligt. Alla fönster ska ha beskrivande text som hjälper användaren vid navigering av programmet. Användargränssnittet ska vara grafiskt med ett huvudfönster och underfönster som öppnas i samma fönster(frames). Användaren ska kunna på ett smidigt sätt navigera sig genom de olika fönstren genom att klicka på knapparna. De funderingar som vi hade innan vi bestämde oss för hur vårt gränssnitt skulle se ut grundar sig på efterforskning och jämförande med andra typer av reseföretag som SWEBUS och SJ, där användarnas förväntningar av sådana system undersöktes. En tanke som är av stor vikt för design av gränssnittet är dess användare. I vårt fall så riktar vårt system sig åt de anställda på EuroBossClass. Därmed måste man ta hänsyn till att dessa användare har en annorlunda inställning och krav till systemet än privatpersoner som t.ex. vill boka biljetter själva på SJ eller SWEBUS. De generella aspekterna som användarvänlighet och tillgänglighet kvarstår dock.

Vi har haft funderingar kring en framtida utveckling av ett kundsystem, där kunden kan själv boka sin egen biljett precis som på SJ:s webbsida. Men det finns för närvaranade ingen intresse från EuroBoss att ta fram ett sådant kundsystem.

När det gäller utseende på vårt gränssnitt har vi försökt att skapa ett homogent system med samma färg, utseende och ramindelning. Detta för att skapa en trygg miljö för användaren och undvika missförstånd. De texter som förekommer i gränsnittet har samma teckensnitt och teckensnittstorlek. Detta av samma principer som är nämnda ovan.

4.2 Uppkoppling mot databasen

Det första som man lägger märke till i koden är de följande två rader av kod som vi använder i varenda fil som kommunicerar med databasen.

```
Set cn = Server.CreateObject("ADODB.Connection")
cn.Open "DSN=d4584-euro;Password=@01#$dcacigolbnhpladklcdc"
```

Här startar vi objektet som heter ADODB.Connection. Objektet vi startar här är ett ActiveX Data Object. Vi placerar sedan objektet i en variabel som vanligt.

På nästa rad anropar vi objektet via variabeln, tillsammans med en av dess egenskaper, nämligen Open, som öppnar databasen.

För att stänga en databaskoppling använder vi ActiveX-objektets variabel igen, men den här gången med egenskapen Close.

cn.Close.

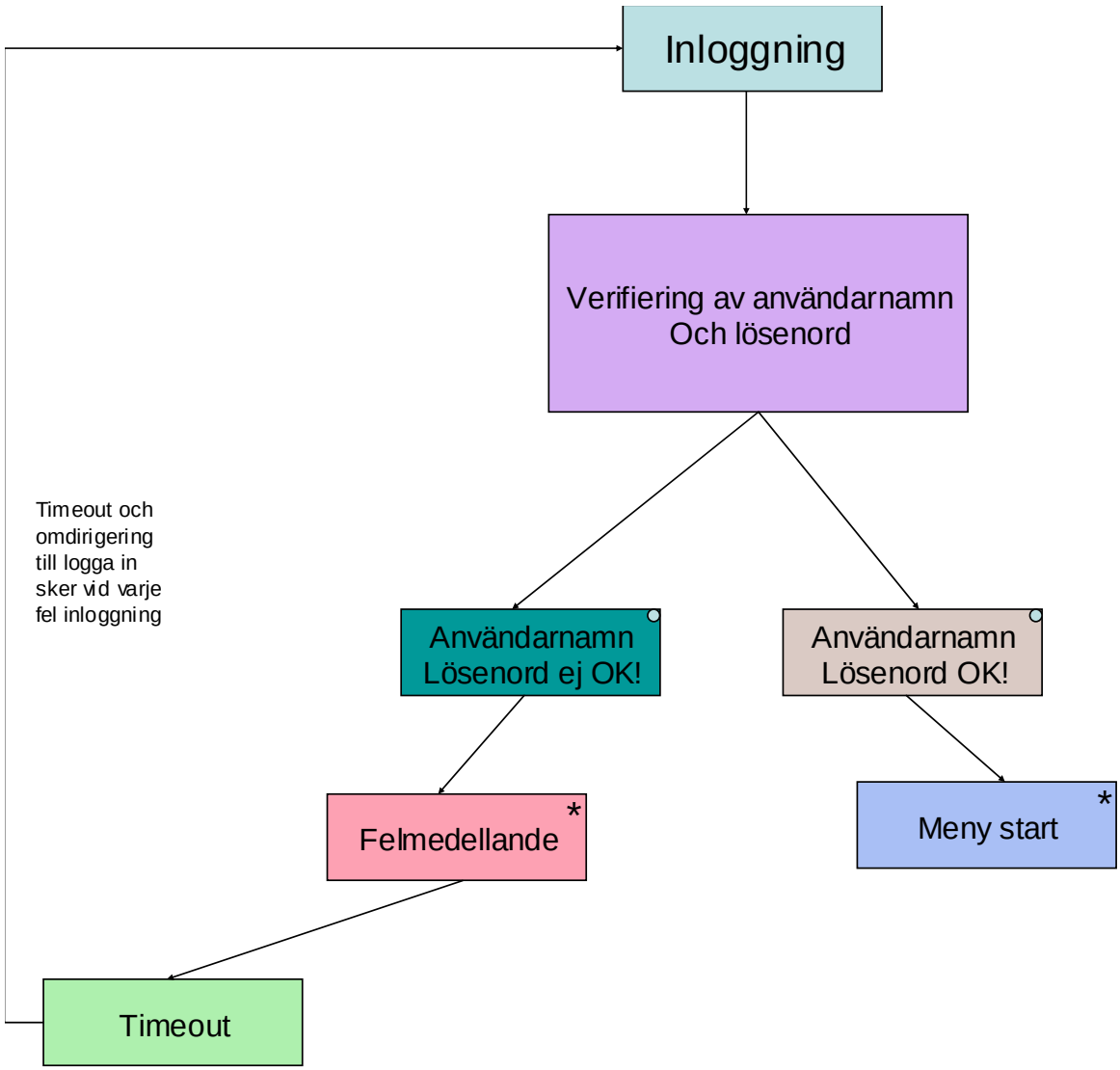
Från början visste vi inte att vi var tvungna att skapa en connection ovan för varje gång vi kommunicerar med databasen. Efter att vi insåg detta så skapade vi en subrutin connect() i filen funktioner.asp.

```
sub connect(cn)  'Hanterar uppkoppling
dim db
db=Server.MapPath("./euro.mdb")
Set cn = Server.CreateObject("ADODB.Connection")
cn.Open "driver={Microsoft Access Driver (*.mdb)};dbq=" & db
end sub
```

Med hjälp av denna subrutin så kan databasen öppnas även om vi lägger upp det på en webbhotell då den är oberoende av DSN (Data Source Name) namnet och lösenordet. Man anropar subrutinen genom att skriva ”connect cn” istället för att skriva de två långa raderna.

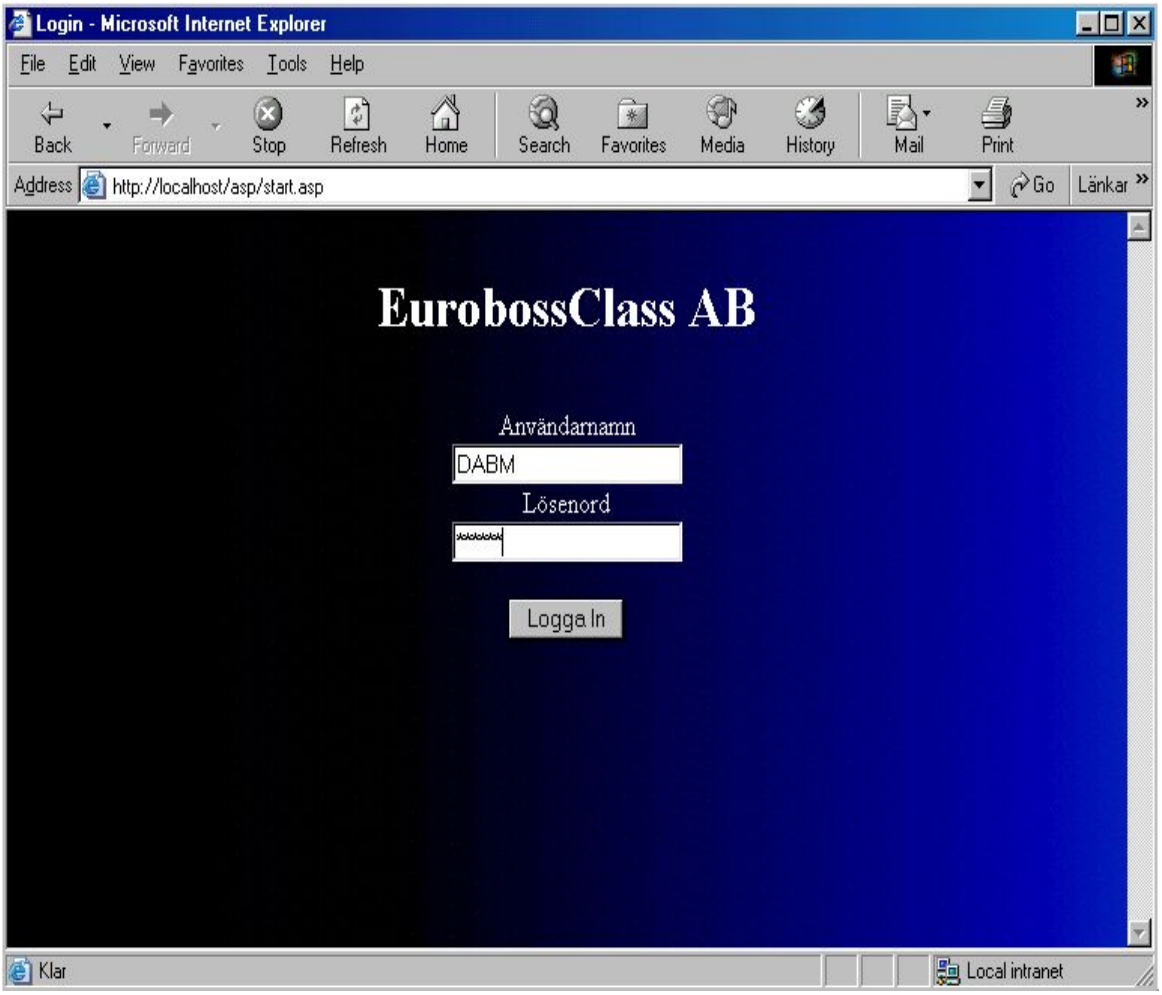
4.3 Inloggning

Vi inleder beskrivningen av vår *Inloggningsfunktion* genom att en översiktligt bild på hur funktionen exekveras vid Inloggning. Vidare måste vi klarlägga att vår uppdragsgivare EuroBossClass har begärt från oss att ej lämna ut koden till vår *Inloggning* funktion av säkerhets skäl.



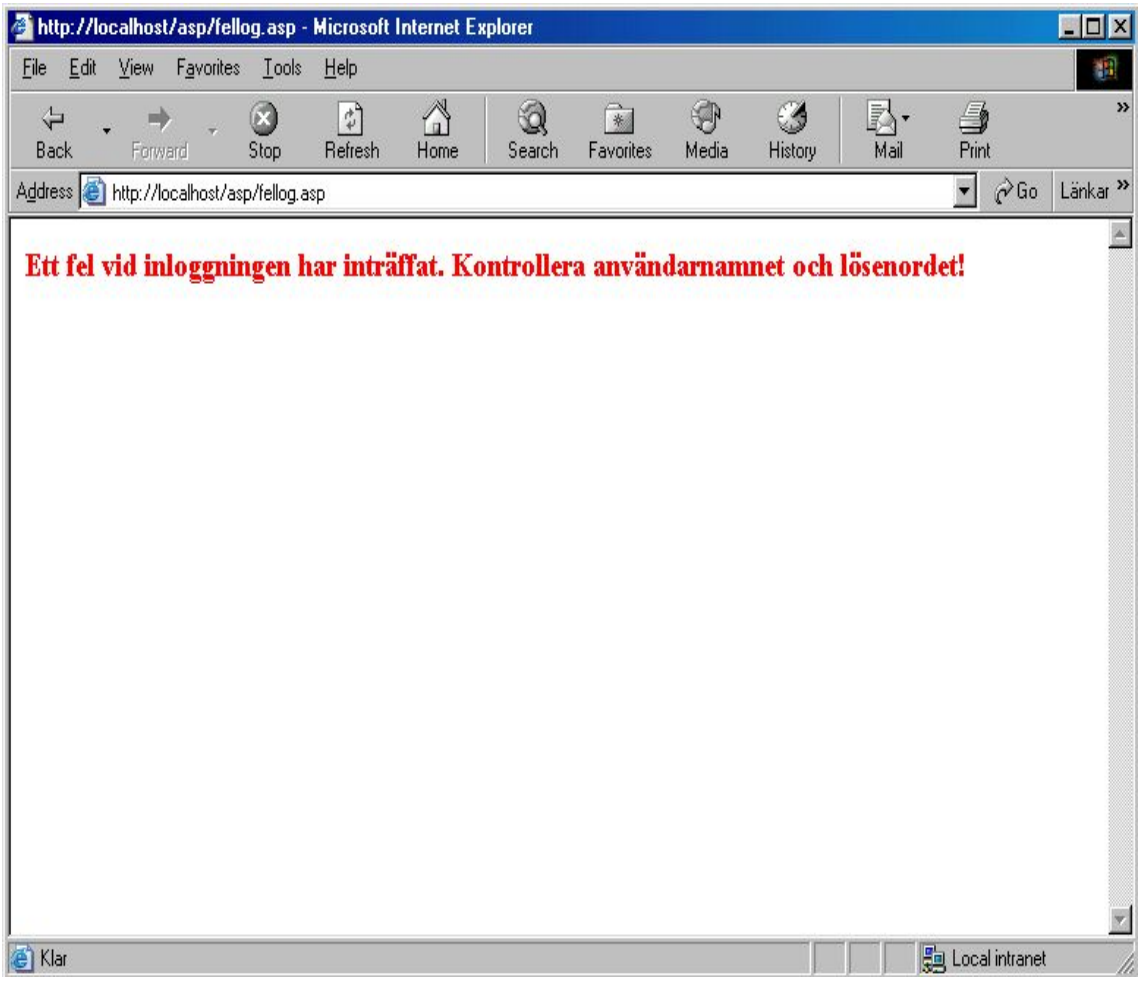
Figur 4.1 Inloggning.

Som vi tidigare har nämnt så ska bokningssystemet fungera som en webbsida, som endast företagets personal och andra behöriga har tillgång till. Detta har vi löst med hjälp av en inloggningsfunktion. Användaren uppmanas att mata in ett lösenord och ett användarnamn. Uppgifter om behöriga användare och deras lösenord har vi placerat i tabellen TAdmin.
För denna inloggningsfunktion har vi använt oss av Cookies då det i framtiden är tänkt att varje användare ska få sin egen profil (cookie) sparad på sin hårddisk. Med profil menas det att varje anställd inom företaget kommer att tilldelas egna privilegier, då webbsidan kommer att ändra utseende både visuellt och innehållsmässigt.



Figur 4.2 Logg in

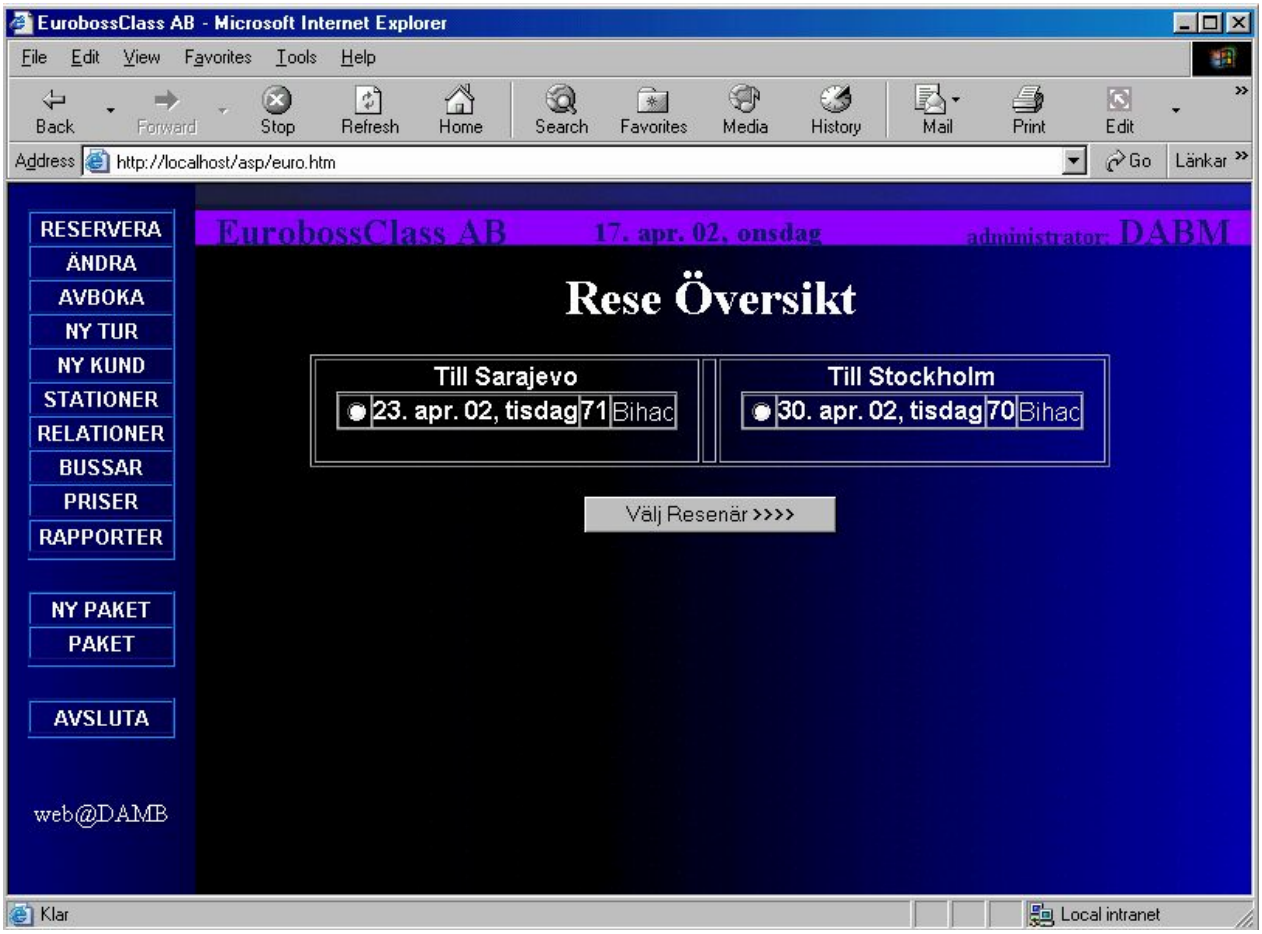
Om vi matar in fel användarnamn och lösenord kombination blir vi omdirigerade till filen *fellog.asp* som visar ett felmeddelande (figur 4.1 Felinmatning). Därefter anropas filen *start.asp* och användaren har en möjlighet att logga in på nytt.



Figur 4.3 Felinmatning

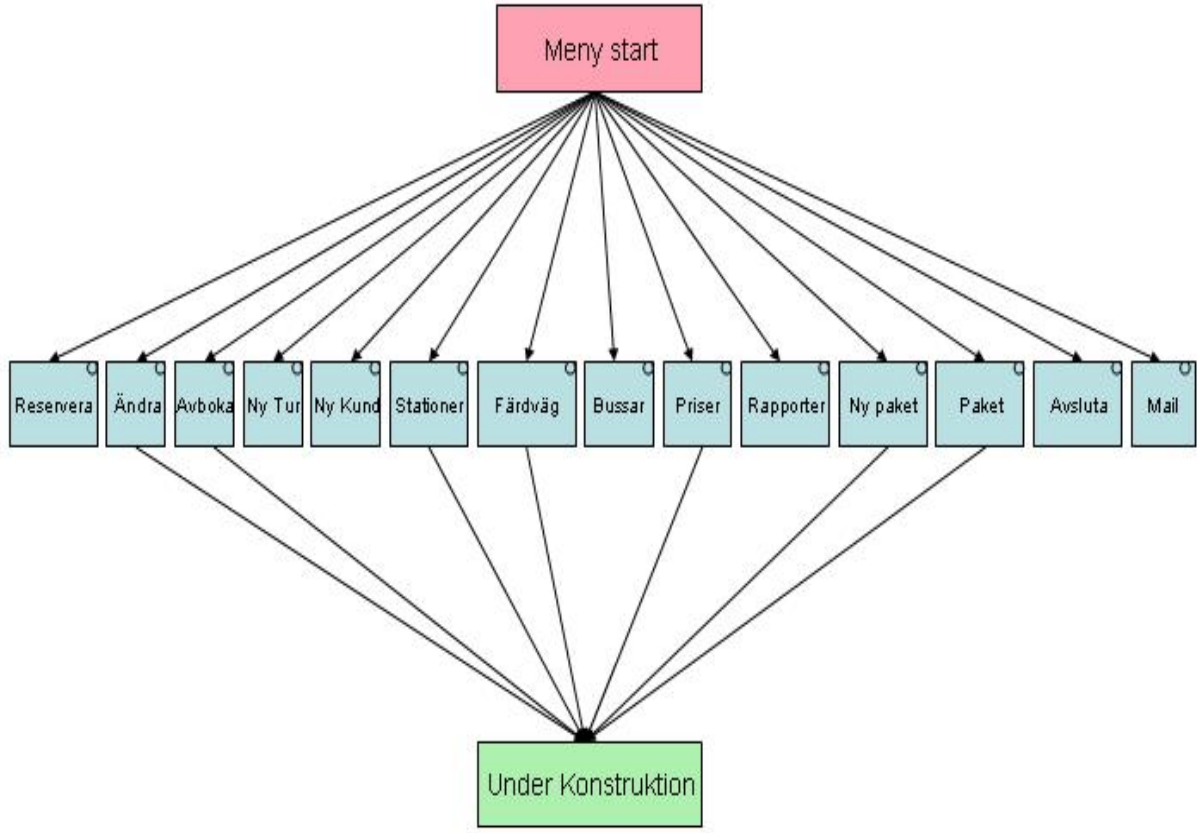
4.4 Huvudmeny

Om vi därmed matar in rätt användarnamn och lösenordskombination blir vi omdirigerade till filen *euro.htm*. *Euro.htm* som använder sig av tre frames. Vi har en frame längst till vänster (menyn), en längst upp på webbsidan och en huvud frame som visar de olika sidorna beroende på vilket val i menyn vi gör.



Figur 4.4 Huvudmeny

Figuren nedan är en JSP diagram över de olika meny knappar som finns att tillgå i vänstra frame (menyn).



Figur 4.5 JSP diagram över meny valen

Filen *top.asp* har hand om den frame som ligger längst upp på webbsidan (Banner frame). Den returnerar dagens datum, dag (ex. 17.apr.02, onsdag) och administratörens beteckning.(ex:DAMB-Danial Abdali, Bernes Maksumic)

Figur 4.6 Frame-indelning

Filen *meny2.asp* sköter meny-frame, alltså den längst till vänster. Här har vi även gjort så att de olika användarna inte behöver se alla menyval. Till exempel så är bara chefen eller revisor intresserad av de finansiella aspekterna. När de loggar in så skall ytterligare ett menyval dyka upp (i detta fall Finans) som ingen annan behörig personal har tillgång till.

På så vis kan vi skapa en prioritering bland personalen, där alla anställda inom företaget har sin egen personliga profil. Det här är dock bara i experimentfasen och kommer att vidareutvecklas. Som ett litet exempel så är chefen och revisorn tilldelad ytterligare ett menyval som heter FINANS då de loggar in i bokningssystemet.

Vi löst detta med hjälp av följande kod:

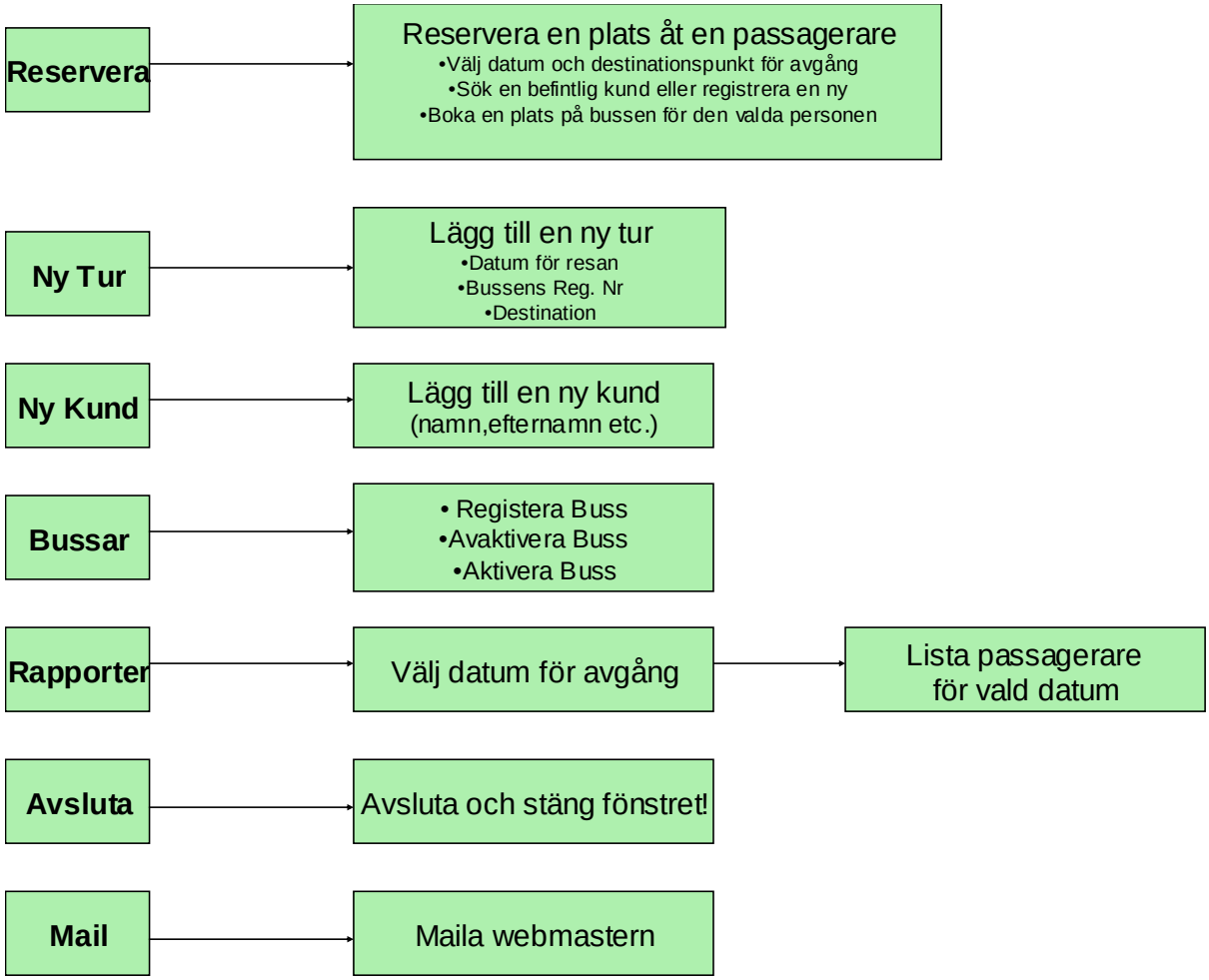
```
<% if admin="Chefen" or admin="Revisor" then %>
<td valign="top" align="center" onmouseover="this.bgColor =
'DodgerBlue';" onmouseout="this.bgColor = 'Transparent';">
<A class=bijel href="finanser.asp" target=rbottom>
```

```
<font face="Arial" size="2"><b>FINANS</b></font></a>
</td>
</tr><% end if %>
```

Filen *meny.asp* är den fil som startas i huvudframe då vi loggar in i systemet. Det är just samma fil man anropar först då man väljer menyknappen RESERVERA. Mer om detta då vi tar upp när vi beskriver de olika menyvalen mer ingående.

4.4.1 Diverse meny val

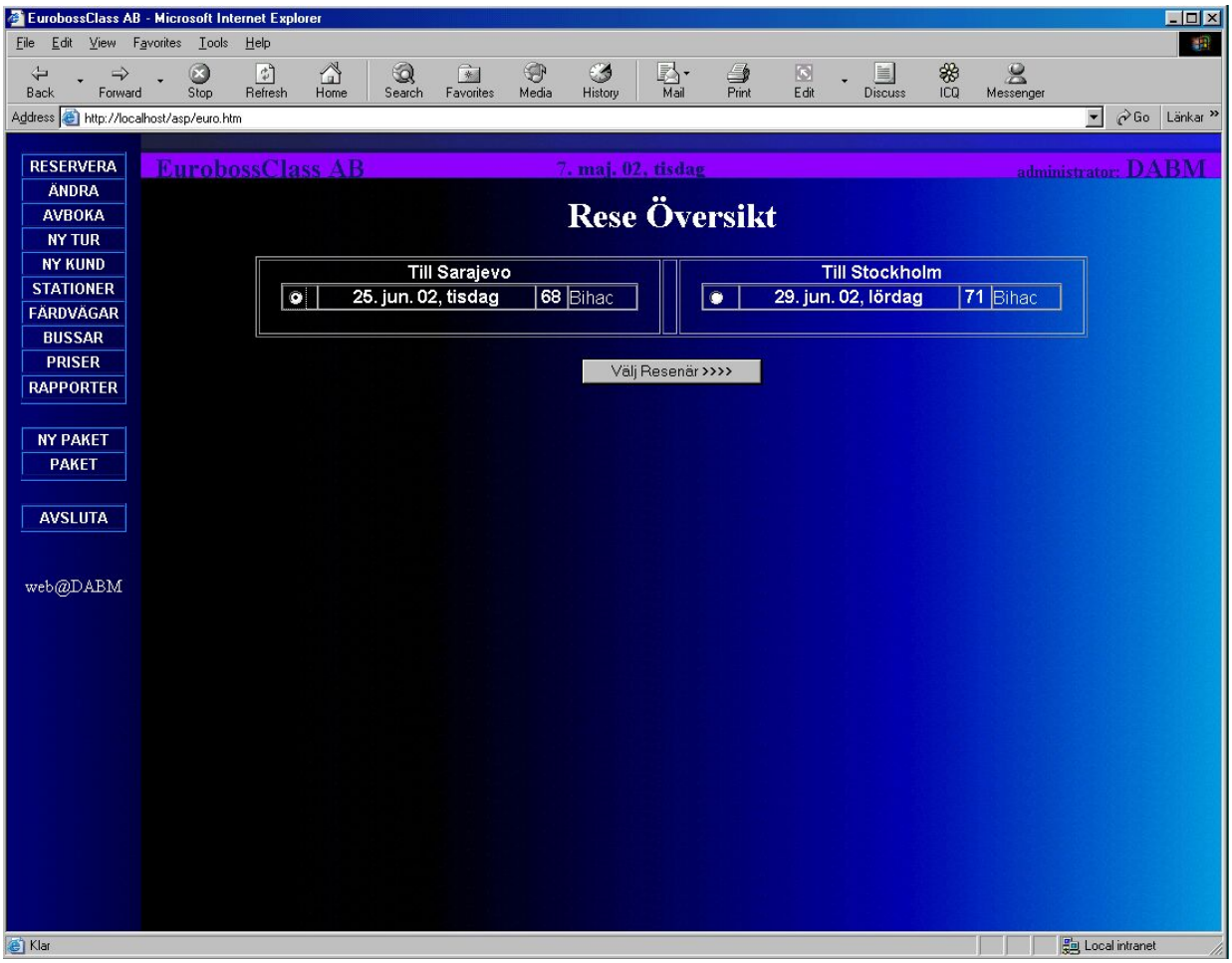
Nedan följer en figur som har till funktion att ge en övergripande bild om de olika funktionaliteterna bakom varje meny knapp som visas på huvudmenyn för inloggade individen. Mer detaljerad beskrivning av varje undermeny kommer att finnas längre fram i rapporten.



Figur 4.7 Diverse meny val

4.5 Reservera

Under menyn *Reservera* finns allt som behövs för att kunna boka en plats för en specifik kund. Man börjar först med att välja ett avgångsdatum för resan.

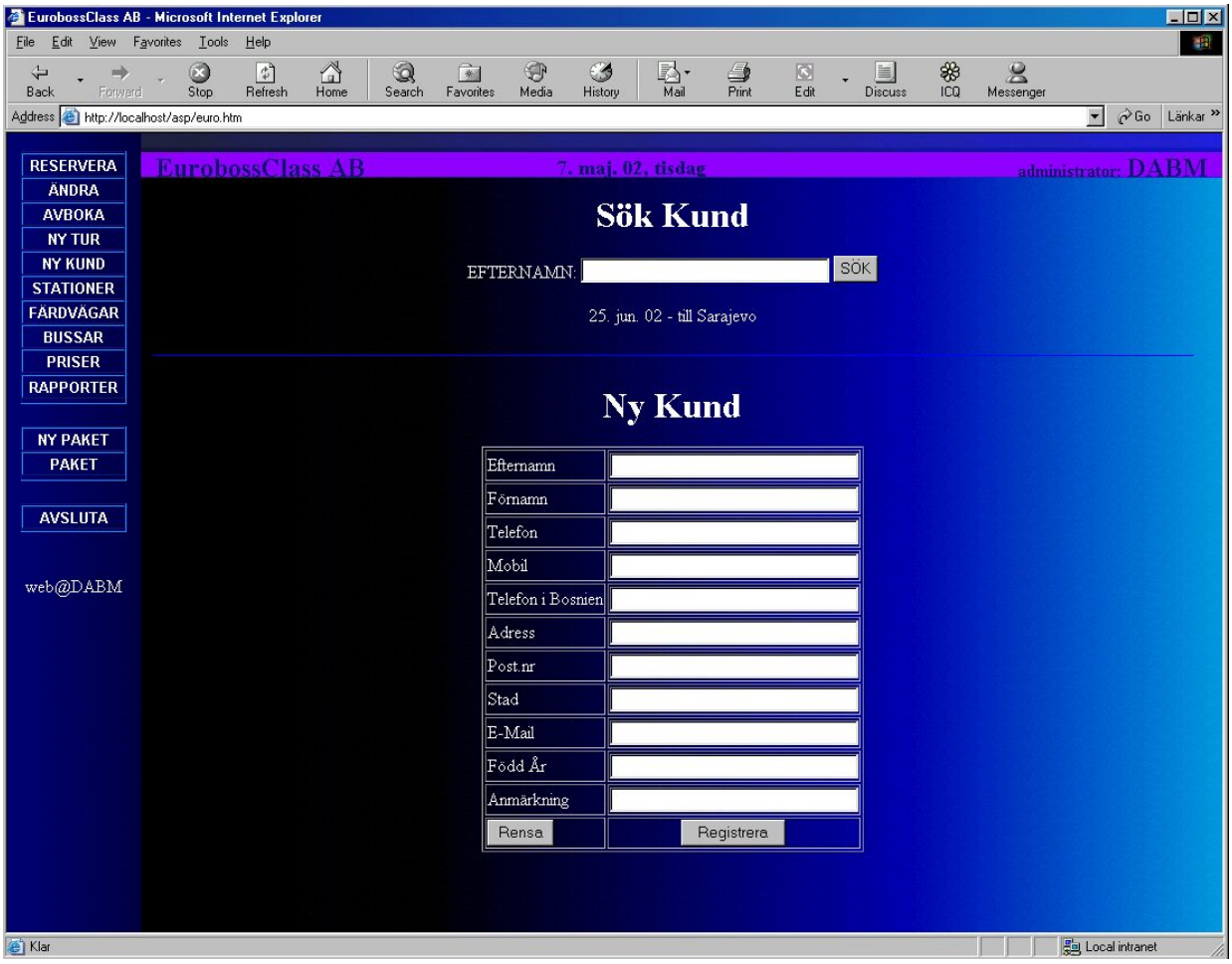


Figur 4.8 Val av datum för avgång

Det är filen meny.asp som har hand om avgångsdatumet. Vi har gjort så att vi har delat upp de olika avgångarna i två tabeller för att de ska bli lättare att överblicka. Under texten *Till Sarajevo* ser man alla registrerade avgångar till Sarajevo och under texten *Till Stockholm* ser man alla registrerade avgångar till Stockholm. För varje avgång visar vi vidare en ”radio button” (m.h.a. denna vet vi vilken avgång som är vald), innehållande uppgifter som datum för avgång, antalet lediga platser i bussen och färdvägen. Information om detta hämtar vi från tabellen **TResa**. Vi har även gjort så att gamla avgångar inte visas på webbsidan då de inte är aktuella längre(dvs. efter avgångsdatumet). Efter att datum för avgång är vald navigerar man sig fram till *Välj Resenär* >>>> knappen: Här kommer man till filen kund1.asp.

Det första vi gör här är att kolla om ett avresedatum är valt. Om så inte är fallet får användaren ett felmeddelande som uppmanar honom/henne att göra just detta (Datum för resan är inte vald!). Vi kontrollerar även vilken av radio knapparna som är markerad. Då bägge två är markerade innebär det en tur och retur resa.

Filen kund1.asp har hand om att söka efter en tidigare registrerad kund i databasen, men vi har även lagt till en funktion där man kan registrera kunden för första gången. Denna funktion är densamma som vi använder oss av i *Ny Kund* menyn.



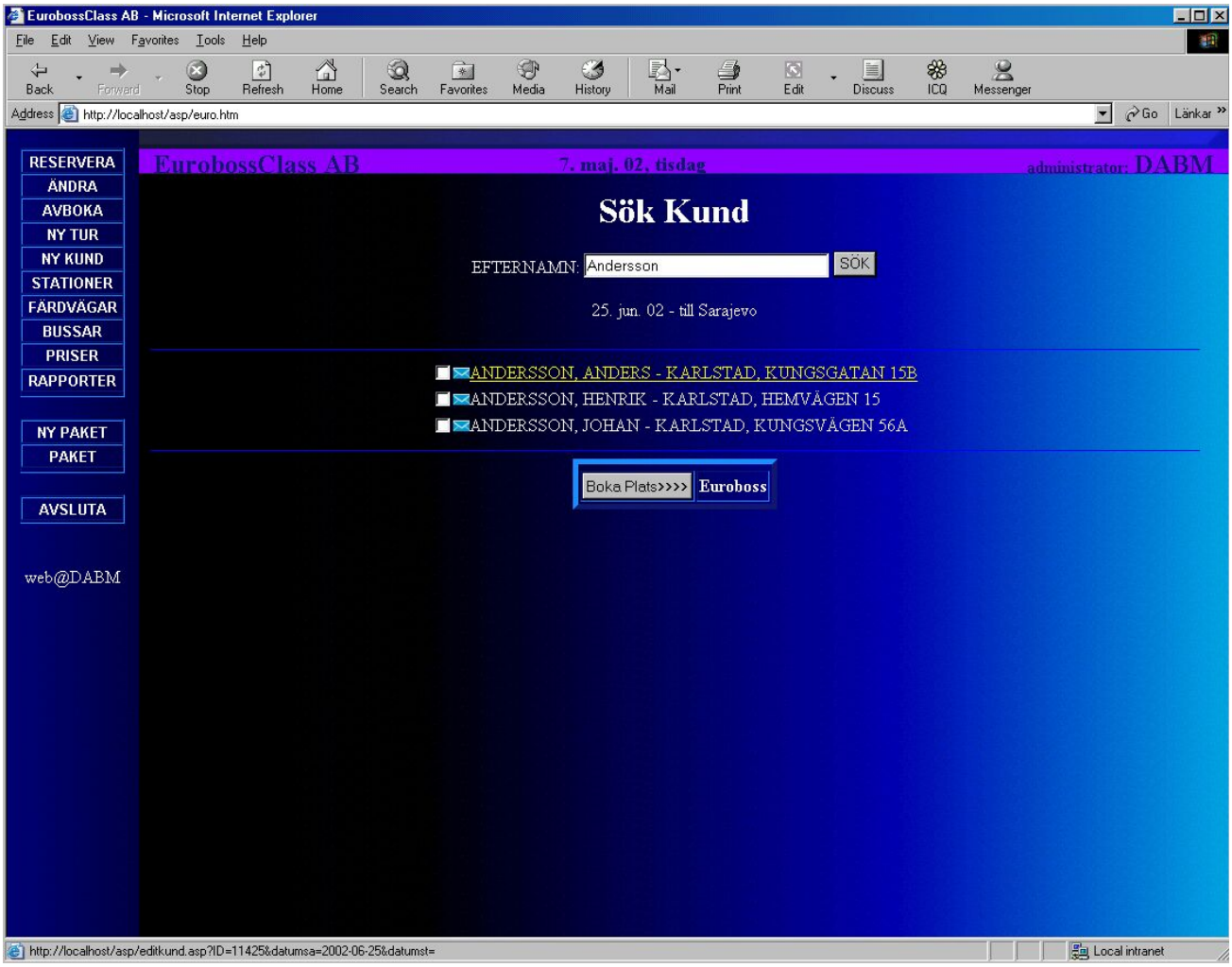
Figur 4.9 Sök eller lägg till kund

Vi söker kunden efter dennes efternamn, eftersom det krävs speciellt tillstånd att lagra information om personnummer. Under sökfunktionen syns det valda avgångsdatumet för kunden (25. jun. 02 - till Sarajevo). Här visas även om resan är en tur och retur resa med sina respektive avgångsdatum.

Om kunden sedan tidigare inte finns i databasen så måste han/hon registreras först. Därefter söker man efter kunden i databasen för att säkerställa att alla inmatade uppgifter är registrerade.

Söker vi efter till exempel efternamnet Anderson så kommer alla registrerade kunder med efternamnet Andersson att visas i bokstavsordning. En fördel med sökning med efternamn är att många familjer reser ihop och då kan man enkelt boka biljetter åt hela familjen genom att markera i check boxen de resande med samma efternamn som tillhör samma familj.

För varje hittad kund visar vi en check box och om kunden har en e-mail adress visas även ✉ - symbolen vid kundens namn. Vidare visas kundens hemstad och adress.



Figur 4.10 Sök och välj kund

Det går även att klicka på varje hittad kund för att ändra eventuella uppgifter om kunden.

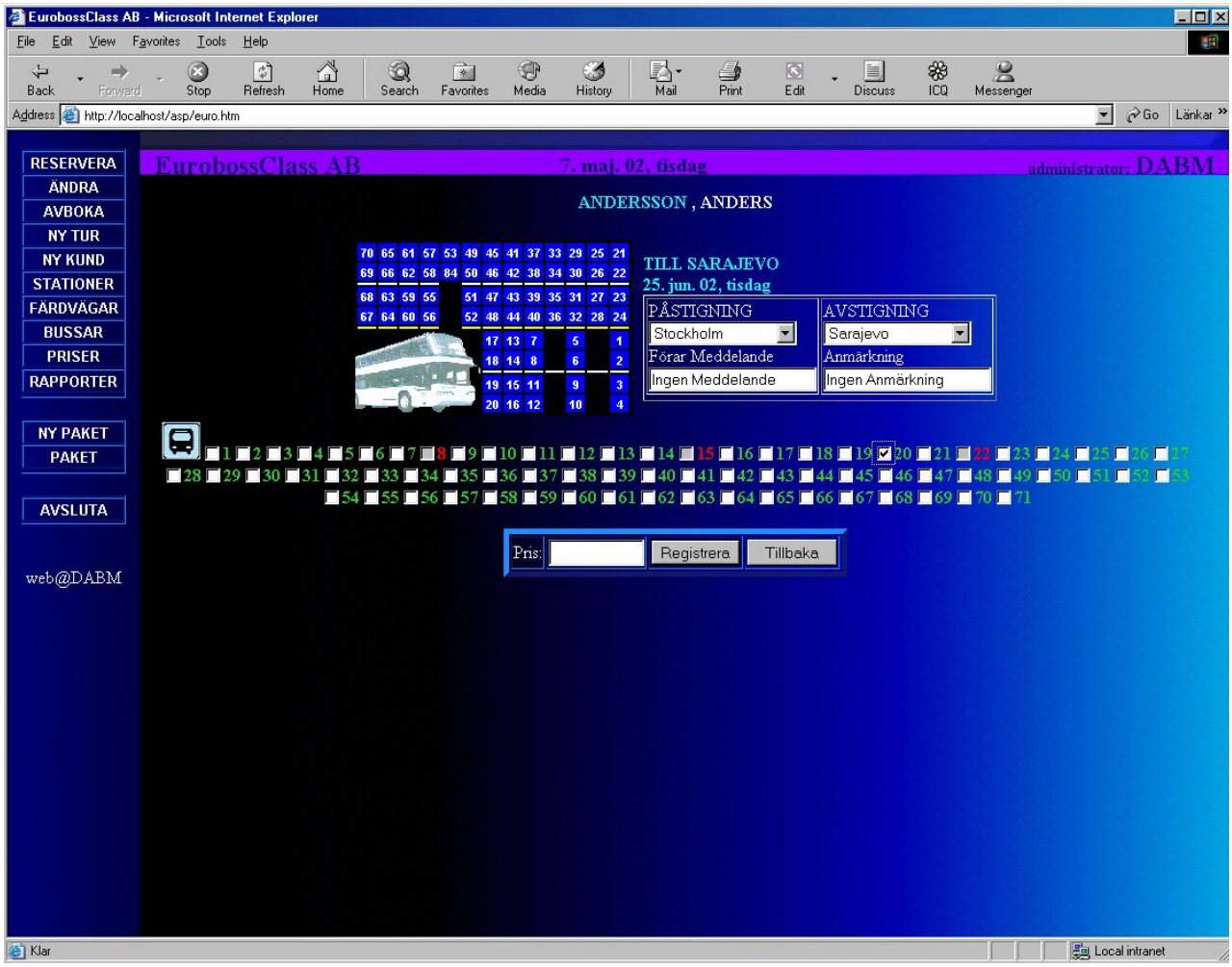
Ex.

☐ ✉ ANDERSSON, ANDERS – KARLSTAD, KUNGSGATAN 15B

Om vi klickar på raden ovan så går vi till filen editkund.asp. Här returneras uppgifterna som är inmatade i databasen till vanliga inputboxar där vi kan på ett enkelt sätt editera dem. Filen kundupdate.asp uppdaterar databasen med ändringarna gjorda i inputboxarna.

Efter att vi har sökt efter kunden med efternamnet Andersson och fått fram alla Andersson i databasen är det dags att välja kunderna. Man kan välja en eller flera kunder genom att kryssa för checkboxen för varje kund. Vi antar att Anders Andersson är den valda kunden.

Efter att ha navigerat oss fram till *Boka Plats* >>>> knappen kommer vi fram till följande webbsida. Det är filerna reservation1.asp och addtur.asp som sköter denna webbsida.



Figur 4.11 Boka plats för Andersson Anders

Som det framgår av bilden så är alla bokade platser markerade med röd färg och är otillgängliga för markering. Längst upp på sidan ser vi kundens namn, under detta ser vi slutdestination för hans resa och datum. Vidare så väljer vi från en listbox påstignings- och

avstigningsstation för kunden. Eventuella meddelanden till föraren och anmärkningar görs i de två inputboxar placerade under listboxarna.

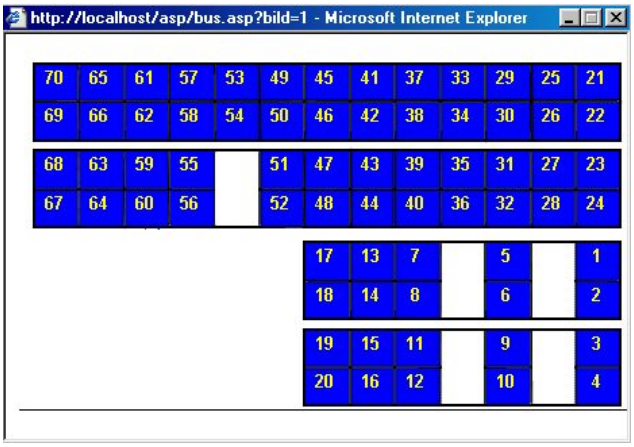
Här görs även en koll om att minst en passagerare är markerad och om flera passagerare är markerade så måste antalet bokade platser vara lika med antalet valda kunder. Om detta inte sker får användaren följande felmeddelanden.

Ingen kund är markerad!

respektive

Antalet platser är olika från antalet resande!

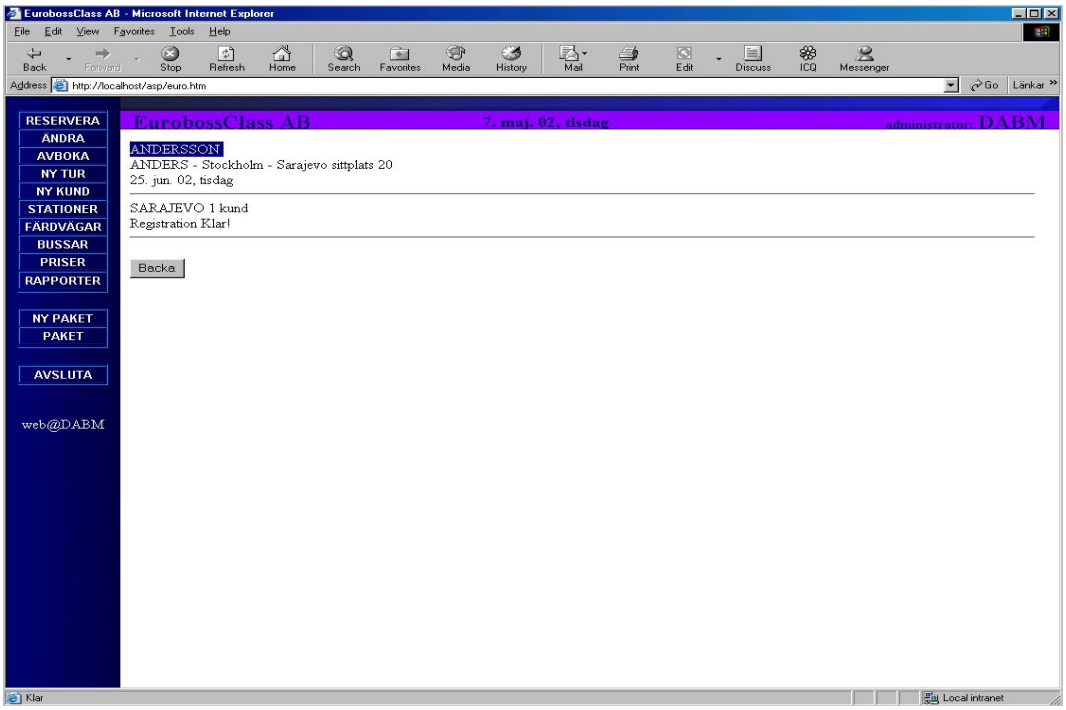
Om användaren klickar på följande bild  så visas en översiktsbild över bussen där det framgår hur platserna är strukturerade i bussen. Detta gör vi mest för att vissa kunder kan ha speciella önskningar om var i bussen de vill sitta. Det är filen bus.asp som sköter detta.



Figur 4.12 Översikt på platserna i bussen

Nu bokar vi platsen nr. 20 för Anders Andersson och navigerar oss fram till *Registrera* knappen. När detta är gjort får användaren en bekräftelse om registreringen.

Registreringen görs i tabellen **TTur** där vi lägger till kundens kundnr, på- och avstigningsstationsnummer, sittplatsnummret, förarmeddelandet och anmärkning. Vi lägger även till administratörens beteckning (DABM i detta fall) och om resan är en tur och retur resa markeras attributen *Retur* i tabellen **TTur** med en fylld checkbox.



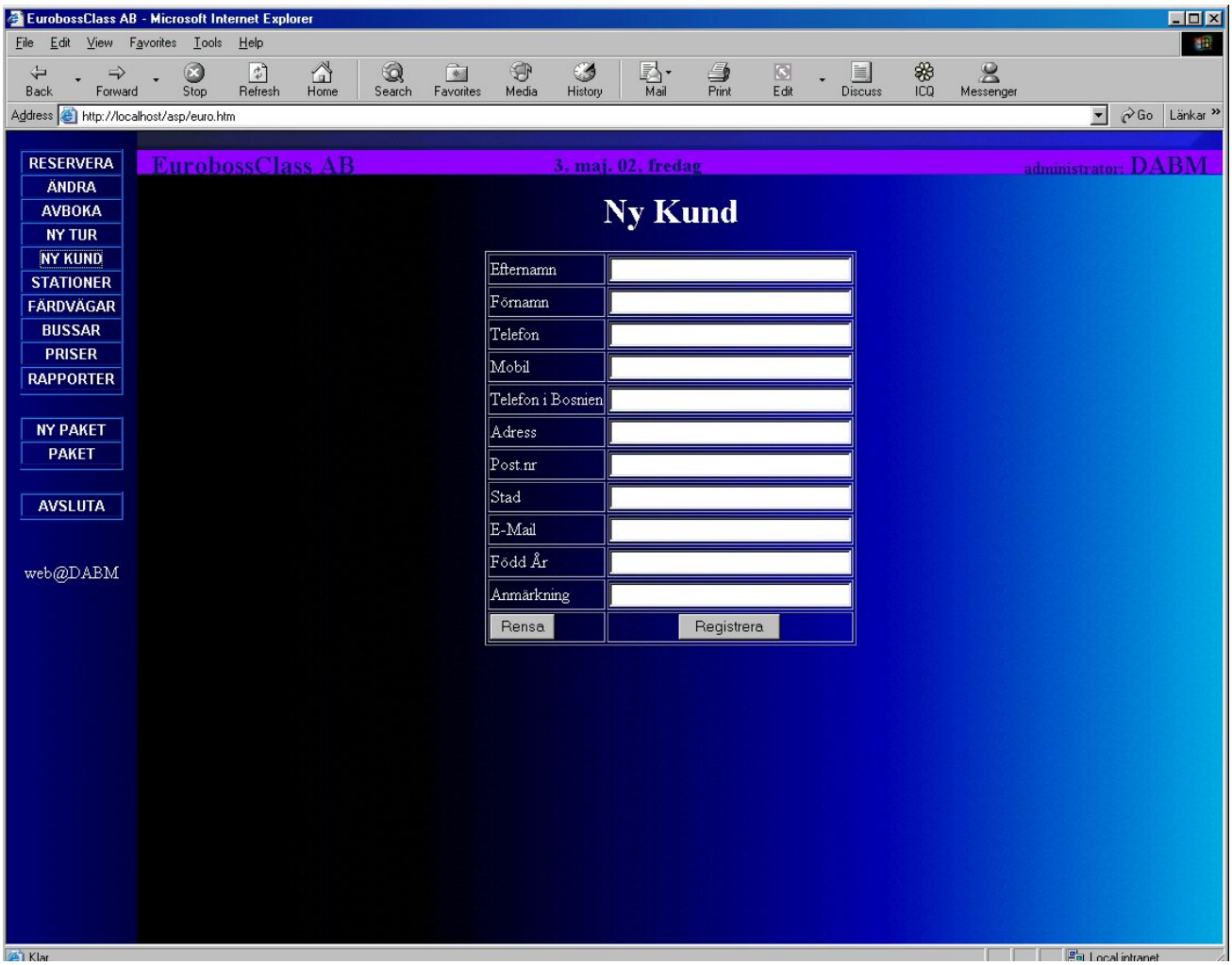
Figur 4.13 Bekräftelse

4.6 Ny Kund

Filen kund2.asp bygger upp själva utseendet på valet *Ny Kund* under menyn. Detta gör vi i subrutinen nykund(). Alla inputbox ligger centrerade och med ett litet mellanrum så det blir lättare och inte så rörigt för ögat att överblicka sidan.

```
Response.write("<td>Efternamn</td><td><input size='30' name='efternamn' ></td><tr>")
Response.write("<td>Förnamn</td><td><input size='30' name='fornamn' ></td><tr>")
Response.write("<td>Telefon</td><td><input size='30' name='telefon' ></td><tr>")
```

Med raderna ovan skapar vi inputboxar som vi ger namnen efternamn, fornamn, telefon o.s.v. för varje imputbox. Se figuren 4.14



Figur 4.14 Ny Kund

Vidare skall användaren mata in Mobiltelefonnummer, Telefon i Bosnien, Adress, Postnummer, Staden de bor i, e-mail adressen, Födelsedatum (kommer att användas senare i projektet för prisräkning) och speciell anmärkning om kunden (t.ex. om kunden inte vill sitta i motsatt riktning än färdriktningen).

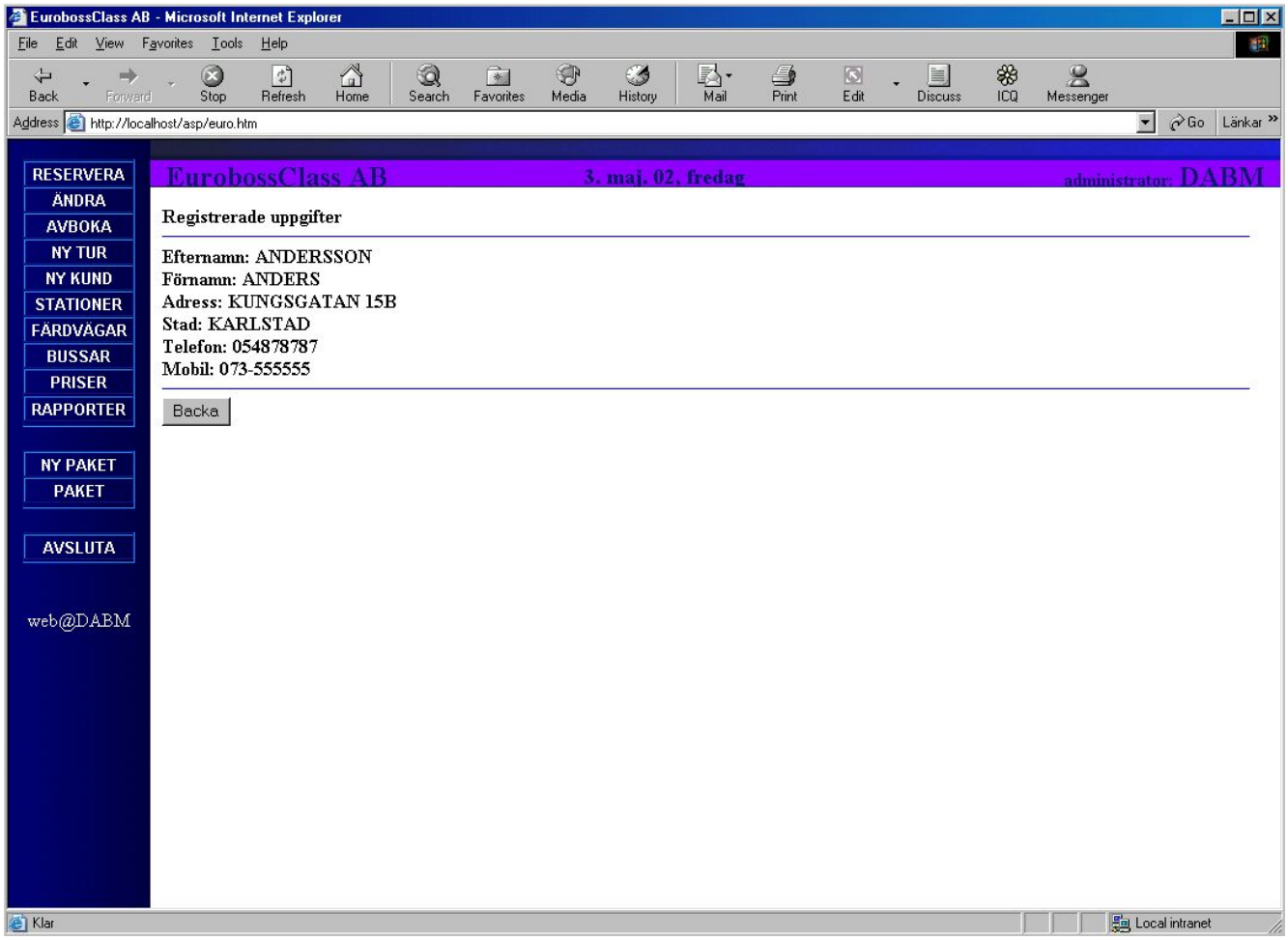
Längst ner på *Ny Kund* sidan har vi placerat två knappar. Knappen Rensa rensar tidigare skrivet innehåll i alla inputboxarna.
Efter att man har matat in alla uppgifter om kunden klickar man på knappen Registrera.
Då kommer vi till filen addkund.asp som är själva förbindelsen till databasen.

Nu gör vi en uppkoppling till databasen och öppnar den för skrivning.
Vi går in i tabellen TKund och lägger in det som har skrivits i inputbox till databasen. Det som är skrivet i inputboxen Efternamn lägger vi till som en ny kunds efternamn, det som är skrivet i inputboxen Förnamn lägger vi till som kundens förnamn i tabellen TKund osv...

Om ingeting skrivs i rutan Född År så skriver vi 0 i databasen. Här gör vi även ett koll att tillräckligt många uppgifter skrivs in för en registrering. Det minsta man måste skriva in i inputboxarna är förnamn och efternamn. Om detta inte uppfylls så får användaren följande felmeddelande:

Inte tillräckligt många uppgifter för en registrering! Förnamn och efternamn är obligatoriska!

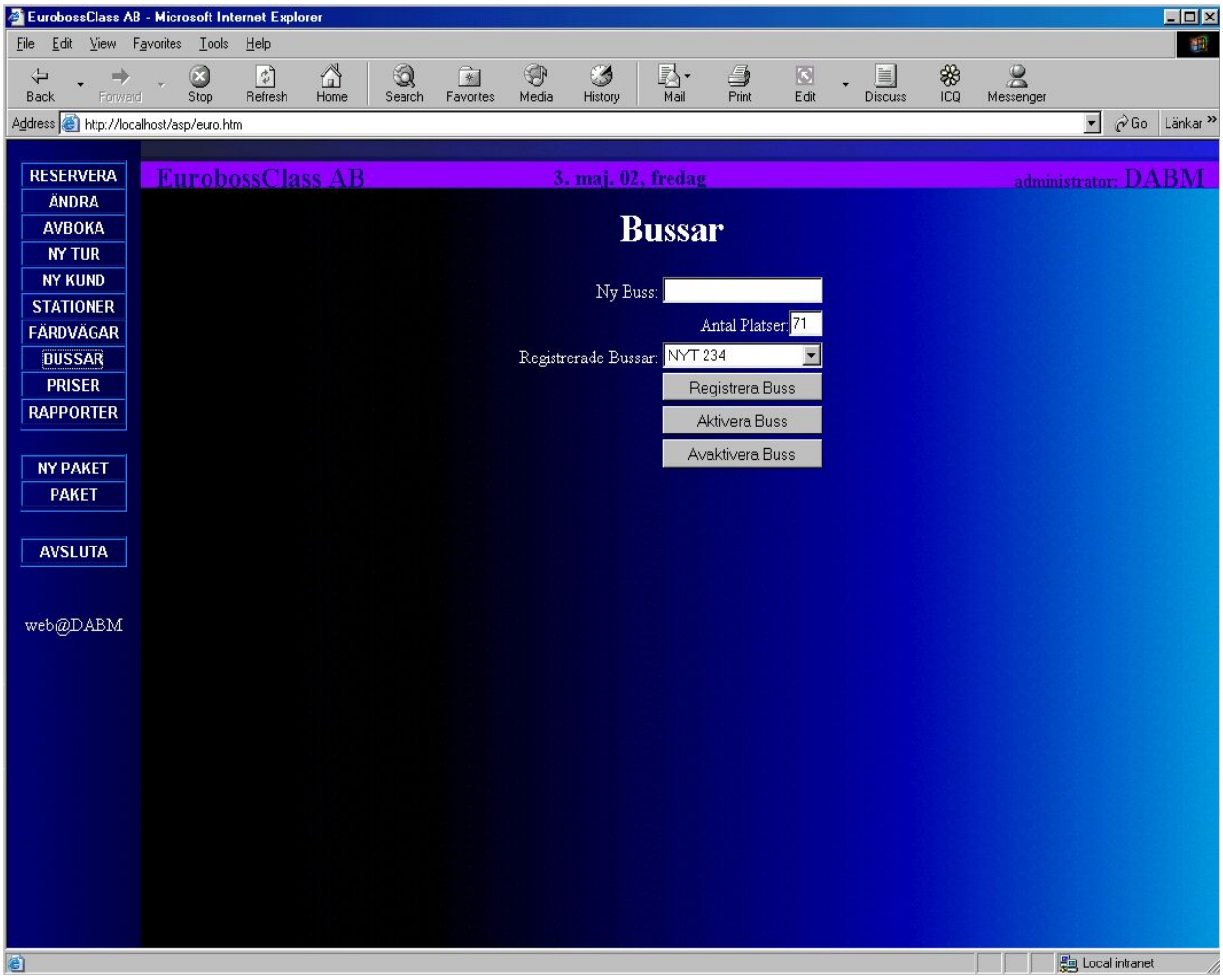
Om man därmed skriver in alla nödvändiga uppgifter om kunden så får man en bekräftelse om att dessa uppgifter har registrerat i databasen.



Figur 4.15 Bekräftelse om registrerad kund

4.7 Ny Buss

Eftersom företaget inom snar framtid troligen kommer att expandera så har vi gjort så att de på ett enkelt sätt ska kunna registrera nya bussar.



Figur 4.16 Busar

Det är i filen buss.asp som vi bygger upp själva utseendet på valet *Bussar* under menyn. Här har vi två inputboxar en listruta och tre knappar. Vi förklarar de genom att börja uppifrån.

I den första inputboxen skriver man registreringsnumret på den nya bussen. Medan den andra skriver man in antalet platser som bussen har. Listrutan visar alla registrerade bussar i databasen genom att hämta deras registrationsnummer från databasen (tabellen TBuss).

Vidare har vi tre knappar Registrera Buss, Aktivera Buss och Avaktivera Buss som alla tar oss till filen regbuss.asp om vi klickar på de.

I filen regbuss.asp såsom i buss.asp görs först en uppkoppling till databasen.

Sedan, beroende på valet (av de tre knapparna) går vi till de olika funktionerna registrera eller aktivera.

```
select case val
case "REGISTRERA BUSS"
        registrera
case "AKTIVERA BUSS"
        aktivera true
case "AVAKTIVERA BUSS"
        aktivera false
end select
```

Om man väljer alternativet *Registrera Buss* går man till funktionen registrera. Här gör man ett koll att man har verkligen matat in den nya bussens registrationsnummer och antalet platser som bussen har. Om så inte är fallet uppmanas man att göra detta genom ett felmeddelande.

Efter att man har matat in registreringsnumret och antalet platser bussen har, så lägger vi till denna information till tabellen **TBuss** och attributen *Aktiv* sätts som true.

Man får också en bekräftelse på att bussen har registrerats:

Ex. Registrerad Buss: ABC 123

Meningen bakom knapparna Aktivera Buss och Avaktivera Buss är att om en buss går sönder eller är på reparation så kan man på ett enkelt sätt avaktivera den utan att behöva ta bort den från databasen. När bussen är i drift igen så aktiverar man den enkelt genom att välja från listrutan vilken buss man vill aktivera och klicka på aktivera knappen.

Det knappen Aktivera Buss respektive Avaktivera Buss gör är att de sätter en checkbox i tabellen **TBuss**, attributen *Aktiv* till true respektive false.

Även här får man en bekräftelse på att bussen är aktiverad eller avaktiverad.

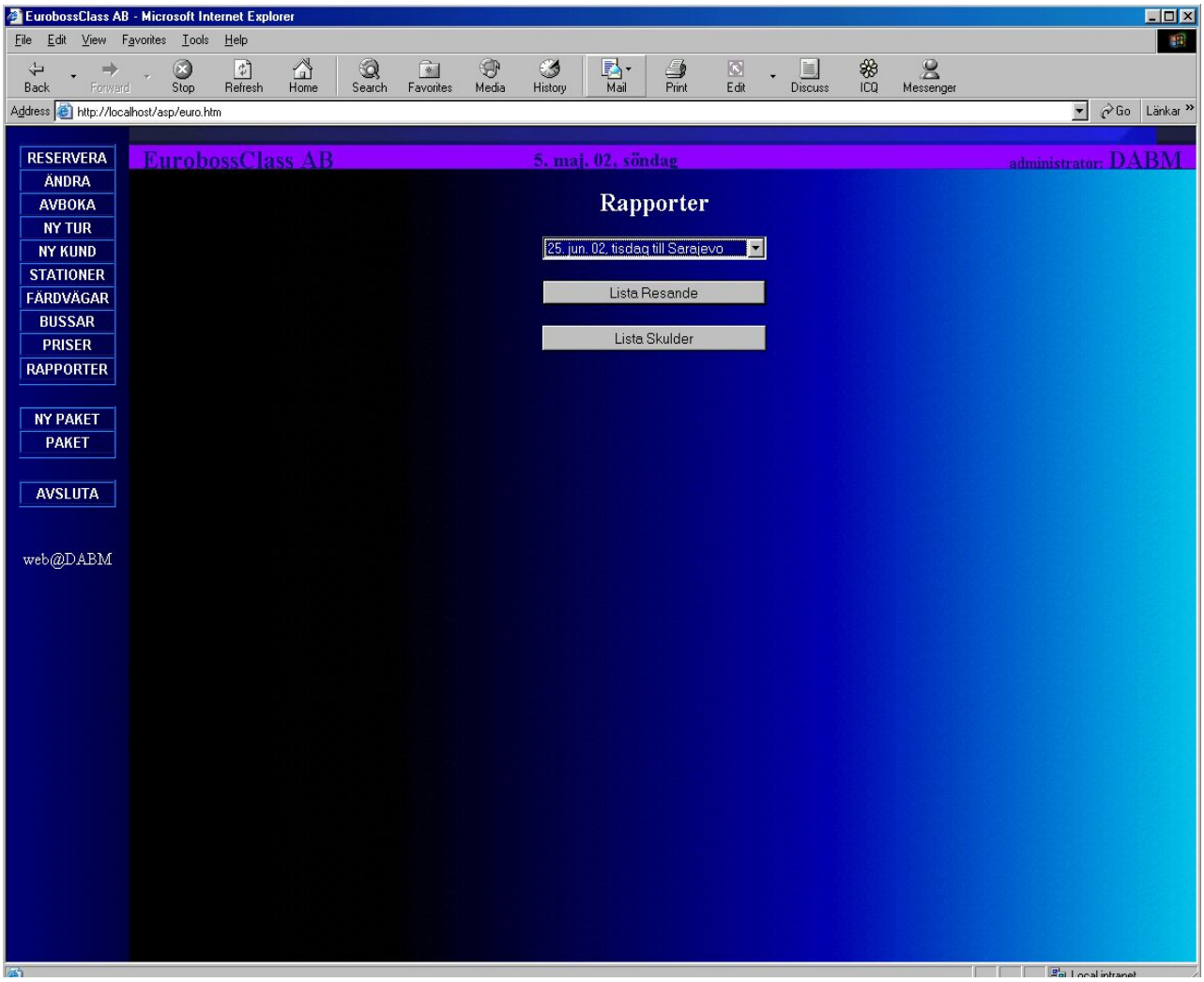
Ex. Aktiverad Buss: ABC 123

Avaktiverad Buss: ABC 123

4.8 Rapporter

I framtiden kommer att under menyn Rapporter ligga alla rapporter som kan vara av intresse för Euroboss. Det kan vara allt ifrån att lista upp passagerare vid ett viss datum till listor på skulder (folk som inte har betalat biljetten). För närvarande har vi gjort så att man kan se passagerarnas namn och efternamn som reser ett visst datum.

Det är filen rapporter.asp vi anropar först då vi navigerar oss till *Rapporter* menyn. I listboxen längst upp på webbsidan visas alla avresedatum både från Stockholm och Sarajevo. Detta har vi gjort genom att från tabellen **TResa** välja datum för avgång och slutstation.

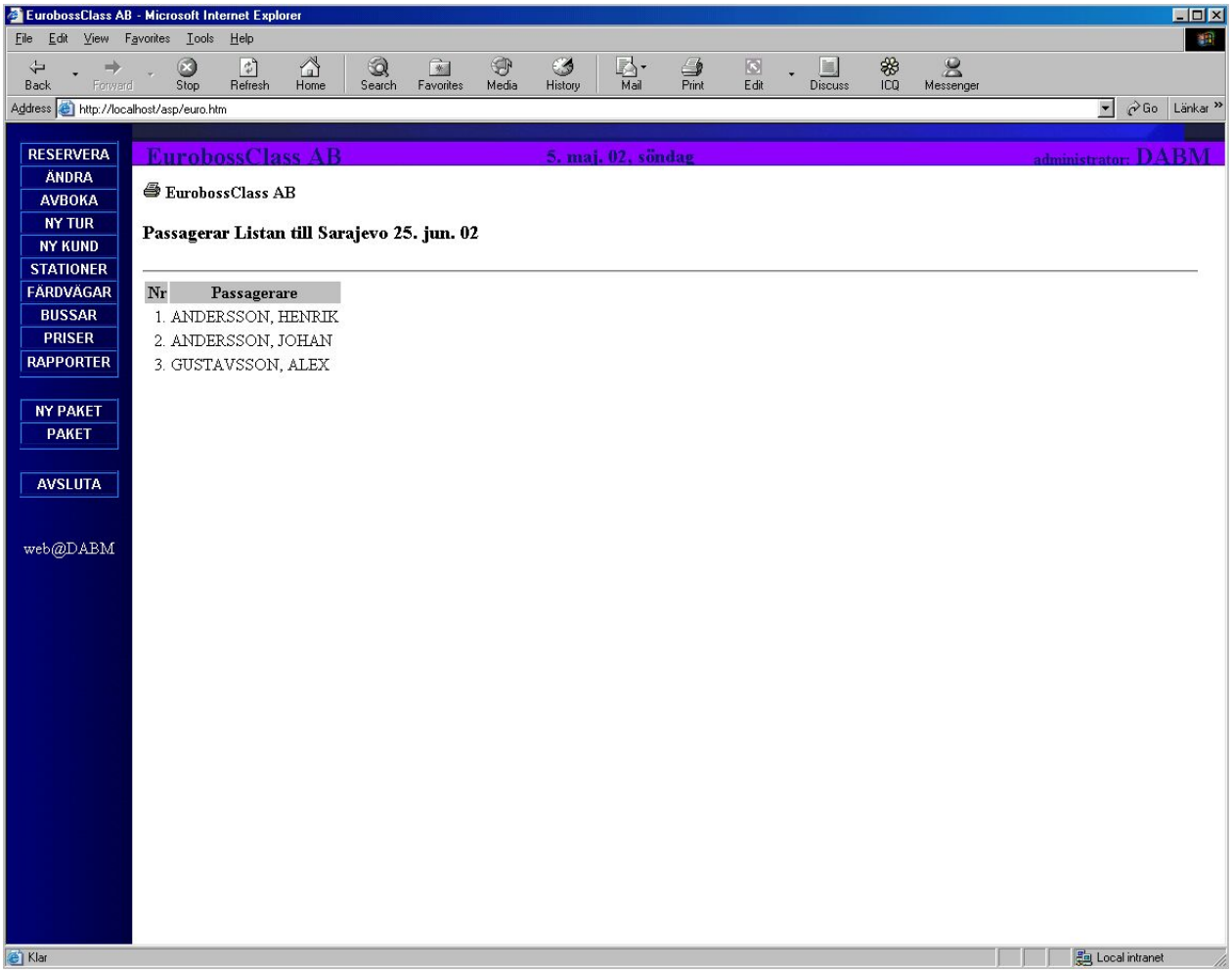


Figur 4.17 Rapporter

Filen menyval.asp sköter händelserna bakom knapptryckningen på *Lista Resande* och *Lista Skulder*. Vi misstänker att det i framtiden kommer att finnas fler knappar här och av just den anledningen har vi gjort en extra fil som sköter händelserna bakom knapptryckningen.

Om man för närvarande klickar på lista skulder så kommer man till filen construction.htm som uppmanar användaren att denna sida är under konstruktion.

Däremot, om man väljer ett datum för avgång i listboxen först och sedan klickar på *Lista Resande* så kommer man till filen lista.asp som visar lista resande för denna avgång.



Figur 4.18 PassagerarListan

Detta åstadkommer vi genom att gå till **TTur** tabellen och visa alla resande för det valda datumet. Vi ordnar passagerarna i bokstavsordning där efternamnet kommer först. Man ser även direkt antalet resenärer i bussen eftersom vi ökar en räknare för varje ny passagerare som hittats.

Webbsidan är konstruerad på så sätt att man ska kunna skriva ut passagerarlistan på ett enkelt och användarvänligt sätt. Genom att lägga till följande kodrad kan vi åstadkomma just detta.

```
<a href="javascript:window.print()"><img src="print.gif"
                        border=0></a>
```

Sidan skrivs ut genom att användaren klickar på skrivarikonen längst upp till vänster på webbsidan.

4.9 Summering

Kapitlet beskriver arbetet för uppbyggnaden av vårt användargränsnitt och de resonemang och funderingar som har förts fram vid konstruktion av gränssnittet.

De generella kraven som vår uppdragsgivare EuroBusClass ställde i kravspecifikationen är att systemet ska vara lätt tillgängligt och gränssnittet ska vara tydligt. Alla fönster ska ha beskrivande text som hjälper användaren vid navigering av programmet.

För att få en idé om hur vårt gränssnitt skulle se ut jämförde vi andra typer av konkurrerande reseföretag som SWEBUS och SJ, där användarnas förväntningar med sådana system undersöktes.

När det gäller utseende på vårt gränssnittet har vi försökt att skapa ett homogent system med samma färg, utseende och ramindelning. Detta för att skapa en trygg miljö för användaren och undvika missförstånd. De texter som förekommer i gränsnittet har samma teckensnitt och teckensnittsstorlek.

Det finns funderingar kring en framtida utveckling av ett kundsystem, där kunden själv kan boka sin egen biljett precis som på SJ:s webbsida. Men det finns för närvarande inget intresse från EuroBoss att ta fram ett sådant kundsystem och vi har inte heller tagit ställning till den frågan.

5 Värdering

Denna rapport är en dokumentation för vårt examensarbete som gick ut på att skapa ett fungerande bokningssystem för vårt uppdragsgivare EuroBossClass AB.

Vi som uppdragstagare kände oss något fundersamma kring hela projektet. Detta med tanke på att vi som studenter inte hade någon tidigare erfarenhet av liknande situation.

Vi kände oss plötsligt som två anställda på ett företag som är utvalda av sitt företagsledning för denna uppgift. Man förväntades sig att vi skulle leverera ett resultat och vi skulle bedömas utifrån detta resultat som ett mått på huruvida vi har lyckats med vår uppgift eller inte. Den känslan var främmande för oss båda. Till att börja med var det svårt att handskas med sådana känslor men allt eftersom arbetet pågick och att vi blev mer involverade i uppgiften övergick denna känsla till något positivt. Det är viktigt att nämna att vår uppdragsgivare EuroBossClass har under hela arbetets gång varit mycket hjälpsamma och stöttat oss till fullo. Deras inställning gentemot oss som uppdragstagare har varit mycket positivt och uppmuntrande. Därför såg vi det hela som utmaning och en upplevelse som skulle bringa alla parter nyttig erfarenhet och kunskap.

Det första steget i designarbetet var att skriva en kravspecifikation utifrån de önskningar och krav som EuroBossClass ställde på systemet. Detta utgjorde en väsentlig punkt för vårt fortsatta arbete och huruvida vi har lyckats med uppdraget eller ej. Det var viktig för oss att avgränsa vårt arbete och koncentrera oss på det primära i vårt arbete. Risken att uppgiften skulle bli för stort och därmed ej hanterbart var överhängande. I det här skedet av arbetet hade vi stor nytta av kursen Systemkonstruktion på Karlstads universitet, där den behandlar just ämnet projekt och systemutvecklingsaspekter. Vi hämtade mycket inspiration vid planering av projektet och konstruktionen av vårt användargränssnitt, såsom utseendet på webbsidan, färg och ramindelning(frames) från just denna kurs.

När vi väl hade enats om innehållet i kravspecifikationen skapades en relationdatabas och därefter påbörjade arbetet med prototypen.

Våra ambitioner under hela projektets gång var att implementera vårt system utifrån de krav som var ställda på oss. Systemet skulle vara så generell och flexibel som möjligt och det ska vara enkelt att bygga ut den med flera funktioner.

Arbetet med prototypen pågick parallellt med konstruktion och implementation av databasen under sex veckor.

Efter de inledande veckorna var alla viktiga beslut fattade om vilket programspråk och vilken databas vi skulle använda oss av och vi kände för första gången att vi hade något konkret att skriva om och dokumentationen hade börjat på allvar.

När vi väl började med implementeringen visade det sig att det var svårare än vi hade föreställt oss. Vi var tvungna mycket tidigt att ta ställning till några viktiga punkter. Ett av de första problemen var att utforma en entitet/relation modell (E/R-modell). Det var mycket svårt att veta vilka attribut man ska ta med, hur man ska koppla ihop de olika tabellerna och bestämma relationerna mellan tabellerna. En annan designaspekt var hur vi skulle lägga upp våran kod på nätet, eftersom ASP koden exekveras där.

Om vi skulle lägga den på en webbhotell så var vi tvungna att konstant skicka den modifierade koden till servern via något (FTP) File Transfer Protocol program. Detta skulle göra felsökningen mycket svårare och dessutom ta mycket tid.

Då kom vi på att vi kunde använda våran egen dator som server och exekvera koden på den. Vi installerade PWS (Personal Web Server) från Microsoft och skapade en DSN på samma dator. På så sätt kunde vi nu exekvera programmet på en ”virtuell server”, vilket sparade mycket tid åt oss och dessutom gjorde felsökningen mycket lättare.

När det gäller fönsteröppning och ramindelning har vi använt oss av JavaScripts. Till skillnad från ASP körs dessa på klientsidan och rekommenderas varmt när gäller fönsterhantering.

Även arbetet med implementationen tog längre tid än beräknat. Detta berodde på bland annat att vi hade ingen erfarenhet av ASP sedan tidigare. En hel del tid gick åt att lära sig språket och dess funktionalitet. ASP är ett relativt enkelt programmeringsspråk att lära sig och även utan större förkunskaper kan vanliga användare designa riktigt avancerade webbplatser [6]. Det fanns även lite arrogans från vår sida när det gällde implementeringen, då vi inte insåg dess omfattning och den tid som krävdes för att fullborda programmet.

När det gäller framtida aspekter finns stora möjligheter att vidareutveckla själva systemet och lägga till fler funktioner. Grundtanken från början var att skapa grundpelarna för detta system och bygga den så generellt som möjligt så att det skulle var anpassningsbar för framtida ändringar. Vårt arbete var ju indelad i två delar

Primära mål:

1. implementera ett system där man ska kunna registrera en ny avgång för bussen.
2. Genom att välja den registrerade avgången ska man på ett enkelt sätt kunna lägga till passagerare för denna avgång.
3. Om passageraren finns med i databasen sedan tidigare, dvs om passageraren har rest med EuroBoss förut skall man kunna hitta honom/henne i databasen med en sök funktion.
4. Finns passageraren inte med i databasen ska man kunna registrera honom/henne som en ny passagerare i databasen.
5. Det ska gå att boka en sittplats i bussen åt en vald kund. En redan upptagen plats måste vi markera på något sätt och ”spärra” så att det inte går att boka en sittplats två gånger.
6. Det ska på ett enkelt sätt gå att registrera en eller flera nya bussar.

Alla dessa krav är uppfyllda med ett undantag av registrering av en ny tur (datum för avresa utifrån färdvägen, val av buss och slutdestinationen). Detta gör vi för närvarande manuellt i databasen i tabellen **TResa**, men vi har redan inlett arbetet med att kunna göra registreringen av ny tur automatiskt på webbsidan. (se avsnitt 4. Användargränsnitt och Implementation).

Sekundära mål:

1. Hantering av peketleverans.
2. Skapa nya färdvägar som innehåller nya busstationer.
3. Ändring och avbokning av biljetter för specifik kund ska tillåtas.
4. Räkna priser för en resa inom de olika zonerna är funktioner som är aktuella för framtiden.

Dessa önskemål kommer vi att beröra efter denna rapport.

Vi har i nuläget lyckats att konstruera ett fungerande system som är både tillförlitligt och som överenskommer med de krav som vår uppdragsgivare ställde på oss. Vi är mycket nöjda med våra egna insatser och tycker att vi har uppnått de mål som var uppsatta från början. Det är av stor glädje för oss att även EuroBossClass delar samma uppfattning som vi. De har testat vårt system och imponerats över den prestation som vi har åstadkommit.

6 Slutsats

Vi vet idag att vi har lyckats skapa ett system som både fungerar bra och är tillförlitligt. Systemet ska kunna klara av att utrustas med fler funktionaliteter. Som vi nämnde i början så kommer systemet att klara de primära mål som är uppsatta, vilket är att söka eller lägga till en ny passagerare till databasen. Systemet ska dessutom klara av reservation av en plats åt passageraren vid en angiven avgångstid. Man ska också på ett enkelt sätt kunna registrera en ny avgång (alltså datum för en ny resa) och nya bussar (ifall företaget köper fler bussar).

Vidare önskningar från vår uppdragsgivare som pakethantering, leverans och prissättning baserad på zoner är funktioner som vi ej har kunnat ägna någon som helst tid till, men vi är beredda att om det så önskas jobba med vidareutveckling av systemet efter denna rapport.

Slutligen kan vi konstatera att arbetet har varit mycket stimulerande och lärorikt. Vi har varit kapabla att tillmötesgå de krav som var ställda på oss och vi tycker att vi har lyckats göra ett bra resultat.

Referenser

[1] Erik Ronne, ’ASP’, ACTIVE SERVER PAGES. Docendo Läromedel AB, 1999

[2] Web Studio, http://www.webstudio.com

[3] David Flanagan, JavaScript: The Definitive Guide, 4th edition (December 15, 2001)

[4] Burholt Olle, Lennartzon Per, Grunderna I Acess 97, Pagina Förlag AB, 1997.

[5] Date C.J., An introduction to database systems, Addison-Wesley, 6:e upplagan 1994.

[6] G. Andrew Duthie, Microsoft ASP.NET Step by Step, Microsoft, December 19, 2001

A Förklaring till datamodellen

A.1 Tabell TTur

Tabell	**Beskrivning**
TTur	**Innehåller info om en viss tur.**
TurID	Den unika numret som en tur erhåller.
ResaId*	Identifierar en resa.
KundId*	Den unika numret som en kund får.
Avstigning	Numret på station där passagerare stiger av.
Påstigning	Numret på station där passagerare stiger på.
Retur	Tur och retur biljett.
Pris	Pris för en resa.
PlatsNr	Sittplatsen i bussen för en passagerare.
Inbetalt	Pengarna som kunden betalat in.
KontrollNr	Euroboss interna kontroll nummer.
FörarMedd	Meddelande till föraren.
Informerad	Föraren informerad om meddelandet.
AdminMedd	Medellande till Administratören. (T.ex. om någon kund måsta ha en speciell plats.)
Admin	Administratörens beteckning.

Fältnamn	**Nyckel**	**Datatyp**	**Fältstorlek**	**Format**	**Oblig.**	**Index**	**Dubb.**
TurID	Primär	Autonumber	Long int.		X	Yes	No
ResaId	Främmande	Number	Long int.		No	Yes	Yes
KundId	Främmande	Number	Long int.		No	Yes	Yes
Avstigning		Number	Long int.		No	No	No
Påstigning		Number	Long int.		No	No	X
Retur		Yes/No	X		No	No	X
Pris		Number	Long int.	Gen.num.	No	No	X

platsNr		Number	Long int.		No	No	X
Inbetalt		Number	Long int.	Gen.num.	No	No	Yes
KontrollNr		Text	50		No	No	No
FörarMedd		Text	50		No	No	X
Informerad		Yes/No	X		No	No	X
AdminMedd		Text	50		No	No	X
Admin		Text	20		No	No	X

A.2 Tabell TResa

Tabell	**Beskrivning**
TResa	**Innehåller info om en viss resa.**
ResaId	Identifierar en resa.
Datum	Datum för en viss resa.
SlutStation	Slutstation för en resa.
Res	Antal återsående platser i bussen.
BussId	Numret på bussen som kör. (om flera)
Relation*	För närvarande bara en(Bihac). I framtiden ska det gå att skapa flera Relationer med nya stationer.

Fältnamn	**Nyckel**	**Datatyp**	**Fältstorlek**	**Format**	**Oblig.**	**Index**	**Dubb.**
ResaId	Primär	AutoNumber	Long int.		X	Yes	No
Datum		Date/Time	X		No	No	X
Slutstation		Text	50		No	No	X
Res		Number	Long int.		No	No	X
BussId		Number	Long int.		No	No	X
Relation	Främmande	Text	50		No	Yes	Yes

A.3 Tabell TKund

Tabell	**Beskrivning**
Tkund	**Innehåller info om en specifik kund.**
KundId	Den unika numret som en kund erhåller.
Efternamn	Kundens efternamn.
Förnamn	Kundens förnamn.
Telefon	Kundens telefon .
Stad	Staden kunden bor i.
Adress	Kundens adress.
Mobil	Kundens mobiltelefon.
TelBosna	Kundens telefon i Bosnien.
Mail	Kundens mail.
Ålder	Kundens ålder.
Anmärkning	Övrig information.
PostNr	Post numret till kunden.

Fältnamn	**Nyckel**	**Datatyp**	**Fältstorlek**	**Format**	**Oblig.**	**Index**	**Dubb.**
KundId	Primär	Autonumber	Long int.		X	Yes	No
Efternamn		Text	50		No	No	X
Förnamn		Text	50		No	No	X
Telefon		Text	50		No	No	X
Stad		Text	50		No	No	X
Adress		Text	50		No	No	X
Mobil		Text	50		No	No	X
TelBosna		Text	50		No	No	X
Mail		Text	50		No	No	X
Ålder		Number	Long Int.		No	No	X
Anmärkning		Text	100		No	No	X
PostNr		Text	50		No	No	X

A.4 Tabell TBuss

Tabell	**Beskrivning**
TBuss	**Innehåller info om buss(för tillfället bara en buss)**
BussId	Identifierar en buss
RegNr	Registernumret på en buss
Aktiv	Huruvida om bussen är bokad att köra el. ej.
AntalPlatser	Totala antalet sittplatser på bussen.

Fältnamn	**Nyckel**	**Datatyp**	**Fältstorlek**	**Format**	**Oblig.**	**Index**	**Dubb.**
BussId	Primär	AutoNumber	Long int.		X	Yes	No
RegNr		Text	50		No	No	X
Akriv		Yes/No	Yes/No		No	No	X
AntalPlatser		Number	Long int.		No	No	X

A.5 Tabell TStationer

Tabell	**Beskrivning**
TStationer	**Innehåller info om stationerna.**
StationID	Identifierings nummer för en station.
Station	Namn på stationen. Ex.Stockholm
Beskrivning	Beskrivning av stationen. Ex. IKEA vid Hydro-pumpen.

Fältnamn	**Nyckel**	**Datatyp**	**Fältstorlek**	**Format**	**Oblig.**	**Index**	**Dubb.**
StationID	Primär	AutoNumber	Long int.		X	Yes	No
Station		Text	50		No	No	X
Beskrivning		Text	50		No	No	X

A.6 Tabell TFard

Tabell	**Beskrivning**
Tfard	**Innehåller info om relationer.**
Färdväg	Identifierar en ny färdväg för en resa.

Fältnamn	**Nyckel**	**Datatyp**	**Fältstorlek**	**Format**	**Oblig.**	**Index**	**Dubb.**
Färdväg	Primär	Text	50		No	Yes	No

A.7 Tabell TST

Tabell	**Beskrivning**
TST	**Innehåller info om en viss resa.**
SRCID	Identifierare
StationId*	Namn på stationen. Ex.Stockholm
Färdväg*	Identifierar en färdväg för en resa.
StNr	Ordningsnumret på en station.
Zon	Identifierar en zon.
TidSa	Avgångstid från en station då bussen ska till Sarajevo.
TidSt	Avgångstid från en station då bussen ska till Stockholm.

Fältnamn	**Nyckel**	**Datatyp**	**Fältstorlek**	**Format**	**Oblig.**	**Index**	**Dubb.**
SRCID	Primär	AutoNumber	Long int.		X	Yes	No
StationId	Främmande	Number	Long int.		No	Yes	Yes
Färdväg	Främmande	Text	50		No	Yes	Yes
StNr		Number	Long int.		No	No	X
Zon		Number	Ling int.		No	No	X
TidSa		Text	50		No	No	X
TidSt		Text	50		No	No	X

A.8 Tabell TAdmin

Tabell	**Beskrivning**
TAdmin	**Innehåller info om sidans administratörer.**
AdminID	Identifieringsnummer för en administratör.
Admin	Administratörens namn eller nick.
Password	Administratörens password.
Beteckning	Beteckning.

Fältnamn	**Nyckel**	**Datatyp**	**Fältstorlek**	**Format**	**Oblig.**	**Index**	**Dubb.**
AdminID	Primär	AutoNumber	Long int.		X	Yes	No
Admin		Text	50		No	No	X
Password		Text	50		No	No	X
Beteckning		Text	50		No	No	X

B Filöversikt