Avdelningen för Datavetenskap

Sten Hansson Bjerke & Anders Friberg

# Distribution av kalenderdata från Exchange på HTML-sidor

## Distribution of calendar data from Exchange on HTML-pages

Datavetenskap
C-uppsats

Datum/Termin: 06-01-19
Handledare: Kerstin Andersson
Examinator: Stefan Lindskog
Löpnummer: 2006:07

# Distribution av kalenderdata från Exchange på HTML-sidor

**Sten Hansson Bjerke**

**Anders Friberg**

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

Sten Hansson Bjerke

Anders Friberg

Godkänd, 060119

Handledare: Kerstin Andersson

Examinator: Stefan Lindskog

# Sammanfattning

Uppsatsen handlar om det arbete vi utfört åt Prevas AB i Karlstad. Uppdraget gick ut på att synkronisera tre olika systems kalenderfunktioner så att det som skrivs i en kalender syns i alla. För att klara det behövde vi först undersöka hur de tre systemen fungerade vilket delade upp arbetet i en utredningsfas och en konstruktionsfas. Utredningsfasen ledde till att vi begränsade arbetet till att endast omfatta en envägskommunikation mellan två av systemen i konstruktionsfasen. Detta arbete beskrivs i detalj i uppsatsen.

# Distribution of calendar data from Exchange on HTML-pages


## Abstract

*Our paper is about the work we performed for Prevas AB in Karlstad. The assignment was to synchronize three different systems calendar functions, so that what's going into one calendar is visible in all the others. To manage this we needed first to survey how the three systems functioned, which split the work into an examination phase and a construction phase. The examination phase led to us limiting the work to only concern a one-way communication between two of the systems in the construction phase. This construction work is explained in detail in this paper.*

# Innehållsförteckning

# Figurförteckning

# 1 Inledning

Syftet med detta arbete har varit att utreda hur en datakoppling mellan 3 olika system skulle kunna skapas och därefter skapa denna koppling. Uppsatsen kan sägas bestå av tre delar: en utredningsdel som utmynnar i ett lösningsförslag, en del som beskriver vår implementation av denna lösning och till sist en resultatdel som beskriver den färdiga lösningen. Utredningsdelen omfattar anledningen till arbetet, beskrivning av systemen före arbetets start, först översiktligt sedan mer i detalj. Utredningsdelen fortsätter sedan med att beskriva hur undersökningen genomfördes och till slut vad implemetationsarbetet bestämdes att bestå av. I implementationsdelen går vi först igenom de verktyg och språk vi använde i vår implementering av kalendern och den miljö utvecklingen skulle ske i. Därefter följer en beskrivning av själva implementationen och de största problemen vi stötte på, samt lösningarna på dessa. Resultatdelen innehåller en beskrivning av det färdiga arbetet uppdelad i två kapitel. Första kapitlet (kapitel 4) ger en beskrivning av hur resultatet ser ut och hur det fungerar och andra kapitlet (kapitel 5) är ett uppföljningskapitel där vi har med den respons vi har fått på vårt arbete från uppdragsgivaren och vad vi hade kunnat göra bättre eller annorlunda.

# 2 Undersökning

Detta kapitel beskriver bakgrunden till arbetet och de olika delarna i det system vi undersökt. Vi har även dokumenterat de viktiga delarna i vår undersökning. Som avslutande del kommer ett lösningsförslag.

## 2.1 Bakgrund

Här beskriver vi de förutsättningar som fanns innan vi började vårt arbete, vilka mål det var tänkt att vi skulle uppnå och hur dessa ändrades efterhand.

### 2.1.1 Förutsättningar

Prevas har traditionellt använt ett system för att hantera var dess personal befinner sig. Exempelvis kan personalen med detta system tala om att man är på semester, att man är på möte 10-11, att man gått för dagen etc.

Prevas system består av 3 delar. En del utgörs av en Microsoft Exchange-server som arbetar mot personalens Outlook-klienter. Andra delen är en Siemens telefonväxel med ett managementprogram från Färöarna vid namn Totalview. Den sista delen utgörs av ett av Prevas egenutvecklat intranät. Varje del i systemet har som funktion att tids- och lokalbestämma personal i olika utsträckning, det vill säga både växeltelefonister och medarbetare ska kunna ta reda på var en kollega är vid en specifik tid. Telefonväxeln har även funktionen att koppla vidare samtal enligt schemalagd önskan, t.ex. till hemtelefon vid hemarbete. Prevas har i nuläget produktutveckling endast i Karlstad. I övriga landet är verksamheten till största delen konsultering. Alltså jobbar de flesta konsulterna på plats hos kunden och sällan på sitt kontor. Därför är det extra viktigt att kontaktinformationen hålls uppdaterad så att de hela tiden kan nås.

Konsultpersonal på plats hos kund har inte tillgång till Totalview och kan därför ej direkt påverka vidarekopplingsschemat. Webbaserad Outlook och intranätet går däremot bra att använda. I teorin ska informationen i alla delsystem vara synkroniserad så att t.ex. om man bokar tid för ett möte genom intranätet så ändras telefonväxelns schema automatiskt så att omkoppling sker under mötet. Men som det ser ut i nuläget finns ingen koppling mellan intranätet och någon av de andra delarna. Kopplingen mellan Exchange och Totalview är bristfällig. Alltså är i nuläget informationen i delsystemen inte synkroniserad. Vår uppgift är alltså följande:

- Utreda om och hur man kan koppla ihop delsystem för synkronisering.
- Ge förslag på en eller flera olika lösningar för vilket delsystem som ska vara master.
- Implementera den valda lösningen.

I nuvarande form av systemet kan information matas in i vilket delsystem som helst. Inmatningsmöjligheterna varierar dock mellan delsystemen.

### 2.1.2 Måländringar

Vårt ursprungliga mål var som beskrivs i stycke 2.1.1 att få de tre delarna Totalview, Outlook och intranät i Prevas tidsbokningssystem att kommunicera med varandra. Från början trodde vi att detta skulle bli ett förhållandevis enkelt arbete, men det visade sig efter hand att så inte var fallet. Det första problemet vi mötte var svårigheten att försöka få fram information

om ett kommersiellt program, i det här fallet Totalview, som kan vara användbart för en programmerare. Kanske inte så konstigt om man tänker efter eftersom en utvecklare naturligtvis vill skydda sin kod, göra utvecklingen själva och sälja sina egna lösningar. Inte så bra för vår del dock eftersom det hindrade oss från att ens förstå hur Totalview fungerar. Med sådana motgångar kände vi oss tvungna att göra utvecklarna till Totalview till viljes och låta dem sköta utvecklingen själv, vilket det också visade sig att de gjorde då det under den tid vi arbetade på Prevas skedde en uppgradering av Totalview som förbättrade kopplingen till Outlook. För oss återstod alltså länken mellan intranätet och Exchange.

Vi insåg att en tvåvägskommunikation mellan intranätet och Exchange skulle bli alltför omfattande och dessutom inte var nödvändig. Exchangeinformationen behöver endast synas på intranätet. Att manipulera denna skall endast göras av personen som ändå har tillgång till denna via Outlook. Vi presenterar informationen från Exchange parallellt med den gamla informationen från intranätsdatabasen. Det här betyder att vi nu inte längre gör någon egentlig synkronisering utan istället visar upp information från ett system i ett annat.

## 2.2 Komponentbeskrivning

I denna del beskriver vi översiktligt de olika delarna i systemet.

### 2.2.1 Microsoft Exchange Server

Microsoft Exchange Server (MES) är en mjukvara för kommunikation och samverkan. Mjukvaran kör på en server som låter dig skicka och ta emot e-post och annan form av interaktiv kommunikation genom ett nätverk. MES är designad att användas tillsammans med en mjukvaruklient som t.ex. Microsoft Outlook. [1]

### 2.2.2 Outlook-klient

”Microsoft Office Outlook 2003 är ett personligt informationshanterings- och kommunikationsprogram där du hanterar e-post, kalendrar, kontakter och annan personlig och grupprelaterad information.” [2]

I vårt fall koncentrerar vi oss på kalenderfunktionen eftersom det är den informationen som ska synkroniseras med de övriga systemen. Informationen som hanteras i Outlook är dock lagrad i Exchange så vi kommer ej ha någon direkt kontakt med Outlook förutom i testsyfte.

### 2.2.3 Totalview

Totalview är ett program utvecklat för att kunna kontrollera en Siemens Hipath 3000 telefonväxel. Tyngdpunkten ligger på att kunna bestämma till vilken telefon man vill koppla inkommande samtal. Det finns även en ganska begränsad möjlighet till schemaläggning.

Totalview är uppdelat i server och klient för att kunna fungera distribuerat. Klientdelen finns i tre olika lägen: administratör, receptionist och vanligt användarläge.

- En vanlig användare kan endast manipulera sin egen information d.v.s. endast ändra på sitt eget omkopplingsschema.
- Receptionisten har även översikt och möjlighet att se hela personalens omkopplingsinformation och eventuellt ändra i den.
- Administratören har utöver detta möjligheter att ändra inställningar.

### 2.2.4 Intranät

Intranätet är uppbyggt av ASP-sidor som genererar HTML-kod. Eftersom det är webbaserat så kan det nås från hela världen från en webbläsare. Det finns väldigt mycket funktionalitet på sidorna men vi koncentrerar oss på kalendern för de anställda. Här kan man välja en viss dag och beskriva vad man gör den dagen, detta kommer sedan att visas i kalendern som vem som helst på företaget kan titta i. Din närvarostatus kommer även att visas i listan av anställda. Inmatningsmöjligheterna i kalendern ligger mellan Totalview och Outlook.

## 2.3 Intranät

Här beskrivs hur intranätets kalender fungerar för en vanlig användare och vilka funktioner som finns tillgängliga. Vi beskriver även hur intranätet är uppbyggt och hur det kommunicerar med SQL-databasen.

### 2.3.1 Kalender

| Sign. | KP | | Stat | Efternamn / förnamn | Befattning | Tel. arbete | Tel. mobil |
|---|---|---|---|---|---|---|---|
| ASFR | | | | Friberg, Anders | | | |
| KEGU | | | | Gustavsson, Kent | | [redacted] | [redacted] |
| MAHA | | | | Hagberg, Magnus | | [redacted] | [redacted] |
| JNHA | | | | Hansson, Johan | | [redacted] | [redacted] |
| STHB | | | | Hansson Bjerke, Sten | | | |

*Figur 2.1 Telefonlista (telefonnummer övertäckta)*

I Figur 2.1 finns ett utklipp av vyn från sidan ”medarbetare” i Prevas intranät. Den visar alla anställda på ett valfritt kontor, även specialfall som t.ex. support, med signatur, namn, befattning och telefonnummer. Det finns även lokaler med i listan där kalendrarna fungerar som bokningssystem. De tre ikonerna är: personlig beskrivning (konsultprofil), mail och kalender. Statusikonens färg är beroende på närvarostatus. Den är grön vid närvarande status, röd vid frånvarandestatus och blå vid möte.

Genom att trycka på statusikonen kommer man till den anställdes kalender, se figur 2.2.

**Kalender** [Sten Hansson Bjerke]

Till medarbetarlistan
Till semesterlistan

Förra Nästa

Mars 2005

| V:a | Måndag | Tisdag | Onsdag | Torsdag | Fredag | Lördag | Söndag |
|---|---|---|---|---|---|---|---|
| 9 | | 2005-03-01 | 2005-03-02 | 2005-03-03 | 2005-03-04 | 2005-03-05 | 2005-03-06 |
| 10 | 2005-03-07 | 2005-03-08 | 2005-03-09 | 2005-03-10 | 2005-03-11 | 2005-03-12 | 2005-03-13 *Arbetar hemma* |
| 11 | 2005-03-14 | 2005-03-15 | 2005-03-16 | 2005-03-17 | 2005-03-18 | 2005-03-19 | 2005-03-20 |
| 12 | 2005-03-21 | 2005-03-22 | 2005-03-23 | 2005-03-24 | 2005-03-25 | 2005-03-26 | 2005-03-27 |
| 13 | 2005-03-28 | 2005-03-29 | 2005-03-30 | 2005-03-31 | | | |

Förra Nästa

*Figur 2.2 Kalendern*

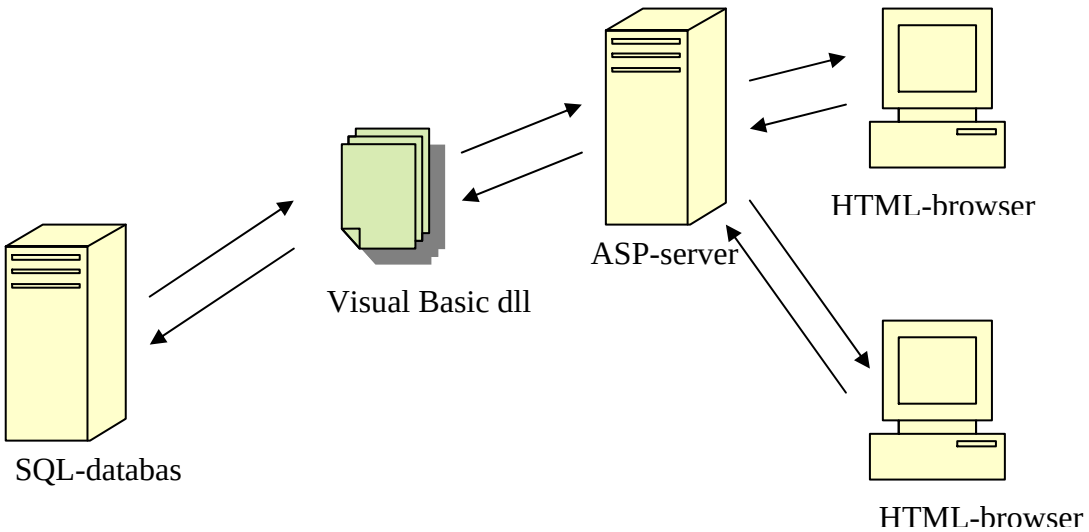
Kalendern visar månadsvis den anställdes tidsplanering. Man ser även på sidan den anställdes namn. Dagarna visas med datumet i formatet år-månad-dag. Dagens datum understryks genom att visas med en ljust blå färg. Den eventuellt inmatade statusen visas under datumet och den av användaren inmatade textsträngen visas under statusen. Man kan bläddra mellan månader genom länkarna ”Förra” och ”Nästa”. För att ändra på en dags innehåll trycker man på datumet som är en länk till redigeringssidan, se figur 2.3.


*Figur 2.3 Redigeringssidan*

På redigeringssidan hämtas datum och vecka automatiskt från dagens datum medan slutdatum används i de fall där en bokning sträcker sig över flera dagar eller månader t.ex. vid semester. Information är den text den anställde vill förmedla om sin bokning. Började och slutade används vid kortare bokningar t.ex. ett möte från 8.00 till 10.00. Statusen kan sättas till några förinställda värden och det påverkar färgen på statusikonen i medarbetarlistan, figur 2.1. ”När tillbaka?”-rutan kan användas vid sådana tillfällen när man har ett möte utanför kontoret och sluttiden inte betyder att man är tillgänglig direkt efter. Innehållet i denna ruta skrivs dock aldrig ut någon annanstans, så den är ointegrerad.

### 2.3.2 Konstruktion

Intranätet skapades av Prevas för internkommunikation inom företaget och det är uppbyggt enligt figur 2.4. Det krävs en inloggning för att kunna komma åt intranätet från en webbrowser utanför företaget. Användaren anropar en ASP-sida på ASP-servern som genererar HTML-kod. ASP-sidorna använder objekt som är instanser av klasser som ingår i dll:en för att läsa och spara information i databasen. Observera att ASP-sidorna utöver databas-access även har mycket funktionalitet som inte kräver dll-filen.



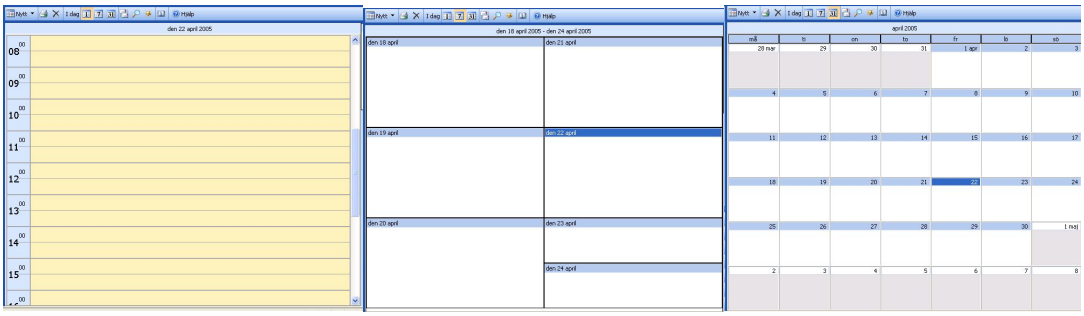

*Figur 2.4 Intranätets uppbyggnad*

ASP står för Active Server Pages. ASP exekverar inuti IIS (Internet Information Services) som är en komponent i Microsofts servrar. En ASP-fil kan innehålla text, HTML, XML och script. När en browser begär en ASP-fil ger IIS begäran till ASP-motorn som exekverar scriptet i filen och returnerar filen som ren HTML-kod till browsern. All exekvering sker alltså på serversidan.

Dll:en är skriven i Visual Basic 6.0 och registrerad på ASP-servern. Dll:en innehåller ett flertal klasser som alla sköter sin del i databashanteringen. En klass t.ex. kan hantera den anställdes information medan en annan klass tar hand om den anställdas kalender. Dll-filens klasser kommunicerar genom SQL(Structured Query Language) med databasen.

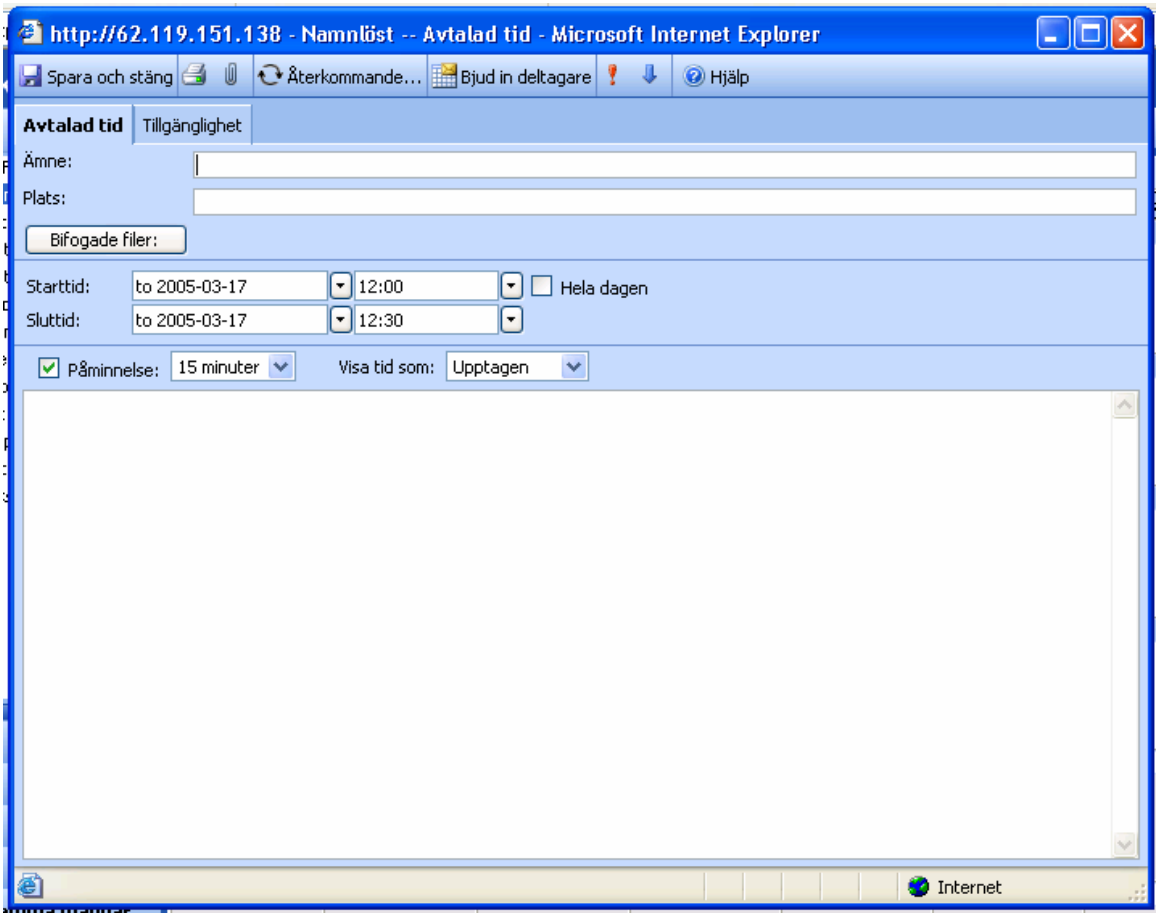
## 2.4 Outlook

Förutom mailservice har Outlook också en kalenderdel. Outlook-kalendern kan ses i tre olika vyer: en dag, en vecka eller en månad, se figur 2.5. Dagsvyn visar ett detaljerat schema över dagen indelad i halvtimmar medan vecko- och månadsvyn visar hela dagar med dagens datum markerat i blått.



*Figur 2.5 De tre kalendervyerna f.v. dagvy, veckovy och månadsvy*

Vilken vy man än väljer kommer man med ett dubbelklick på en dag till en editeringssida, se figur 2.6.




*Figur 2.6 Bokningsdialog*

I ämnesraden i bokningsdialogen, se figur 2.6, skriver man in vad bokningen gäller t.ex. ett möte, och i platsraden var mötet äger rum. Beroende på vilken vy man klickar sig in i sidan från ställs start- och sluttid in olika. Från dagsvyn ställs starttiden till det klockslag man klickar på medan de andra vyerna ställer starttiden till den nuvarande närmast kommande halvtimmen. Sluttiden sätts som standardvärde till en halvtimme efter starttiden. Om bokningen gäller hela dagen kan man kryssa i det. Om bokningen är t.ex. ett möte kan man om man vill välja att programmet ska påminna på en förutbestämd tid innan starttiden. ”Visa tid som” kan användas för att färgkoda bokningarna i endagsvyn. Det har ingen effekt i de andra vyerna.

I verktygsraden ovanför flikarna finns alternativet återkommande. Där kan man ställa in om man har en bokning som gäller flera dagar i följd eller följer ett visst mönster t.ex. varannan onsdag.

Alternativet bjud in deltagare används om man vill ta reda på vilka som kan komma på ett möte. Mötestiden skickas då ut till alla inbjudna som får tacka ja eller nej (förutsatt att de också har Outlook).

Alternativen hög och låg prioritet visas med ikoner med ett utropstecken respektive nedåtriktad pil i verktygslisten men verkar inte spela någon som helst roll.

## 2.5 Totalview

Totalview var lite svårare att undersöka funktioner för eftersom vi inte kunde experimentera som vi ville. Vi fick inte något eget konto till det utan fick låna ett från en anställd, och det vore inte så bra att ändra hans inställningar.

Det första som syns när man loggat in är users-fliken, se figur 2.7, som är en lista över användare med olika fält. En användare behöver inte nödvändigtvis vara en person utan kan vara t.ex. en telefonsupport. Nästa fält, state, visar en färgkodad boll. Färgerna motsvarar ett tillstånd t.ex. ledig eller upptagen, se figur 2.9. I Prevas fall har man sedan tidigare gått in och ändrat på färgkodningen för att bättre passa in mot intranätet.

Call-fältet visar statusen på användarens telefon, om den är upptagen, vidarekopplad eller ledig. Vid vidarekoppling syns även det telefonnummer som rings upp.

”Local” visar användarens internnummer och ”mobile” dess mobiltelefonnummer.

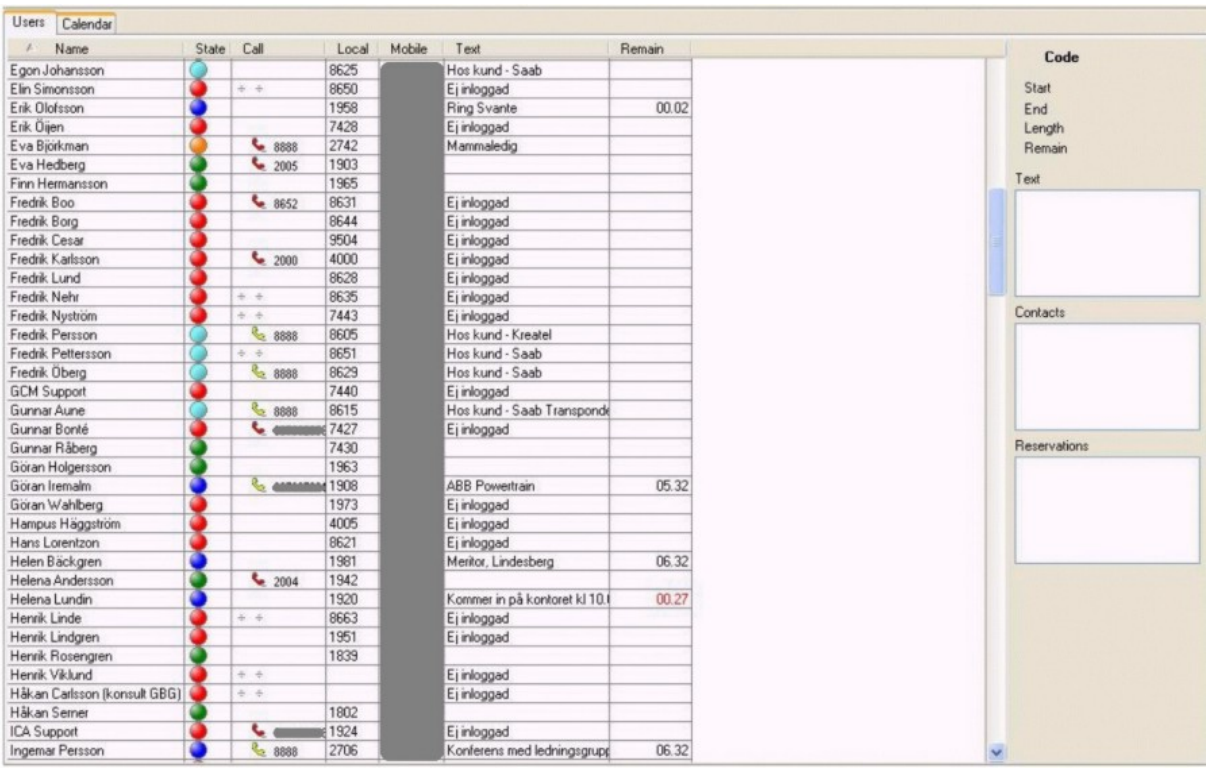
Textfältet visar vad användaren skrivit in i kalendern för den aktuella tidpunkten eller, om inget skrivits in för tidpunkten, nästa kalenderinmatning som skrivits in där. Fältet remain

visar hur lång tid det har pågått respektive hur lång tid det är kvar. Textfältet syns även till höger om listan(eller under om man skulle vilja) tillsammans med en lista med sätt att kontakta användaren: alla telefonnummer och e-mail adresser.

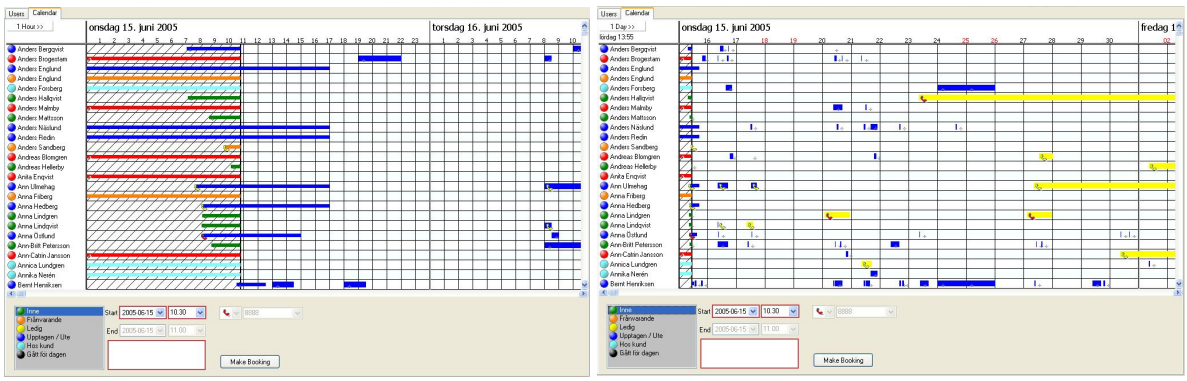
En annan lista visar alla bokningar som gjorts i kalendern.

Calendar-fliken visar en kalender med olikfärgade staplar för olika typer av bokningar. Staplarna har samma färgkodning som bollarna i users-fliken och kalendern kan visas i några olika tidsperspektiv; 1, 2, 4-timmar eller 1 dag, se figur 2.8.

Under stapelkalendern finns några fönster där man antingen kan titta på information om en befintlig bokning eller göra en ny bokning.




*Figur 2.7 Users-fliken i Totalview (telefonnummer övertäckta)*




*Figur 2.8 Kalendervyer f.v. 1-timme, 1-dag*


*Figur 2.9 Tillstånd*

## 2.6 Analys, metodbeskrivning

Denna del beskriver tiden från det första avsparksmötet tills det att vi var redo att börja implementera.

### 2.6.1 Avsparksmötet

I avsparksmötet i januari deltog vi, Kerstin Andersson som handledare från Karlstad Universitetet och Jakob Bjerke som uppdragshandledare från Prevas. Vi fick en muntlig snabbgenomgång av uppdraget som ytterligare utvecklade den vi fått skriftligt via mail. Det visade sig att Prevas för tillfället hade brist på lokaler, men att de snart skulle nyanställa och därmed utöka med ett antal rum. Detta skulle ske någon gång i mars. Eftersom uppdraget till en början var av undersökande karaktär bestämdes det att vi kunde klara oss utan lokaler under den delen och att nyinförskaffandet av lokaler skulle sammanfalla ganska bra med när vi skulle påbörja implementationsfasen. Utöver detta upprättades en tidsplan.

### 2.6.2 Introduktionstur på Prevas

Vi kom överens om att ha en introduktionstur på Prevas på avsparksmötet. Under denna fick vi en utförlig bakgrundsbeskrivning och en lätt översikt av systemet. Vi fick även träffa de personer som vi skulle komma att ha mest kontakt med: Prevas IT-ansvariga, Jakob Bjerke - intranätsdelen, Björn Tuneld - Totalview och Kjell Hendriksson - Exchange. Vi fick se alla tre systemen användas både från kontorsmiljö och via Internet vilket var upplysande och gav mer insikt och förståelse eftersom vi fick prova programmen själva.

Till slut fick vi även en del skriftligt material över de olika systemen att studera: en pärm med information om Totalview, en pärm med dokumentationen till intranätet och boken Microsoft Exchange Server 2003 Administrator's Pocket Consultant (William R. Stanek)[4].

### 2.6.3 Fördjupningsmöte

Cirka två veckor efter introduktionsturen, när vi studerat informationen vi fått, hade vi ett fördjupningsmöte med våra tre kontakter. På detta möte fick vi veta mer om hur arkitekturen på systemet ser ut (se stycke 2.6.3.1). Vi fick en inblick i att Totalviews utveckling kan komma att bli något komplicerad p.g.a. företagspolitik (se stycke 2.6.3.2). Vi fick även en ny kontaktperson i form av Claes Brisby som är den som introducerade och installerade Totalview på Prevas och därför kan sägas vara den mest kunniga personen på detta system inom Prevas. Vi fick även efter detta möte egna inloggningskonton så vi kunde komma åt de webbaserade delarna av systemet, både för att kunna studera det och för att vi skulle kunna kommunicera via det interna mailsystemet och på så sätt snabbare kunna få svar och inte fastna i skräppostfilter och liknande.

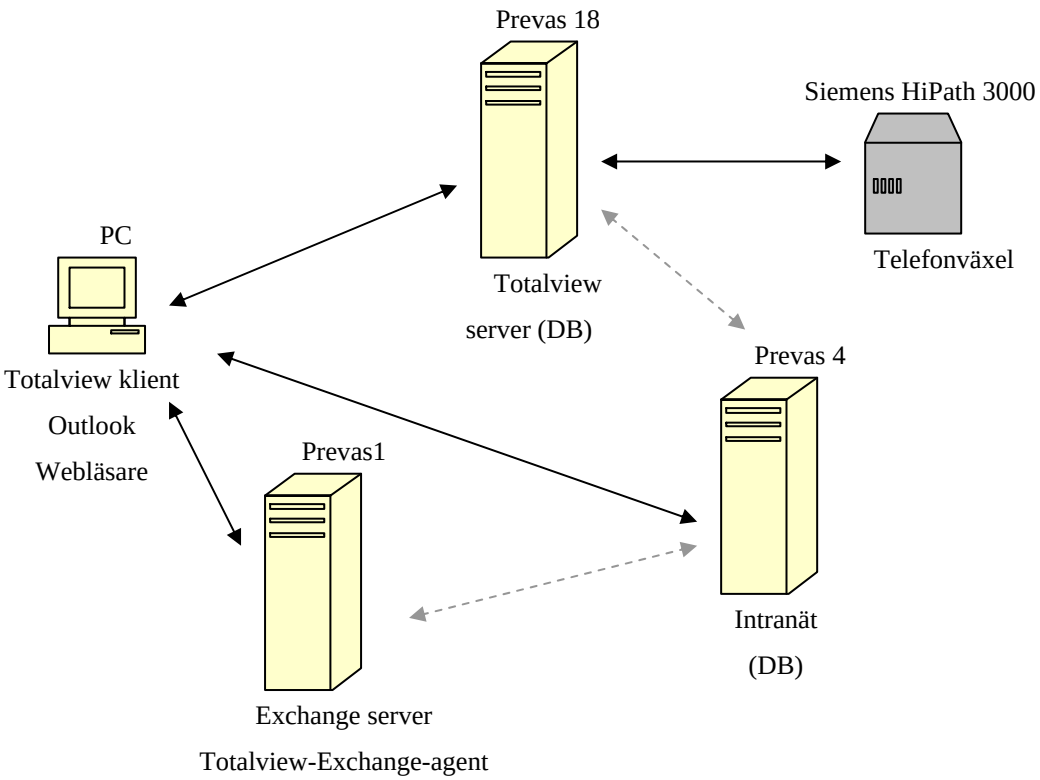
#### 2.6.3.1 Arkitekturen på systemet

Som visas i figur 2.10 arbetar Totalview-klienten, Outlook och intranätsaccess genom webbläsare mot varsin server från en PC. Prevas18 är beteckningen på Totalview-servern. Denna kommunicerar med telefonväxeln, en Siemens HiPath3000, som sköter den praktiska omkopplingen.

Intranätsservern, Prevas4, kan man komma åt från en vanlig webbläsare, genom inloggning, och visar då information från databasen i HTML-format.

Exchange-servern, Prevas1, kan man komma åt både internt på kontoret genom Outlook och utifrån med hjälp av en webbversion av Outlook.

De starka pilarna visar befintliga kommunikationsförbindelser och de svaga pilarna visar det Prevas vill att vi ska implementera.




*Figur 2.10 Arkitektur på system före arbete*

#### 2.6.3.2 Politik

Prevas har köpt en färdig lösning från DITT (Data, IT & Telelösningar) bestående av Totalview programmet och Siemens HiPath 3000 hårdvara. Det är oklart vilka tjänster som ingår i paketet i nuläget. Vi vet ej om DITT tillhandahåller, eller om det ens finns, ett API (Application Programming Interface) för Totalview gratis eller om de säljer ett. DITT har ju inte utvecklat Totalview själva så det kan vara en känslig fråga eftersom Formula har utvecklat programvaran och kanske inte vill erbjuda en sådan lösning (gratis eller alls).

### 2.6.4 Fortsatt undersökande

Materialet vi fick från introduktionsmötet visade sig vara av begränsad användningsnytta, med undantag av intranätsdokumentationen som var egenutvecklad av Prevas. Vi resonerade oss fram till att Totalview-delen skulle vara svårast att hitta information om eftersom den är

utvecklad av ett litet företag och har en mycket begränsad spridning och begränsat användningsområde. Intranätsdelen hade vi god dokumentation om och Exchange är så väldistribuerat att vi bedömde att det skulle bli relativt lätt att hitta det vi ville veta om det. Vi bestämde oss för att försöka oss på det svåraste först och började söka efter information om Totalview på Internet. Det visade sig snart att den enda källan till något så när bra information var Totalviews egen hemsida och även den var bristfällig. Totalview satte under arbetets gång hemsidan till att endast ge tillgång till hjälpdokument till dem med lösenord, vilket Prevas inte fått något på grund av att de ej köpt programvaran direkt från företaget. Som tur var kunde vi ändå komma åt dessa dokument genom en miss i sidkonstruktionen. Vi fann dock att dokumenten inte innehöll något användbart för oss. Vi spenderade sammanlagt ungefär två-tre veckor med att fruktlöst leta efter andra källor, t.ex. färdiga lösningar. Även sökande inom Prevas gav ganska magert resultat om än inte helt resultatlöst. Till slut tvingades vi inse att vi måste gå vidare utan tillfredställande resultat vilket ledde till att vi beslutade oss för att ha kvar Totalview-Exchange kopplingen i det skick den är och låta Prevas sköta utvecklingen av den i framtiden.

Efter detta bakslag bestämde vi oss för att i stället ge oss på den enklaste delen d.v.s. intranätet där vi kunde studera dokumentationen direkt. Efter att ha satt oss in i dokumentationen såg vi till att skaffa ASP-koden för de relevanta kalenderdelarna för närmare granskning och även i viss mån träna inför implementationsdelen genom att förstå och kommentera koden. Vi försökte även få tillgång till källkoden som användes till att skapa databasdrivrutinerna som används i ASP-koden, men det var svårare och av någon anledning kunde vi inte få den. Eftersom vi började känna lite tidspress utvecklade vi ett förslag på lösning parallellt med att vi undersökte möjligheterna med Exchange i stället för att vänta tills allt var klart.

## 2.7 Lösningsförslag

Det förslag till lösning vi framlade löd:

*Förslaget går ut på att Exchange/Outlook agerar som master, dvs. all tidsbokning sker primärt i Outlook. Den befintliga länken mellan Exchange och Totalview används som den är(eventuellt kan den förbättras något genom uppdateringar eller inställningar).*

*Intranätets kalender läser från Exchange och uppdaterar kalendern när man uppdaterar webbsidan. Möjligen blir det bara envägskommunikation mellan Intranätet och Exchange, men tvåvägskommunikation borde kunna lösa sig det med.*

*När det gäller informationsmängden ligger ju Outlook en bit ovanför intranätet vilket är skälet till att vi valt det som Master. En funktion som Outlook har men inte Intranätet är möjligheten att ha parallella bokade tider. Som det är nu kan Intranätet bara visa ett bokningsblock per dag. För att lösa det finns tre alternativ:*

*1. Intranätets kalender visar bara den första bokade tiden*

*2. Man sammanställer alla Outlookbokningar till en text för Intranätet*

*3. Intranätet ändras så att det också kan hantera parallella bokningar*

*Av dom tre punkterna är förstås punkt 3 den bästa, men övergår nästan säkerligen den tid vi har att arbeta på Prevas*

*Vad vi behöver för att genomföra lösningen:*

*Exchange SDK*

*Visual basic .NET (?)*

Responsen på förslaget var sparsam men positiv från nyckelpersonerna. Då ingen hade något att invända eller tillägga bestämde vi oss för att genomföra det.

## 2.8 Undersökningen sammanfattat

Arbetet med undersökning var nu avslutat. Detta arbete gick långsamt framåt på grund av diverse problem. Vi hade svårt att hitta information på så specifika områden som krävdes, många av våra sökningar var fruktlösa och tiden gick. Den potentiella vinsten i att hitta en färdig lösning var så stor att vi valde att satsa någon vecka åt detta. Med facit i hand kanske vi skulle ha begränsat denna sökning till en kortare tidsperiod.

Det lösningsförslag som framlades var enkelt på grund av tidsbrist, men också på grund av att vi vid detta tillfälle inte fått tillgång till källkoden för intranätet.

Vi hade trots dessa problem fått en bra översikt av systemen och vad som måste göras.

# 3 Implementation

I denna del beskrivs först de verktyg och språk vi använt och därefter arbetet med att implementera vårt lösningsförslag.

## 3.1 Språk

**VB**

VB (Visual Basic) är en medlem i BASIC-familjen av programspråk. Det finns en del varianter av VB t.ex. VB script men ren VB är ett fristående objektorienterat programspråk.

**HTML**

HTML (Hyper Text Markup Language) är det språk som blivit standard på webbsidor. Hur en sida visas upp för en besökare styrs av vilka taggar författaren har använt sig av i koden. Författaren styr dock bara textens logik och struktur, läsaren kan själv bestämma, genom inställningar i webbläsaren, hur texten ska presenteras. HTML som standard hanteras av World Wide Web Consortium (W3C)

**ASP**

ASP (Active Server Pages) är ett sätt att kunna baka in avancerad kod i HTML-kod på webbsidor för att på så sätt kunna göra den mer dynamisk. Till skillnad från HTML-kod körs ASP-kod på serversidan och kan därför aldrig ses av användaren. Användaren kan bara se den HTML-kod som är resultatet av ASP-koden. ASP är ett språk Microsoft har utvecklat.

**XML**

XML (eXtensible Markup Language) är en metod för att lagra strukturerad data i en textfil. Dess uppgift är i första hand att förenkla delande av data mellan flera olika system, speciellt system som bara är anslutna till varandra via Internet.

**SQL**

SQL (Structured Query Language) är ett standardiserat språk för att ställa frågor om och modifiera data i en relationsdatabas.

**WebDAV**

WebDAV (Web-based Distributed Authoring and Versioning) är ett protokoll som kan användas som filsystem över Internet. I vårt fall används det för att kommunicera med Exchangedatabasen.

## 3.2 Verktyg

Här beskriver vi de verktyg som vi har använt under implementationen.

**Visual Studio 6.0 (VB, ASP)**

Visual Studio 6.0 är en äldre version av Visual Studio som bland annat innehåller komponenterna Visual Basic 6.0 och Visual InterDev. Visual Basic 6.0 är en utvecklingsmiljö för språket med samma namn. Visual InterDev är ett hemside-utvecklingsverktyg för vanliga HTML-sidor eller script-sidor t.ex. ASP.

**Firefox (XML)**

Firefox är en webläsare som innehåller en kompetent XML-parser och visar XML-dokument med en färglagd trädstruktur.

## 3.3 Förberedelser

Innan den egentliga implementeringen kunde komma igång var vi tvungna att få tillgång till en dator och installera rätt programvara. Vi fick en lånedator som vi fick sitta med i receptionen. Dock var ingen utvecklingsprogramvara installerad på den så det var det första vi fick ta itu med. Vi fick installera delar av programpaketet Visual Studio 6.0, närmare bestämt Visual Basic 6.0 och InterDev 6.0 för utveckling i VB och ASP.

Nästa steg var att se till att koden vi skulle börja ändra var den senaste versionen av intranätet. ASP-koden låg aktivt på intranätservern och var bara att hämta där. Dll-koden var svårare att hitta eftersom endast den kompilerade dll:en var aktiv och koden till den inte använts under en tid. Eftersom dll:en var utvecklat i Visual Studio låg de olika kod-filerna lagrade tillsammans i ett Visual Basic-projekt. Tydligen hade enligt dokumentation ett versionhanteringsprogram vid namn Microsoft SourceSafe används under utvecklingen. Vi installerade detta program och lyckades få fram det projekt som lagrats där. Det visade sig dock att Prevas i Karlstad inte använt sig av SourceSafe i någon större utsträckning och den

enda versionen som fanns lagrad var den ursprungliga. Det fanns dock ett antal nyare versioner lagrade som vanliga filer på intranätservern. Vi fick utföra en sökning på servern och leta fram alla de olika versionerna av dll-koden. Tyvärr var den dator vi fick tillgång till lite seg för sökningen för att hitta alla versioner tog hela natten. Men sedan kunde vi titta när filerna ändrats senast för att få reda på vilken version som var den senaste. För att vara säkra kollade vi också de ändringar som gjorts i koden och jämförde med senast intranätsdokumentationen.

För att vi skulle kunna laborera med koden utan att störa det riktiga intranätet behövde vi någon form av sandlåda som vi kunde experimentera fritt i. Det första försöket att skapa en sådan gick snett. Planen var att vi skulle göra en egen dll-fil som våra ASP-filer kunde anropa och sedan få den registrerad på intranätservern. Filen skulle ha ett namn som var skilt från den redan existerande Prevasintranet.dll. Efter att ha lagt till en enkel funktion i koden och kompilerat den skickade vi den för registrering. Direkt efter registreringen kraschade intranätet helt. Som tur var det en fredag eftermiddag och få personer berördes den lilla tid det tog att omregistrera den gamla filen. Efter att ha tänkt efter lite och diskuterat lite sinsemellan kom de som hade hand om registreringen fram till att det verkar som om intranätet är beroende av att dll-filen ska ha namnet PrevasIntranet.dll.

Veckan efter kom en av IT-ansvariga fram med idén att ge oss en spegel, en identisk kopia, av den aktiva servern att ha som utvecklingsmiljö för att inte störa driften av den riktiga intranätsservern. Tyvärr hade ingen tid att genomföra denna idé direkt då den person som skulle kunna genomföra det var upptagen med annat. Påföljande dag genomfördes halva speglingen. Databasen blev kopierad till den nya testservern, men återigen var nyckelpersonerna för upptagna med annat och först dagen därpå var speglingen slutförd och det först sent på eftermiddagen. Vi kunde nu försöka göra ett nytt test med att registrera vår modifierade dll-fil. Dock tänkte ingen på att dll-filen var tvungen att registreras på servern dvs. registreringskommandot var tvungen att exekvera på den. När vi gjorde vårt registreringsförsök registrerade vi endast dll-filen lokalt på vår dator, dock tog vi bort original dll:en på servern så hemsidan kunde inte fungera eftersom denna dll inte kunde hittas. Vi hade även problem med att servern var tvungen att stoppas innan en ersättning av dll-filen kunde göras eftersom den annars var skrivskyddad.

Vi blev dock lovade att detta skulle lösas till nästa vecka. Vilket det i stort sett hade gjort. Som det blev så används ett program vid namn ”psexec” som utför kommandon på en annan

dator. Alltså kunde, genom ”psexec”, registreringskommandot av dll:en utföras på servern. På detta sätt kunde dll:en nu registreras på servern. Med dll:en registrerad kunde vi nu testa den ändring vi hade gjort. Resultatet var positivt och alltså kunde vi nu påbörja utvecklingen av vår lösning.

## 3.4 WebDAV

Inledningsvis visste vi ingenting om hur vi skulle implementera kopplingen mellan intranätet och Exchange. Vår första uppgift var därför att bestämma oss för vilka metoder, verktyg och språk vi hade att välja på för att genomföra vårt arbete. Efter några försök att hitta ett färdigt exempel på hur man kunde skriva ut en kalender, hämtad från en Exchange-server, började vi söka efter exempel på hur man kan hämta ut kalenderdata enkelt. Efter ett tag kom vi på idén att ladda ner Exchange SDK (Software Developers Kit) i tron att det skulle behövas för att få kontakt med Exchange. SDK:n visade sig dock mest innehålla exempel i olika språk och även ett hjälpbibliotek. Hjälpbiblioteket var dessutom detsamma som finns på Microsofts supportsida för utvecklare [3]. Det innehöll också några hjälpande program bl.a. ett för att utforska en Exchange-databas.

Efter att ha hittat och jämfört en del exempel märkte vi att något av ADO(ActiveX Data Objects), CDO(Collaboration Data Object) eller WebDAV förekom i alla. Vi tyckte, baserat på de exempel vi hittat, att WebDAV verkade mest lättanvänt så vi valde att använda det. Dessutom var det osäkert om CDO överhuvudtaget skulle gå att använda då det verkade kräva Outlook för att fungera och det kanske inte alltid skulle finnas. En stor anledning till beslutet var att vi hittade ett mycket hjälpsamt exempelprogram i SDK-paketet som använde WebDAV och var skrivet i VB-script.

## 3.5 Tester med WebDAV

Med programmet från SDK:n kunde man lista innehållet i en ”mapp” om man skrev in sökvägen. Med en lätt modifiering av programmet fick vi det att lista innehållet i vår testkalender. Modifieringen innebar att vi skickade med ett username och password för att klara av inloggning. Ett litet problem uppstod i att vi inte visste sökvägen till kalendrarna men lyckligtvis använde Prevas defaultinställningarna så det tog inte så lång tid innan vi hittade dem. Sökvägen till kalender är på formen:

https://<servername>/exchange/<user>/calendar

där <servername> i vårt fall är Prevas1 vilket är Exchange-serverns namn.

<user> var däremot svårare att få fram. På Prevas används Exchange-funktionen att ha alias som ersätter den riktiga e-post (SMTP) adressen personen i fråga har. Detta alias är i Prevas fall uppbyggt av två bokstäver från förnamnet och två från efternamnet, totalt fyra bokstäver. Det viktiga med detta alias är att det är unikt och kan på så sätt användas istället för den mycket längre e-postadressen. Aliasen är detsamma som inloggningens användarnamn på intranätet vilket underlättade för oss då vi kom på att man kunde använda aliasen, som är fritt tillgängligt, istället för det fullständiga SMTP-namnet som <user> i sökvägen. Om en anställd på Prevas t.ex. heter Anna Ottosson kan sökvägen till dennes kalender bli:

https://Prevas1/exchange/anot/calendar

Det första exemplet var som sagt gjort i VB script. Då koden för intranät-dll:en är skriven i VB6.0 ville vi hellre ha ett exempel gjort i detta språk för att, när vi var klara, allt skulle vara gjort i samma språk även om VB6.0 och VB script är väldigt likartade.

## 3.6 WebDAV och XML

Nästa mål var att hämta ut rätt data från databasen. Vi kom fram till att WebDAV kommunicerar med Exchange-servern genom att skicka ett XML-dokument med en förfrågan till Exchange-databasen med http. Servern i sin tur svarar med den efterfrågade informationen i ett nytt XML-dokument. Denna kommunikation är implementerad i VB genom ett httpXML-objekt. Vi började leta efter ett exempel gjort i VB6.0 och hittade lite exempelkod på en av Microsofts hemsidor [3]. Med hjälp av det kunde vi börja bygga vårt första egna program och ganska enkelt få tillbaka ett svar och spara det till en XML-fil. Enda missödet var att vi först försökte skriva ut hela svaret med den grafiska komponenten msgbox, som visar text i ett enkelt fönster med en ”OK”-knapp att stänga fönstret med. Men vi märkte snabbt att det var alldeles för mycket text för att få plats där eftersom msgbox är skapad för att visa mindre meddelanden. Vi använde istället Firefox webbläsare för att titta på XML-filen för att man fick en bra bild av strukturen i detta program.

Innan hade vi en ungefärlig uppfattning om vilka funktioner vi skulle överföra från Outlook till intranätet men nu kunde vi se alla funktioner i en trädstruktur med ett namn för varje funktion som i figur 3.1.


*Figur 3.1 Propfind XML-träd*

Vi hämtade ut mötena ett och ett då de var strukturerade som grenar i trädet. Grenarna var enskilda möten och löven i dessa grenar var egenskaperna hos mötena. Vi kunde alltså hämta mötena genom att göra en lista av mötesgrenarna som vi parsade igenom en och en efter position av löven i noderna och kunde på så sätt extrahera information om mötena. För att t.ex. hämta ut subject(rubrik) för ett möte sökte vi på en fast position i en gren på trädet.

## 3.7 Databasfiltrering

Att söka på fasta positioner visade sig vara ett felaktigt sätt att gå tillväga. Vi fick diverse konstiga svar vid testkörning där fält verkade ha försvunnit eller bytt plats. Efter viss kontroll av filutskriften såg vi att egenskaperna *inte* hade fasta platser i trädstrukturen. Tomma fält fick strukturen att se annorlunda ut och eftersom vi sökte på position fick det utskriften att haverera. Vi fick söka noder efter namn istället. Vi hade dittills använt oss av WebDAV metoden PROPFIND som hämtar all information från en viss plats i Exchange-databasen. Vi kom på att en filtrering redan vid begäran av information skulle underlätta för oss senare då vi bara behöver hämta information för en månad i taget. Därför bytte vi till SEARCH som tar emot en SQL-fråga som den ställer till Exchange-databasen. Svaret man får tillbaka ser annorlunda ut där endast de noder man eftersöker återfinns. Detta visas i figur 3.2. Exemplet

på SQL-förfrågan vi hade fungerade inte fullständigt från början och vi var tvungna att ändra på sökvägarna till vissa egenskaper.


*Figur 3.2 XML SEARCH-träd*

När vi hade filtrerat svaret trodde vi att vi hade fått fasta platser på noderna och ändrade tillbaka till att söka efter egenskaper på fasta positioner igen. Anledningen var att det går snabbare att gå direkt till en position och hämta data än att utföra en linjär sökning på namn och sedan hämta data. Detta fungerade bra i testerna vi gjorde på en kalender där alla bokningar gjorts i det webbaserade Outlook. För att testa några funktioner som inte finns på den webbaserade versionen skapade vi några möten med den icke webbaserade versionen. Det fungerade inte som det var tänkt. Det visade sig att det är skillnad på hur de olika versionerna skriver in möten i databasen. Skillnaden är att den webbaserade, när den upptäcker en tom post, skickar in en tom sträng till databasen, medan den andra ignorerar den helt och inte skickar in någonting. Detta leder förstås till att XML-svaret ser olika ut beroende på vilken version som har använts för att skapa ett möte och om det finns några utelämnade poster i mötesinformationen. I vårt fall var problemet att det skapade oordning i de fast numrerade poster vi ville ha när mötena skrivits in med den icke webbaserade versionen och därmed omöjliggjorde en sökning på fasta positioner, igen. Vi fick återgå till att söka på namn istället för position.

Outlook har möjligheten att sätta ett möte som private för att andra inte ska kunna se vad man skrivit in om mötet. I XML-dokumentet man får som svar av en WebDAV-förfrågan finns det en post som kallas Ishidden som vi länge trodde var denna funktion. Vid närmare granskning visade det sig dock finnas en till post kallad sensitivity. En snabb kontroll visade att det var sensitivity som motsvarade private-egenskapen och vi korrigerade därefter.

## 3.8 Skapande av klasser

Vi behövde strukturera egenskaperna vi hämtade ut från varje möte i databasen. Vi skapade för detta ändamål en klass vi kallade cAppointment där den information vi behövde om varje möte kunde lagras. För att kunna hantera tidsegenskaperna hos möten var en konvertering nödvändig. För att inte bry oss om specifika saker som sommartid/vintertid och tidzoner så letade vi upp en modul som sköter denna konvertering. Denna konvertering sker i appointmentklassen genom ett anrop av funktionen InternetTimeToVBLocalTime i modulen modTimeZone. Vi försökte nu få vår kod mer överskådlig och uppdelad samt förbereda den på att integreras i PrevasIntranet-dll:en. Därför skapade vi cExchangeCalendar-klassen.

I huvudmetoden, GetPeriod, i denna klass hade vi förut hårdkodat användarnamn, lösenord och databassökväg. Dessa blev nu fält i klassen och fick inmatningsmetoder som förberedelse för att sedan hämtas in från en fil. GetPeriod har som parametrar två datum som den använder för att filtrera vilka möten som ska hämtas. I vårt fall vill vi visa en hel månad och då behövs första dagen på månaden som en parameter och första dagen på efterföljande månad som andra parameter.

Funktionen returnerar en collection av cAppointment-objekt. Ett collection-objekt är ungefär som en lista som kan innehålla objekt, men inte andra collection-objekt. Objekten i en collection är oftast av samma typ, men detta är inget tvång. Ett collection-objekt har metoder och egenskaper för att visa t.ex. sitt innehåll och sin storlek.

## 3.9 Integrering av dll med ASP

Vi hade nu kommit så pass långt att det var dags att försöka koppla ihop vår kod i dll-filen med ASP-filen. Till att börja med skrev vi en enkel utskriftsfunktion i ASP-filen för att kunna se om vi lyckades, tillsammans med ett anrop till våran kod. Resultatet blev ett ospecificerat säkerhetsfel. Som tur var hade vi redan misstankar om vad felet kunde bero på. När man kopplar upp sig mot webbOutlook måste man nämligen besvara en certifikat-förfrågan, se figur 3.3, första gången i varje session. Detta görs genom att manuellt trycka på en knapp, något som servern naturligtvis skulle ha problem med. I brist på andra uppslag utgick vi från att det var detta som var felet och började titta runt efter lösningar. Vi letade efter ett sätt att ignorera eller acceptera certifikatet. Det vi hittade var en server-orienterad variant av httpxml-klassen kallad serverhttpxml. Eftersom koden skulle köras på en server tyckte vi att det verkade vara en bra ide att ändra, speciellt som det skulle lösa certifikat-problemet och inte innebar några större ändringar av koden i övrigt.


*Figur 3.3 Certifikatförfrågan*

När vi bytte till serverhttpxml fick vi problem med XML-träd traverseringen; nodsökning via namn fungerade inte som det gjorde innan. En av de medföljande funktionerna, selectSingleNode; en funktion som letar upp en nod efter dess tag-namn, lyckades inte längre hitta noder via namn. Vi kunde inte få den att fungera igen trots många försök, så vi gav upp och skrev istället en egen sökmetod som fungerade på det sätt selectSingleNode skulle ha gjort.

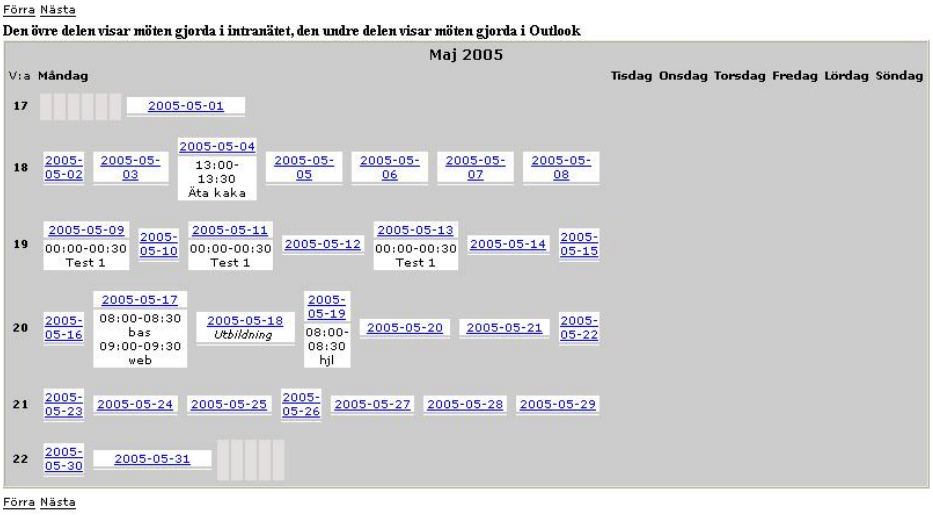
Nästa problem vi stötte på var det som tog i särklass längst tid att lösa. Vi gjorde ett nytt försök att koppla ihop dll- och ASP-filerna och fick felmeddelandet: ”Cannot create active x object” och en kryptisk felkod. Vi fick även veta vilken rad ASP-kod som orsakade felet, men då det var den rad som anropade vår metod för sökning av möten i dll-koden var det inte mycket hjälp till lösning. Eftersom det enda vi hade att gå på var felkoden började vi söka på Internet efter dess betydelse, efter att vi försökt lösa problemet med enkla kodändringar. Tyvärr var träffarna på Internet oense om vad problemet kunde vara. Efter att ha tittat igenom många sidor började det dock framstå som om det oftast berodde på antingen problem med rättigheter eller saknade referenser. Inget av dem gav dock en bra lösning så framsteget var inte stort. Efter att ha kört fast helt och inte kommit på någon lösning själva, fick vi rådet av vår handledare på Prevas att gå igenom koden rad för rad och se vilken rad som misslyckas. Med hjälp av detta tidskrävande förfarande kom vi till slut fram till att det var skapandet av serverhttpxml som var problemet. Detta, kombinerat med att felkoden kunde betyda saknade referenser/filer, fick oss att kolla om det kunde behövas några saknade dll-filer på servern. Det visade sig att på testdatorn (där allt fungerade som det skulle) fanns en nyare version av msxml-dll som inte fanns på servern. Att registrera denna fil (msxml4.dll) var inte så enkelt. Vi upptäckte att den i sin tur behövde en annan fil för att fungera och till slut var det 3 olika dll-filer som behövde finnas på servern. Efter detta fungerade kopplingen och vi kunde komma åt data på Exchange-servern och skriva ut den i en webbläsare.

Vi började nu utveckla ASP-koden för att integrera vår Exchange-information i den befintliga intranätskalendern. Vi var först tvungna att se till att sökvägen i Exchange-databasen var korrekt, det vill säga att rätt kalenderdata hämtades för rätt användares kalender. Sökvägen måste, som vi tidigare nämnt, innehålla användarens alias det vill säga Prevas interna förkortning av deras namn. Denna hämtas från den redan befintliga databasen som innehåller information om de anställda i form av objekt av klassen cEmployee. cEmployee är definierad i PrevasIntranet.dll och innehåller bland annat datan Signature vilken just är aliasen för användaren.

Vi fick fram månadens möten genom att anropa GetPeriod som returnerar en collection innehållande mötena. Vi letade därefter igenom mötena och skriver ut mötena på rätt dag.

## 3.10 ASP

Nu när funktionaliteten var klar var det dags att börja försöka presentera informationen på ett godtagbart sätt. Till att börja med pratade vi med vår handledare igen för att ta reda på mer om hur de ville ha det. Det ursprungliga systemet, figur 2.2, har en cell för varje dag med en länk i datumet till en sida där mer information kan ses eller skrivas in. Prevas önskemål var att dags-cellerna skulle delas upp i två delar. En som såg ut precis som den gamla och en ny del som skulle visa information från Outlook. Eftersom det skulle ta för mycket plats att visa upp all information i cellen för varje möte vore det bra med en liknande länk som för det gamla systemet. Däremot var det lite oklart hur den skulle fungera, och vi funderade på att göra en popup-ruta istället för ett frame-byte. I övrigt fick vi också en del små förslag om bakgrundsfärg och liknande. Det visade sig vara enkelt i teorin att dela upp cellerna i två delar, men svårare i praktiken. Eftersom vi nu var tvungna att arbeta med kod som vi inte skrivit själva var det lite svårt att förstå den, speciellt då den hade en del nästlade iterationer. Efter en del försök, vissa misslyckade som figur 3.4 visar, fick vi det dock rätt och detta visar figur 4.2. Kalendern såg nu ganska bra ut. Vi testade att använda en funktion som gör att ett möte kan bokas in flera dagar i rad och upptäckte att kalendern med vår kod bara visade mötet den första dagen eftersom vi bara tittade på ett mötes starttid när vi avgör när det ska skrivas ut. Vi insåg samtidigt att det skulle bli extra problem om mötet drogs ut över ett månadsskifte. Detta skulle inte gå att klara av med endast den GetPeriod-metod vi hade så vi övervägde att göra en ny metod för detta. Vi hade först en tanke om att modifiera vår sökning av möten så att den endast visade en dag i taget i denna nya metod. Detta visade sig bli svårt då en del funktioner i Outlook gav upphov till specialfall som komplicerade det hela, denna filtrering skulle ha blivit alldeles för avancerad för oss att lyckas med så vi bestämde oss istället att göra en extra filtrering efter den första dvs. den nya metoden filtrerar resultatet av GetPeriod. Vi fick dock ändra lite på GetPeriod-filtreringen också så att den tog med möten som slutade i den månad man filtrerade fram.




*Figur 3.4 Tabell-försök*

Den nya metoden döptes till GetSubPeriod. Denna metod hade vi från början tänkt att den skulle ta en collection och två datum som parametrar och returnera en ny collection som är resultatet av en filtrering av den inskickade collection mellan de två datumparametrarna. Denna resulterande collection innehåller alltså de appointments ur den inskickade collection som har start- och slutdatum inom intervallet mellan de två datumparametrarna. Vi fick dock problem vid integreringen av dll:en och förenklade det hela genom att vi sparade resultatet av GetPeriod-metoden i ett collection-objekt liggande i klassen. På detta sätt behövdes inte en collection som parameter till GetSubPeriod då metoden kunde hämta denna från sin klass.

Vi försökte nu få filtreringen i GetSubPeriod att fungera som vi hade tänkt. Detta visade sig dock vara svårt. Vi gjorde nämligen misstaget att inte tänka på att vi delar upp datum och klockslag vid lagring i vår cAppointment-klass och på så sätt får start- och slutdatum klockslaget 00:00 vid konvertering av dessa lagrade datumsträngar till datatypen Date (som är en exakt tidpunkt). På så sätt blev gränstider på dagar svårdefinierade eftersom 00:00 kan tänkas tillhöra både föregående och efterkommande dag. När vi väl kommit underfund med detta problem kunde vi förbättra filtreringen och på så sätt få de rätta cAppointment-objekten att filtreras fram på rätt dag. För att förenkla metoden ändrade vi den så att den endast tog in en datumparameter(med klockslag 00:00) och filtrerade fram appointments endast för denna dag. Ett problem vi upptäckte var att om man använder sig av alldayevent-egenskapen i Outlook så sätts mötets starttid till 00:00 och sluttid till 00:00 nästa dag. Med vår filtrering såg det då ut som om mötet pågick även dagen efter då sluttiden ingick i påföljande dag. Vi fick ändra vår filtrering så att den ignorerade möten med sluttiden 00:00.

Ett annat problem vi stötte på var när vi ville lägga in de nyligen filtrerade cAppointment i en ny collection. Det gick inte att föra över en cAppointment från den gamla collectionen till den nya på något sätt. Vi fick därför göra en ny metod liknande en copyconstructor, med en cAppointment som parameter, som kopierade parametern till en ny cAppointment värde för värde och returnerade.

Nu var det dags att göra en presentation av mötena på något sätt. Som tidigare nämnts valde vi mellan en pop-up och att byta sida när man tryckte på en länk. Till slut valde vi att byta sida för att vara konsistenta med den befintliga koden. Den nya sidan skulle visa upp alla möten från den aktuella dagen med all information vi hämtat över, d.v.s. subject, location etc. Denna information skrivs ut i tabellform för att få det strukturerat och lättläst, se figur 3.5.


*Figur 3.5 Dag-information*

I stora drag var det hela problemfritt, men vi upptäckte att vi hade hämtat ut fel fält för status från Exchange. Vi hade hämtat ut meetingstatus när det skulle ha varit busystatus. Det var dock inga problem att rätta till.

För att få det att se lite bättre ut ersatte vi vid alla möten med alldayevent den utskrivna start- och sluttiden med en text. Vi tillsatte även en färgkodning på statusfältet.

Nu var det dags att visa upp kalendern igen för att se om den skulle bli godkänd eller om fler ändringar behövdes. Den respons vi fick var nästan uteslutande små kosmetiska ändringar

som ändrat typsnitt på en text och ändrad färgkodning. Två lite större ändringar föreslogs också. Den ena var att ha en annan bakgrundsfärg för Outlook-delen av kalendern. Vi valde samma ljusgula färg som finns i Outlook för detta. Den andra ändringen var en variabel som bestämmer hur många bokningar som visas varje dag i månadsvyn. Denna variabel skulle vara lätt att ändra så att Prevas efter hand själva kan utvärdera vilket värde som passar bäst. Vår lösning kräver visserligen att man går in i ASP-koden och ändrar men vi gjorde variabeln väldigt lättillgänglig och vi tror inte att det kommer att vålla några problem med tanke på kompetensen på Prevas. För att förtydliga att alla bokningar inte alltid visas i månadsvyn har vi olika länktexter till dagsvyn där alla dagens bokningar alltid visas. I figur 3.6 visar vi hur det ser ut när man inte har någon begränsning, dvs. variabeln är satt så högt att den i praktiken aldrig överstigs, på antalet bokningar som visas på en dag. Länken till dagvyn säger då ”mer info”. I figur 3.7 har dock variabeln satts så att antal möten som visas på en dag endast är 3. När fler möten då finns på en dag säger länktexten ”visa alla”.


*Figur 3.6 En dag utan begränsning*


*Figur 3.7 En dag med begränsning*

## 3.11 Implementation avslutad

Vi hade nu kommit till ett läge då vi ansåg att vår implementation var stabil nog att sättas i bruk. Vi lämnade uppgiften att föra över de nya ändringarna till det aktiva intranätet till Prevas personal. Vi glömde dock att lämna goda instruktioner om hur integreringen skulle ske

vilket ledde till att de fick det problem med saknade dll-filer som vi beskriver i stycke 3.9. Detta var dock lätt åtgärdat och kunde lösas via mail. Överlag kan man säga att implementeringsfasen gick i ganska tillfredsställande hastighet.

## 3.12 Respons

Eftersom vårt arbete försenades över sommaren fick vi även tid med att se den respons Prevas anställda gav. Det inledande intrycket från vår kontaktperson var överlag positivt. Dom enda kommentarerna gällde några smärre utseendeändringar som t.ex. ville han hellre ha den detaljerade bokningsinformationen, se figur 3.5, i en pop-up ruta. Ett större test skulle sedan utföras på kontoret i Karlstad under några veckor, för att sedan om det fungerade tillfredsställande sättas i bruk i hela företaget.

Redan innan det större testet hade vår handledare två synpunkter. För det första ville han ha en gräns på 40 synliga tecken på subject i månadskalendern. Det andra var att vi, vid en privat bokning, lämnat alla fält tomma. Han ville hellre att det skulle stå privat i subject-fältet.

Efter några dagar upptäcktes dock en bugg i systemet som gjorde att vissa möten inte syntes fastän dom var publika. Det tog Prevas några dagar innan dom upptäckte vad som kunde vara problemet. Tydligen kunde sökvägen till kalendern i Exchange-databasen vara på olika språk. Detta berodde på vilken Outlook-version som man initierade sitt Outlook-konto i, vilket därmed avgör vilket språk sökvägen får. Även om man ändrar språkinställning på Outlook eller använder en Outlook-klient med ett annat språk förblir sökvägen till kalendern i Exchange densamma. I vårt fall initierade vi våra konton med ett engelsk Outlook vilket fick våra sökvägar att bli engelska, alltså blev den:

https://Prevas1/exchange/<alias>/calendar/

Om vi hade initierat med ett svenskt Outlook hade sökvägen blivit:

https://Prevas1/exchange/<alias>/kalender/

Med tanke på att Prevas även har arbetare i flera länder i världen så används även Outlook med andra språk än svenska och engelska.

Prevas ville väldigt gärna ha detta problem löst och gav oss tre alternativa lösningar:

1. Koda så att ni alltid kan ta reda på vad kalendern heter för varje given person. Till 100%.

2. Koda så att den först provar den ena "Calendar" och om den inte får nåt svar, provar med "Kalender", om den inte får svar, returnerar ett sjysst felmeddelande.

3. Se till att alla byter till engelsk Outlook...

Prevas föredrog ju helst alternativ 1, men vi trodde själva att vår lösning skulle vara mer likt alternativ 2, då vi inte hade en aning om hur alternativ 1 skulle implementeras. Vi försökte dock att ge alternativ 1 ett försök. Problemet var att få reda på kalendermappens namn när allt man visste var användarens alias.

Vi upptäckte ett sätt att söka igenom en användares alla ”mappar” efter bokningar. På detta sätt söktes även kalendermappen igenom oavsett vilken sökväg den hade. Problemet var att sökningen var mycket tids- och resurskrävande så vi var rädda att det skulle kräva alltför mycket av servern om det sattes i bruk i ett stort nätverk. Vi letade därför vidare efter en annan lösning.

Under tiden vi tänkte kom vi på ett par olika sätt att implementera alternativ 2, det bästa skulle ha kunnat avgöra vilken mapp som var kalendermappen ganska enkelt. Det hade dock krävts att det fanns en lista över vad kalender heter på olika språk vilket skulle ha varit ganska klumpigt.

Vi inriktade oss på att hitta ett sätt att ta reda på vad kalendern hade för sökväg istället. Vi fann ingen hjälp att få från Internet men när vi gick igenom ett xml-dokument som var ett svar på en PROPFIND-förfrågan, se figur 3.1, mot Exchange-databasen. I detta dokument fann vi att även kalendermappens egenskaper visades. Vi såg att kalendermappen hade ett contentclass-fält som innehöll bevis på att detta var en kalendermapp, bland dom andra egenskaperna fanns även sökvägen till denna. Nu kunde vi göra använda WebDAV:s SEARCH-metod för att filtrera fram sökvägen för kalendermappen ur användarens Exchange-egenskaper. Detta innebar att det var nödvändigt med två förfrågningar mot databasen, en för att få reda på sökvägen till kalendern och en för att hämta innehållet. Vi skrev en ny metod för att ta reda på sökvägen som fick namnet PathFinder. Detta löste problemet med sökvägen.

Vad vi inte räknat med var ett nytt problem som visade sig göra vissa delar av intranätkalendern obrukbar. Som vi nämnde i stycke 2.3.1 så finns det lokaler med egna kalendrar som används för bokning. Dessa lokaler har ingen alias och ger därför ingen korrekt sökväg till Exchange-databasen. Vi hade glömt att göra en felkontroll när ingen alias fanns. Så när man försökte använda kalendern för en lokal så utförde vår kod en ogiltig åtkomst i

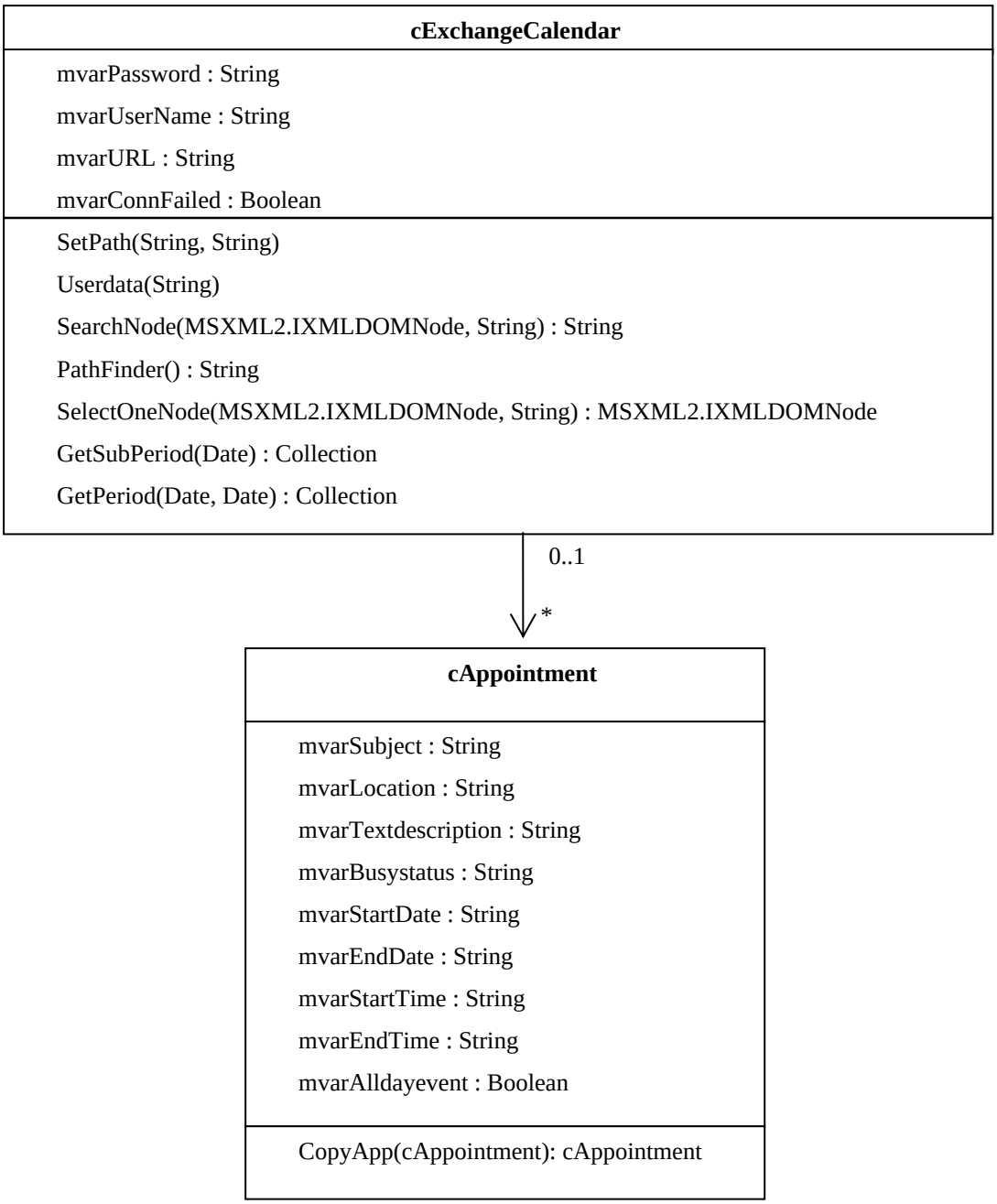
databasen och ingen kalender visades. Detta var dock inget större problem att rätta till. Vi gjorde helt enkelt en felkontroll som inte tillät tomma alias-fält. I fall med saknad alias gav vår kod helt enkelt tillbaka ett tomt resultat vilket ledde till att ingen Exchange-kalender skrivs ut på intranätet.

# 4 Den färdiga kalendern

Det här kapitlet beskriver hur den färdiga lösningen ser ut och fungerar samt hur väl den stämmer överens med den föreslagna lösningen.

## 4.1 Konstruktion

Det här är ett klassdiagram, figur 4.1, över de klasser vi har skapat och beskrivningar av dess metoder. cExchangeCalendar är en klass som hämtar och behandlar information från kalendern i en Exchange-server. cAppointment-klassen är till för att kunna hantera datan i en Outlook-bokning.




*Figur 4.1 Klassdiagram*

**Setpath()** sätter sökvägen till Exchange-databasens kalenderdata med parametrarna servername och alias. Servername är namnet på Exchange-servern, eventuellt kan servername vara en ip-adress om inte Exchange-servern är registrerad i DNS-servern. Alias är ett unikt användarnamn för varje anställd och parametern anger vilken anställds kalender som ska anropas

**Userdata()**läser in användarnamn och lösenord för åtkomst till Exchange-databasen från en fil. I filen finns användarnamn på första raden och lösenord på andra. Parametern Filename är sökvägen till filen

**SearchNode()** använder sig av SelectOneNode() på det XML-träd som är parametern mynode. Parametern Searchvalue är namnet på den nod som eftersöks och denna vidarebefordras till anropet av SelectOneNode(). Resultatet från SelectOneNode() kontrolleras sedan. Om en tom nod är resultatet av SelectOneNode()-anropet returneras en tom sträng, annars returneras innehållet i resultatnoden.

**PathFinder()** kopplar upp mot Exchange-servern med en SQL-förfrågan om sökvägen till kalendermappen och returnerar denna sökväg.

**SelectOneNode()** tar emot ett XML-träd i parametern parent. Metoden traverserar därefter barn-nivån linjärt i en sökning efter en nod med namnet samma som parametern mynodename. Om en nod med samma namn hittas returneras denna, annars returneras en tom nod.

**GetSubPeriod()** utför en filtrering på den kollektion av appointments som är resultatet av GetPeriod(). Parametern SubStart är den dag som bokningarna som ska visas ska pågå under. Resultatet är en ny kollektion som är en delmängd av GetPeriod-kollektionen

**GetPeriod()** kopplar upp mot Exchange-servern och skickar en SQL-fråga och tar emot det datumfiltrerade svaret. Detta svar är bokningar som pågår mellan parametrarna Startdate och Enddate och bokningarna lagras i en kollektion som global data.

**CopyApp()** skapar en kopia av en bokning. Bokningen som kopieras är den som ingår i parametern inputapp

## 4.2 Exekvering i praktiken

För att underlätta förståelsen för hur exekveringen sker och kalenderdata hämtas och visas ger vi här ett exempel på hur just exekveringen kan ske. Det är främst vår egen nyskrivna funktionalitet vi koncentrerar oss på, även om vi nämner den gamla.

Det första som händer är att EmpCalendar.asp begärs genom att användaren trycker på statusikonen på sidan med listan av medarbetare (se figur 1.1). ASP-sidan måste genomgå en exekvering på serversidan för att producera HTML-kod som visar kalendern. ASP-sidan tar reda på dagens datum så den kan skriva ut rätt månad. Sidan kan även ta in datum som parameter och kan på så sätt visa en specificerad månad annat än den nuvarande. Detta används för att bläddra framåt och bakåt, månadsvis, i kalendern.

Därefter skapas ett cEmployee-objekt som med hjälp av parametern sEmpid (användar-id), som skickats med till ASP-sidan, hämtar information om den anställde från en SQL-databas. Ett cEmpCalendar-objekt skapas också och används för att ta fram månadens bokningar för intranätsdelen av kalendern från samma SQL-databas.

Ett cExchangeCalendar-objekt skapas och dess metod UserData används för att läsa in login-data till Exchange-datbasen från en fil. Objektet sätter sedan sökvägen till Exchange-databasen genom SetPath-metoden.

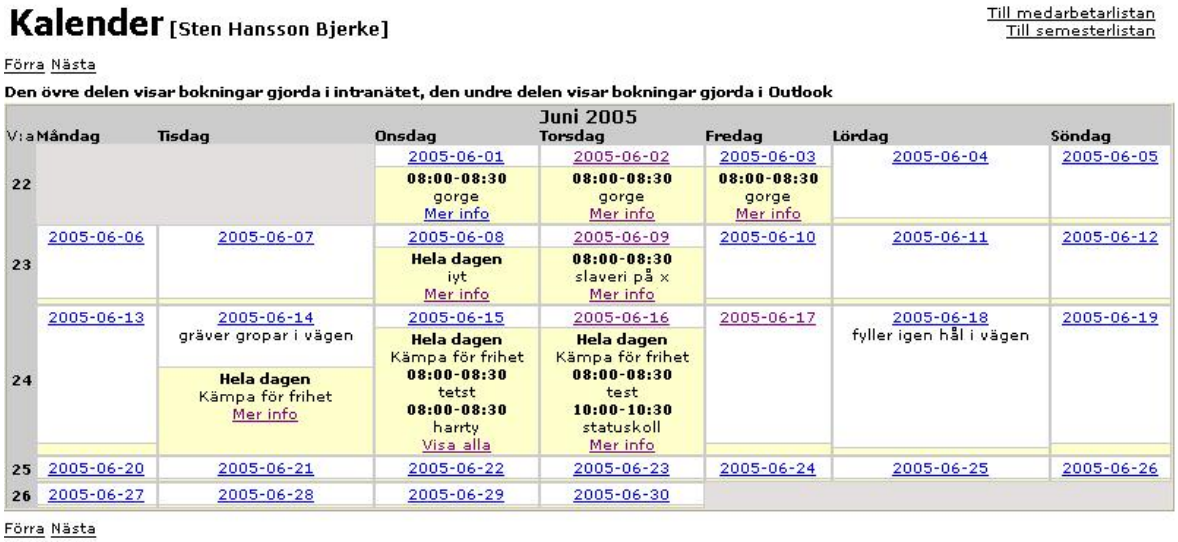
Kalenderdata från Exchange-databasen måste nu hämtas och detta sker genom cExchangeCalendar-objektets metod GetPeriod.

GetPeriod tar in två datum som parametrar, ett startdatum och ett slutdatum. I detta fall är parametrarna satta så att perioden blir en månad. Metoden börjar med att skapa en WebDAV-förfrågan innehållande en SQL-fråga. Denna SQL-frågan begär den information om varje bokning som vi behöver visa upp. SQL-frågan filtrerar även fram bokningar som pågår mellan de två datumparametrarna. Sedan sker en uppkoppling mot Exchange-databasen och WebDAV-förfrågan skickas som ett XML-dokument. Svaret tas också emot som ett XML-dokument där bokningarna ligger strukturerade. Informationen om varje bokning tas fram ur svaret och paketeras i cAppointment-objekt som sedan sparas en efter en i ett collection-objekt. Denna collection returneras men sparas även i klassvariabeln mvarFilteredAppointments som senare används i metoden GetSubPeriod.

När nu all information är hämtad ska den skrivas ut. Detta skall göras på ett strukturerat och kalenderaktigt sätt. Informationen skrivs därför ut i HTML-tabeller med rätt information i rätt tabellcell, där varje tabellcell representerar en dag. För att Exchange-infomationen ska skrivas ut på rätt dag används metoden GetSubPeriod för att, från de för månaden framfiltrerade bokningarna, filtrera fram just en dag.

GetSubPeriod har ett datum som parameter och filtrerar fram alla cAppointments (bokningar) som pågår denna dag ur mvarFilteredAppointments. Dessa returneras som ett collection-objekt.

Intranätkalendern skrivs ut parallellt med Exchange-kalendern i delade celler som kan ses i figur 4.2 och figur 4.3.



**Kalender** [Sten Hansson Bjerke]

Till medarbetarlistan
Till semesterlistan

Förra Nästa

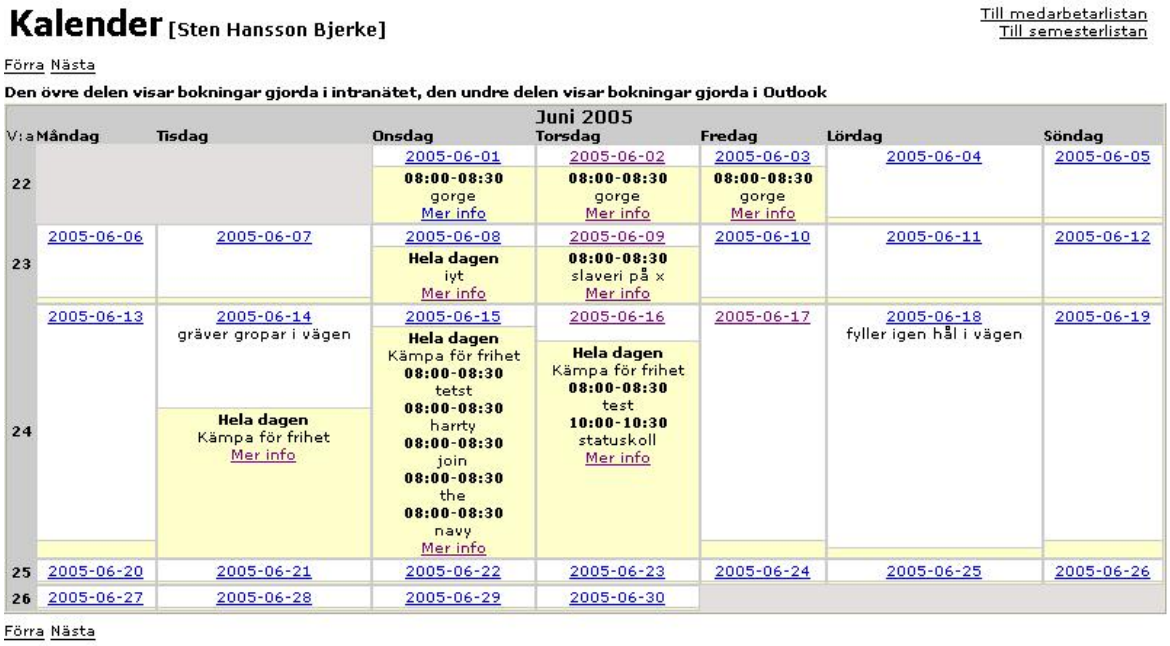
**Den övre delen visar bokningar gjorda i intranätet, den undre delen visar bokningar gjorda i Outlook**

Juni 2005

| V.a | Måndag | Tisdag | Onsdag | Torsdag | Fredag | Lördag | Söndag |
|---|---|---|---|---|---|---|---|
| 22 | | | 2005-06-01<br>**08:00-08:30**<br>gorge<br>Mer info | 2005-06-02<br>**08:00-08:30**<br>gorge<br>Mer info | 2005-06-03<br>**08:00-08:30**<br>gorge<br>Mer info | 2005-06-04 | 2005-06-05 |
| 23 | 2005-06-06 | 2005-06-07 | 2005-06-08<br>**Hela dagen**<br>iyt<br>Mer info | 2005-06-09<br>**08:00-08:30**<br>slaveri på x<br>Mer info | 2005-06-10 | 2005-06-11 | 2005-06-12 |
| 24 | 2005-06-13 | 2005-06-14<br>gräver gropar i vägen<br>**Hela dagen**<br>Kämpa för frihet<br>Mer info | 2005-06-15<br>**Hela dagen**<br>Kämpa för frihet<br>**08:00-08:30**<br>tetst<br>**08:00-08:30**<br>harrty<br>Visa alla | 2005-06-16<br>**Hela dagen**<br>Kämpa för frihet<br>**08:00-08:30**<br>test<br>**10:00-10:30**<br>statuskoll<br>Mer info | 2005-06-17 | 2005-06-18<br>fyller igen hål i vägen | 2005-06-19 |
| 25 | 2005-06-20 | 2005-06-21 | 2005-06-22 | 2005-06-23 | 2005-06-24 | 2005-06-25 | 2005-06-26 |
| 26 | 2005-06-27 | 2005-06-28 | 2005-06-29 | 2005-06-30 | | | |

Förra Nästa

*Figur 4.2 Månadsvy visar max 3 möten/dag*



**Kalender** [Sten Hansson Bjerke]

Till medarbetarlistan
Till semesterlistan

Förra Nästa

**Den övre delen visar bokningar gjorda i intranätet, den undre delen visar bokningar gjorda i Outlook**

**Juni 2005**

| V: a | Måndag | Tisdag | Onsdag | Torsdag | Fredag | Lördag | Söndag |
|---|---|---|---|---|---|---|---|
| 22 | | | 2005-06-01<br>**08:00-08:30**<br>gorge<br>Mer info | 2005-06-02<br>**08:00-08:30**<br>gorge<br>Mer info | 2005-06-03<br>**08:00-08:30**<br>gorge<br>Mer info | 2005-06-04 | 2005-06-05 |
| 23 | 2005-06-06 | 2005-06-07 | 2005-06-08<br>**Hela dagen**<br>iyt<br>Mer info | 2005-06-09<br>**08:00-08:30**<br>slaveri på x<br>Mer info | 2005-06-10 | 2005-06-11 | 2005-06-12 |
| 24 | 2005-06-13 | 2005-06-14<br>gräver gropar i vägen<br>**Hela dagen**<br>Kämpa för frihet<br>Mer info | 2005-06-15<br>**Hela dagen**<br>Kämpa för frihet<br>**08:00-08:30**<br>tetst<br>**08:00-08:30**<br>harrty<br>**08:00-08:30**<br>join<br>**08:00-08:30**<br>the<br>**08:00-08:30**<br>navy<br>Mer info | 2005-06-16<br>**Hela dagen**<br>Kämpa för frihet<br>**08:00-08:30**<br>test<br>**10:00-10:30**<br>statuskoll<br>Mer info | 2005-06-17 | 2005-06-18<br>fyller igen hål i vägen | 2005-06-19 |
| 25 | 2005-06-20 | 2005-06-21 | 2005-06-22 | 2005-06-23 | 2005-06-24 | 2005-06-25 | 2005-06-26 |
| 26 | 2005-06-27 | 2005-06-28 | 2005-06-29 | 2005-06-30 | | | |

Förra Nästa

*Figur 4.3 Månadsvy visar alla möten*

Om man sedan väljer att titta närmare på en dag genom att klicka på ”mer”-länken begärs EmpCalendarExDay.asp. Denna sida får med parametrarna date (den valda dagens datum) och empid (den anställdes id) och det första som händer är att den anställdes data hämtas med hjälp av dess id. Därefter skapas ett cExchangeCalendar-objekt som hämtar inloggningsdata genom Userdata-metoden på samma sätt som Empcalendar.asp. GetPeriod-metoden anropas med startdatum en dag före den valda dagen och med slutdatum två dagar efter den valda dagen. GetSubPeriod tar sedan fram endast de bokningarna som pågår under den valda dagen. Dessa bokningar skrivs ut i tabellform, en och en, efter varandra och med utförlig information.

## 4.3 Lösning kontra lösningsförslag

Om man jämför den färdiga lösningen med det lösningsförslag vi lämnade in i slutet av undersökningsfasen (se kap 2.7) kan man göra några iakttagelser.

Till stor del stämmer lösningen med lösningsförslaget t.ex. blev Outlook/Exchange master som planerat. Kopplingen mellan Exchange och intranät blev dock bara envägskommunikation, detta beroende på att vi tänkte oss envägskommunikation först för att sedan lägga till kommunikation åt andra hållet efteråt. Det visade sig dock att det var ett större

arbete än vi hade tänkt oss och att det skulle ta mycket längre tid att genomföra än vi hade till förfogande.

Av de tre möjliga lösningar till kalenderutskrift vi visade blev det alternativ 2(visa en dags bokningar som en text) som vi genomförde. Alternativ 1(att bara visa dagens första bokning) var ganska uppenbart undermåligt och alternativ 3(ändra hela intranätet att hantera parallella bokningar) var nästan lika uteslutet med tanke på arbetsbörda. En liten justering blev att intranätets gamla funktionalitet fick vara kvar och existerar parallellt med den nya Exchange-kopplingen. Detta för att användare som är vana vid det gamla systemet fortfarande ska kunna använda det. T.ex. anses långtidsbokningar i Outlook något krångliga av många och därför används intranätets kalender för långa ledigheter. Beroende på hur användaren använt systemet innan ändringarna gjordes så kan omställningsperioden ta olika lång tid. Vi tror dock att dom flesta anpassar sig snabbt då vi ej tagit bort någon funktionalitet utan istället endast lagt till ny.

# 5 Slutsatser

Det här kapitlet innehåller de slutsatser vi har dragit av arbetet och möjliga förbättringsområden.

## 5.1 Lärdomar

Vårt arbete har resulterat i några nyttiga lärdomar som man säkert kan få användning av vid liknande arbeten. Den viktigaste är att saker alltid tar längre tid än vad man planerat om man är beroende av att andra ska bekräfta ens beslut eller tillhandahålla material för att genomföra beslutet. Det var ibland svårt att få respons från nyckelpersonerna beroende på att dom var upptagna med annat. Det extremaste exemplet var att vi inte fick en arbetsplats förrän 2-3 veckor efter vi avslutat utredningsdelen och skulle starta med implementationsdelen, men det var också många tillfällen då vi fick vänta en eftermiddag på beslut.

Som vi utförde arbetet blev undersökningsdelen och implementationsdelen vitt åtskilda där det ena avslutades helt innan det andra påbörjades. Så här i efterhand tror vi att det kan ha fungerat bättre med en viss överlappning mellan de olika arbetsdelarna. Vi hade då kanske kunnat sätta oss in i systemet i ett tidigare skede och fått bekräftat att vi hade tagit dom rätta

besluten då vi avslutade undersökningsdelen. Det var under detta arbete dock svårt då vi, som vi tidigare nämnt, inte fick någon arbetsplats direkt.

Ett mer utvecklat lösningsförslag hade kanske förenklat och förkortat implementationsarbetet, men detta hade kanske förlängt undersökningsdelen lika mycket som det förkortat implementationsdelen.

Vi fastnade kanske för länge i vår undersökningsdel då vi letade efter information om TotalView och hur man kunde manipulera detta program. Utvecklarna vill förstås skydda sin kod och vi lärde oss att det inte är någon idé att söka för länge efter sådan information om en kommersiell programvara.

## 5.2 Förbättringsområden

Vissa beslut vi tog under arbetets gång gjordes för att få klart arbetet i tid eller för att det skulle vara för svårt med en bättre lösning. Sådana beslut skulle kunna justeras i efterhand av någon med mer tid eller kunskap.

Ett exempel på detta är dubbelkommunikation. Vår lösning blev ju till slut att intranätet hämtar information från Exchange-databasen och visar upp det i sin kalender men inte tvärtom. Det är något som skulle kunna läggas till senare. En koppling mellan intranätet och Totalview utan att gå genom Exchange är också en utvecklingsmöjlighet men skulle förmodligen kräva ett närmare samarbete med Totalviews utvecklare för att kunna förverkligas.

När intranätet skapades användes Visual Basic 6.0 vilket i nuläget är ett äldre språk. Detta har ersatts av Visual Basic .NET och vi märkte hela tiden under utvecklingen att det var lättare att hitta exempel för utveckling i det nyare språket.

Andra beslut vi tog var val mellan alternativa lösningar. Exempel på det är till exempel att vi ett tag övervägde att låta Exchange-servern mata in information i intranätsdatabasen istället för att låta intranätet hämta information från Exchange-servern. Detta skulle dock kräva att intranätsdatabasen anpassas till att kunna hantera bokningsdata från Exchange. Man skulle också få problem med att hålla data synkroniserad, en bokning kan ju tas bort eller ändras och detta skulle ge upphov till flera operationer som måste utföras på intranätsdatabasen.

Eftersom vi nu hämtar all våra data från, den alltid uppdaterade, Exchange-databasen direkt får vi inte problem med inaktuell data.

Vi stod mellan valet att visa dagsinformationen i kalendern på en ny sida eller i ett pop-up fönster. Vi valde att visa informationen på en ny sida för att behålla den gamla designen på sidan då intranätskalendern fungerade på detta sätt. I efterhand visade sig dock att Prevas tyckte annorlunda och implementerade en pop-up version själva.

# Referenser


[1] *http://www.microsoft.com/Exchange/evaluation/whatis.mspx Hämtad: 25 februari, 2005*

*[2] http://office.microsoft.com/sv-se/assistance/HA010714981053.ASPx Hämtad: 26 februari, 2005*

*[3] http://msdn.microsoft.com/library/default.asp Hämtad kontinuerligt perioden April-Maj 2005*

*[4] Microsoft Exchange Server 2003 - Administrator's Pocket Consultant: Administrator's Pocket Consultant, William Stanek, ISBN: 0735619786 , Microsoft Press U.S.*