CHALMERS

Webbaserat resurshanteringssystem

Effektiv allokering av konsulter

Examensarbete inom Högskoleingenjörsprogrammet i datateknik

JOHN FORS
ALEXANDER PERSSON

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

Webbaserat resurshanteringssystem
Effektiv allokering av konsulter

John Fors, Alexander Persson

Institutionen för data- och informationsteknik
Chalmers tekniska högskola
412 96 Göteborg
Tel: 031-772 1000
Fax: 031-772 3663

Institutionen för data- och informationsteknik
Göteborg, 2014

Sammanfattning

Företaget CGIs Göteborgskontor använder vid tidpunkten för skrivandet av denna rapport Microsoft Excel för att hantera sitt planeringsarbete gällande allokering av konsulter. Målet med detta projekt var att utveckla en webbapplikation med tillhörande databas för att ett stort antal användare skall kunna hantera planeringsarbetet på ett mer tidseffektivt sätt. Denna rapport beskriver implementeringen av ett sådant system. Funktionaliteten för systemet delades upp i olika iterationer, dessa bröts sedan ner i delmoment som implementerades inkrementellt. Det konstruerade systemet använder Microsofts ASP.NET Web Forms, samt Microsoft SQL Server 2008 för den tillhörande databasen. Resultatet blev en webbapplikation och databas som gör planeringsarbetet enklare jämfört med tidigare använt system, med utökad funktionalitet som hanterar frånvaro.

Nyckelord: ASP.NET, SQL Server, Webbutveckling

Förord

Denna rapport sammanställdes för att dokumentera vårt examensarbete vid
Chalmers Tekniska Högskola som utfördes åt företaget CGI.
Vi vill tacka Rolf Andersson på CGI för det givande uppdraget, samt för engagemang och vägledning genom hela projektet.
Vi vill även tacka Joachim von Hacht för hjälp med utformningen av rapporten.

Ordlista

Allokering	En konsult (resurs) involverad i ett projekt, som utför en viss typ av arbete (roll).
API	Application Programming Interface, specifikation av metodanrop som beskriver en programvaras funktion.
Beläggning	En konsults arbetsbörda.
Bootstrap	Ramverk för front-end webbutveckling, visuell design (CSS, JavaScript).
Namespace	Används för att organisera klasser i olika grupper.
ODBC	Open DataBase Connectivity. API utvecklat av Microsoft för att uppnå plattformsoberoende kommunikation mellan applikation och databas.
ORM	Object-Relational Mapping. Teknik för att konvertera relationsdata från en databas till minnesobjekt.
Ramverk	Färdigskriven kod som kan användas för att underlätta implementering av funktionalitet.
Webbapplikation	En applikation vars kod kan exekveras hos både klient och webbserver och som användare kommer åt genom en webbläsare.
Front-end	Front-end är ett begrepp som betecknar användargränssnittet och koden som körs på klientsidan.
Back-end	Back-end är ett begrepp som betecknar kod som exekveras på en server.
Unit-test	Test för att verifiera funktionalitet hos modul.

Lista över figurer, tabeller och kodavsnitt

Innehåll

1. Inledning

1.1 Bakgrund

CGI Group är ett internationellt IT-bolag som förmedlar konsulttjänster, affärssystem och systemintegration. Verksamheten för Göteborgskontoret är främst försäljning, kundanpassning och support av affärssystem.

För att kunna planera in uppdragen för företagets konsulter och få en överblick över konsulternas beläggningsgrad använder CGIs Göteborgskontor idag ett Excel-dokument. Detta Excel-dokument visualiserar resursplanering och beläggningsgrad för konsulterna.

Denna metod har flera brister och försvårar arbetet för administratörerna. Några av problemen som CGI har idag med det befintliga systemet är att det inte går att arbeta parallellt med resursplaneringen. Det finns heller ingen funktionalitet för att få in önskemål om allokeringsförslag från projektledare. Nuvarande system kan inte på ett bra sätt förmedla den färdiga planeringen till konsulterna. Det är även svårt att planera in semester- och helgdagar eller annan frånvaro och ha detta i åtanke när konsulternas beläggningsgrad ska granskas.

Företaget har tidigare drivit ett projekt för att själva ta fram ett webbaserat system. Detta gjordes för att effektivisera allokeringsarbetet och låta processen och dess resultat bli tillgänglig för flera personer. En webbapplikation visade sig vara lämplig för ändamålet. Detta då den enkelt skulle kunna göras tillgänglig genom företagets intranät samt att den nödvändiga kompetensen för att utveckla ett sådant system redan fanns bland de anställda. Ett utvecklingsarbete påbörjades men avbröts dock efter en kort period på grund av en icke hållbar tidsåtgång och viss problematik med att modellera domänen.

1.2 Syfte

Syftet med projektet är att förenkla planeringsarbetet för CGI genom att implementera ett system för resurshantering.

1.3 Mål

Målet är att utveckla en webbapplikation för resurshantering.

1.4 Avgränsningar

På grund av projektets ansenliga storlek kommer arbetet inte att omfatta kommunikationen av den färdiga resursplaneringen till konsulterna. Detta får ses som vidareutveckling.

Säkerhetsaspekten vad gällande kommunikation med webbapplikationen behandlas inte. Detta då applikationen bara kommer att användas på företagets interna nätverk.

2. Metod

Utvecklingen av webbapplikationen kommer att ske med en traditionell objektorienterad, iterativ och inkrementell process för mjukvaruutveckling. Under hela projektets gång kommer en kontinuerlig dialog att hållas med CGI. Ett möte med företagets representant kommer att bokas in, där en kravspecifikation sammanställs och önskemål om design och gränssnitt diskuteras.

Systemet kommer att brytas ned i användarscenarion som implementeras inkrementellt. En fördelning av de olika deluppgifterna och dess turordning kan ses i Gantt-schemat i appendix A.

Projektet kan brytas ner i följande tre iterationer:

- Iteration 1 – Allokeringshantering och planering. Systemet har den nödvändiga funktionaliteten för att administratörer skall kunna utföra arbetet.
- Iteration 2 – Projektledare skall kunna lägga in preliminära allokeringar i systemet. Administratörer kan justera och godkänna dessa.
- Iteration 3 – Planering skall kunna kommuniceras till konsulter med personligt utformat schema tillgängligt genom webbapplikationen. Eventuellt även via e-post och kalenderuppdateringar.

Efter varje iteration är webbapplikationen körbar och klar för vidare utveckling. Fördelen med detta är att projektet kan skalas ner och anpassas till examensarbetets tidsram. Enligt avgränsningarna för examensarbetet behandlas inte iteration 3.

3. Teknisk bakgrund

3.1 HTML, XHTML

HTML (HyperText Markup Language) är språket för att definiera strukturen för webbsidor. XHTML (Extensible HyperText Markup Language) är en variant på HTML som har striktare syntax och är ett XML-språk [1].

3.2 CSS

CSS (Cascading Style Sheets) är ett språk som beskriver formatering och layout av ett dokument. CSS kan appliceras på både HTML, XHTML och andra XML-baserade dokumenttyper [1].

3.3 .NET

.NET är ett ramverk utvecklat av Microsoft. Ramverket ger en bred systemplattform med ett stort objektorienterat klassbibliotek med klasser för vanliga programtillämpningar. Kärnan i .NET-ramverket är den virtuella maskin, ”Common Language Runtime” (CLR) som exekverar .NET-program skrivna i C++, C# eller Visual Basic [2] [3].

3.4 ASP.NET

ASP (Active Server Pages), ASP.NET är en del av ramverket .NET. ASP.NET är en utvecklingsplattform för webbapplikationer. En ASP.NET-applikation kan nyttja klassbiblioteket i .NET-ramverket och dra nytta av CLR-maskinen [4].

3.4.1 ASP.NET Web Forms

ASP.NET Web Forms är en av de tillgängliga programmeringsmodellerna och utgör grunden i ASP.NET-ramverket. Web Forms kan programmeras med en kombination av HTML, C# (eller VB) och skriptspråk som tolkas av webbläsaren. När en aspx-sida (Web Form) efterfrågas så kompileras och körs koden på servern och den genererade sidan skickas till klienten [5].

ASP.NET Web Forms använder aspx-filer tillsammans med tillhörande cs-filer (C#) för att generera ett HTML-svar på ett anrop. Filerna (aspx) använder utöver HTML-element även ASP.NET specifika element som möjliggör definiering av programmerbara element kopplade till HTML [6].

3.4.2 ASP.NET MVC

ASP.NET MVC är en alternativ programmeringsmodell som är ASP.NETs implementering av den generella MVC (Model View Controller) modellen. Till skillnad från Web Forms separerar denna modell applikationens struktur och grafiska representation [7].

3.4.3 Livscykeln av ett anrop till ASP.NET.

Servern hanterar varje anrop genom en instans av Global-klassen (System.Web.HttpApplication). Det kommer alltså att skapas många instanser av Global-klassen. Dessa kan återanvändas och hantera flera anrop under sin livscykel. Detta ställer krav på att klassen är skriven så att den är trådsäker. En instans av Global-klassen svarar på händelser för det specifika anropet genom att skapa objekt av särskilda klasser (moduler) som hanteras av applikationen. Varje anrop kommer i slutändan att resultera i att servern skickar ett svar med de renderade komponenterna tillbaka till klienten. Webbapplikationen kommer på så sätt att hantera anrop tills dess att applikationen stoppas [6]. I figur 3.1 illustreras händelseförloppet vid ett anrop till servern.

Figur 3.1 *– Livscykeln av ett anrop till ASP.NET.*

3.5 PHP

PHP: Hypertext Preprocessor är ett öppet (open source) skriptspråk. Webbsidor skrivna i PHP kan innehålla HTML-kod tillsammans PHP-kod. PHP-koden exekveras då på server-sidan innan den tillsammans med HTML-koden skickas till klienten. Detta möjliggör skapandet av dynamiska webbsidor [8].

3.6 JavaScript

JavaScript är ett plattformsoberoende, objektbaserat och dynamiskt skriptspråk. Språket är tänkt som ett komplement till HTML och CSS för att göra webbsidor mer interaktiva och dynamiska [9].

4. Kravspecifikation

Tillsammans med handledaren på företaget togs en kravspecifikation fram där önskad funktionalitet beskrevs. Ett av de större målen var att projektledarna själva skall kunna lägga in preliminära allokeringar i systemet. Dessa skall sedan kunna justeras och godkännas av administratörerna, de som ansvarar för allokeringsarbetet. Det lämpade sig därför att låta applikationens användare anta en av tre roller: administratör, projektledare eller basanvändare.

Exempel på basanvändare är konsulter som vill få en översikt av hur de är allokerade under de kommande veckorna. Basanvändarna skall alltså med applikationens hjälp kunna uppnå detta, men utöver det inte ha några andra rättigheter i systemet.

Användare med rollen projektledare skall kunna se och hantera de projekt som de leder. Det innefattar att enkelt få en överblick över hur mycket resurser som vart och ett av projekten har allokerat och kunna se hur projektets resurser är fördelade över roller och vilka konsulter som tilldelats dessa roller. Projektledare skall även kunna lägga in förfrågningar om att ändra allokeringar för de projekt som rör den aktuella projektledaren. Detta innefattar att kunna lägga till och ta bort resurser (konsulter/roller) och ändra arbetsomfattning för befintliga resurser. Projektledare skall även kunna se en lista på de allokeringsförfrågningar som ännu inte godkänts av en administratör.

Användare med rollen administratör skall ha tillgång till hela systemet. Detta innefattar full funktonalitet och tillgång till sidor för databashantering där administratören kan lägga till, ta bort och ändra uppgifter i databasen.

Icke-funktionella systemkrav:

- Systemet skall kunna köras på CGIs befintliga IT-struktur och därmed vara tillgängligt på företagets intranät.
- Systemet skall kunna beräkna (med hänsyn till frånvaro) och presentera konsulternas beläggningsgrad.
- Systemet skall kräva inloggning som styr vad en användare skall se och kunna använda.
- Flexibelt system som möjliggör vidare utveckling.

Funktionella krav (grundutförande):

- Lägga till, ändra och ta bort allokering.
- Lägga till, ändra och ta bort frånvaro.
- Lägga till, ändra och ta bort allokeringsförslag.
- Lägga till, ändra och ta bort systemanvändare.
- Godkänna och avslå allokeringsförslag.

I enlighet med iterationsordningen prioriteras hantering av allokeringar och frånvaro (Iteration 1) bland de funktionella kraven. Systemet kan sedan byggas ut med hanteringen av systemanvändare och allokeringsförslag (Iteration 2).

5. Analys

5.1 Nuvarande system

Fram tills nu har företaget använt sig av ett Excel-dokument för att utföra allokeringsarbetet. Arbetssättet utgörs i grund och botten av interaktion mellan två olika vyer i Excel-dokumentet, projektvyn och konsultvyn.

I projektvyn listas alla planerade allokeringar med avseende på ett antal uppgifter, varav de viktigaste inkluderar vilken konsult som utför en fördefinierad typ av arbete (roll) på ett projekt under en viss tid. Varje rad i listan identifierar en unik allokering och redovisar prognosen för denna med veckovis tillhörande arbetsomfattning. Omfattningen indikerar hur mycket av veckans arbetstid konsulten skall spendera på projektet, representerat i procentenheter. Ett utdrag ur projektvyn illustreras i figur 5.1 nedan.

Project Information			Resource Information				Availability					
Customer	EX	Projectname	Role	Comment	Consultant	Status		201409	201410	201411	201412	201413
Testföretag 01		P1	AC_Production		ACP1	Definitiv		100	100	100	100	100
Testföretag 01		P1	AC_Supply_chain		ACSC1	Definitiv		100	100	100	100	100
Testföretag 01		P1	Offshore		O1	Preliminär		100	100	100	100	100
Testföretag 01		P1	Offshore		O2	Preliminär		100	100	100	100	100
Testföretag 01		P1	Offshore		O3	Preliminär		100	100	100	100	100
Testföretag 01		P1	AC_Supply_chain		ACSC2	Preliminär		100	100	100	100	100
Testföretag 01		P1	Solution_Architect		SA1	Preliminär		100	100		100	100
Testföretag 01		P7	Technician		T1	Definitiv		25	25	25	25	25
Testföretag 01		P7	Subcontractor		S_1	Preliminär		25	25	25	25	25
Testföretag 01		P1	AC_Supply_chain		Vacant	Preliminär						
Testföretag 01		P1	AC_Supply_chain		ACSC5	Preliminär		100	100	100	100	100
Testföretag 01		P1	Offshore		O4	Preliminär		100	100	100	100	100
Testföretag 02		P9	AC_Production		ACP2	Definitiv						
Testföretag 02		P9	Developer		DEV1	Definitiv						
Testföretag 02		P9	AC_Finance		ACF1	Definitiv		50				
Testföretag 02		P9	Developer		DEV2	Preliminär		20	20	20	20	20
Testföretag 02		P9	Solution_Architect		SA2	Definitiv		20	20	20	20	20
Testföretag 02		P9	Developer		DEV5	Definitiv						
Testföretag 02		P9	AC_Supply_chain		ACSC5	Definitiv		20	20	20	20	20
Testföretag 02		P9	Developer		DEV3	Preliminär						
Testföretag 03		P2	AC_Finance		ACF1	Preliminär						
Testföretag 03		P2	Developer		DEV2	Preliminär						

Figur 5.1 *– Utdrag ur Excel-dokumentets projektvy.*

Konsultvyn listar samtliga konsulter och deras respektive beläggningsgrad per vecka. Beläggningsgraden beräknas automatiskt med data från projektvyn genom att summera arbetsomfattningarna för de olika projekten veckovis. Ett utdrag från konsultvyn visas i figur 5.2 nedan. Cellerna erhåller en viss färg beroende på beläggningsgraden.

Radetiketter	201409	201410	201411	201412	201413	201414	201415	201416	201417	201418	201419	201420
⊞ACF1	50											
⊞DEV2	20	20	20	20	20							
⊞SA2	20	20	20	20	20							
⊞DEV5	100	100	100	100	100	100	100	100	100	100		
⊟DEV3	5	25	45	25	30	65	30	25	25	22	20	20
Testföretag 02												
Testföretag 03	5	5	25	5	10	45	10	5	5	2		
Testföretag 05		20	20	20	20	20	20	20	20	20	20	20
⊞ACF2												
⊞S_2	100	100	100	100	100	100	100	100	100	100		
⊞T2	50	50	50	50	50	50	50	50	50	50		
⊞S_3				100	100	100	100	100	100	100	100	100
⊞ACP3						100	100	100	100	100	100	100
⊞O5	50	50	50	50	50	100	100	100	100	100	100	100
⊟ACF3	70	80	40	40	40	40	40	40	40	40	40	40
Testföretag 02	40	40										
Testföretag 05	30	40	40	40	40	40	40	40	40	40	40	40
⊞ACSC3	20	120	120	120	120	120	120	120	120	120	120	120

Figur 5.2 *– Utdrag ur Excel-dokumentets konsultvy.*

Excel-dokumentet görs tillgängligt genom företagets intranät och är åtkomligt enbart för de fåtal anställda som ansvarar för allokeringsarbetet inom den svenska företagsregionen. Varje sådan anställd

stämmer en gång i veckan möte med sina projektledare och går igenom vilka allokeringar som behövs för deras respektive projekt framöver. Det är sedan upp till den som utför allokeringsarbetet att med Excel-dokumentets hjälp identifiera de konsulter som har rätt kompetens och lägga in de nya allokeringarna så att konsulterna bibehåller en acceptabel beläggningsgrad. Den ideala beläggningsgraden för en konsult är 100 procent av dennes arbetstid.

Att hantera allokeringsarbetet på detta vis medför ett flertal problem som gör processen onödigt tidskrävande. Excel-dokumentet skrivskyddas då fler än en anställd försöker komma åt filen och kan därav inte redigeras av mer än en person åt gången. Detta tvingar de anställda att komma överens om tider under veckan då varje enskilt planeringsarbete ska ske. Tidsåtgången för detta tillsammans med möten med projektledarna blir stor. Ett annat problem är att konsulterna inte har tillgång till någon översikt av allokeringarna de är inblandade i. Projektledarna måste vidarebefordra planeringen till sina konsulter och informera dem hur deras kommande veckor ser ut. Excel-dokumentet tar ingen hänsyn till avvikelser som ledighet och sjukskrivning, vilket leder till att konsulter med sådan registrerad frånvaro framstår som underallokerade i planeringsöversikten. De få, goda egenskaperna hos Excel-dokumentet är att det går snabbt och enkelt att lägga in, redigera och ta bort data samt att navigera mellan de två vyerna.

5.2 Val av plattform

När det gäller utveckling av webbapplikationer finns det ett stort antal valmöjligheter då en utvecklingsplattform ska väljas. Några egenskaper som eftersöktes var god prestanda, enkel integration med databaser samt möjlighet för nya utvecklare att snabbt komma igång med att utveckla egna applikationer. Två alternativ som passar in på denna beskrivning är PHP och Microsofts ASP.NET. Dessa ramverk används för att konstruera och underhålla majoriteten av dagens webbsidor och webbapplikationer [10]. Eventuella prestandaskillnader mellan PHP och ASP.NET undersöktes i samband med en publikation år 2013, där ett flertal responstest med identiska förutsättningar genomfördes [11]. Det ansågs att avvikelserna var näst intill obefintliga och att inget av ramverken lämpade sig bättre än det andra med avseende på prestanda. I en annan publikation, från 2010, utvärderades ett flertal webbutvecklingsplattformar inklusive PHP och ASP.NET [12]. Resultaten visade att båda två har gott stöd för databashantering och även lämpar sig för mindre erfarna webbutvecklare.

Valet föll på Microsofts webbplattform ASP.NET med C#, då detta enkelt kan driftsättas i företagets befintliga IT-struktur. Utmaningen med en för vår del helt ny utvecklingsplattform bidrog till att ASP.NET var ett intressant val för projektet och bidrog till nyvunna kunskaper.

ASP.NET erbjuder ett flertal olika programmeringsmodeller att förhålla sig till vid webbutveckling. Bland dessa övervägdes Web Forms och MVC. Web Forms har fördelarna av ett stort urval programmerbara komponenter och en kort inlärningsperiod. MVC har en design som förenklar testning och ger utvecklaren full kontroll över HTML-genereringen men kräver en längre inlärningsperiod [13].

Programmeringsmodellen Web Forms valdes då det är snabbt och enkelt att komma igång med denna som ny ASP.NET-utvecklare.

5.3 Val av Verktyg

5.3.1 Dia

Dia är ett program som låter användaren konstruera strukturerade diagram, t. ex. UML-diagram, Entity Relationship-diagram och kopplingsscheman [14]. Programmet valdes för att kunna skapa illustrationer till rapporten.

5.3.2 Git

Git är ett fritt och öppet system för distribuerad versionshantering. Git är konstruerat med prestanda som utgångspunkt och är mycket resurssnålt [15]. Detta verktyg valdes för att stödja utvecklingsmetodiken för projektet och enkelt kunna dela kod mellan utvecklare.

5.3.3 SQL Server 2008

SQL Server 2008 är ett databashanteringssystem utvecklat av Microsoft [16]. ASP.NET kan integreras med flera olika databaser. CGI använder Microsoft Server SQL 2008 i sin IT-struktur, så det blev naturligt att använda denna databas för projektet.

5.3.4 Visual Studio 2013

Visual Studio 2013 är en programutvecklingsmiljö från Microsoft [17].

5.3.5 Internet Information Services (IIS)

IIS är en webbserver integrerad i de flesta versioner av Windows [18].

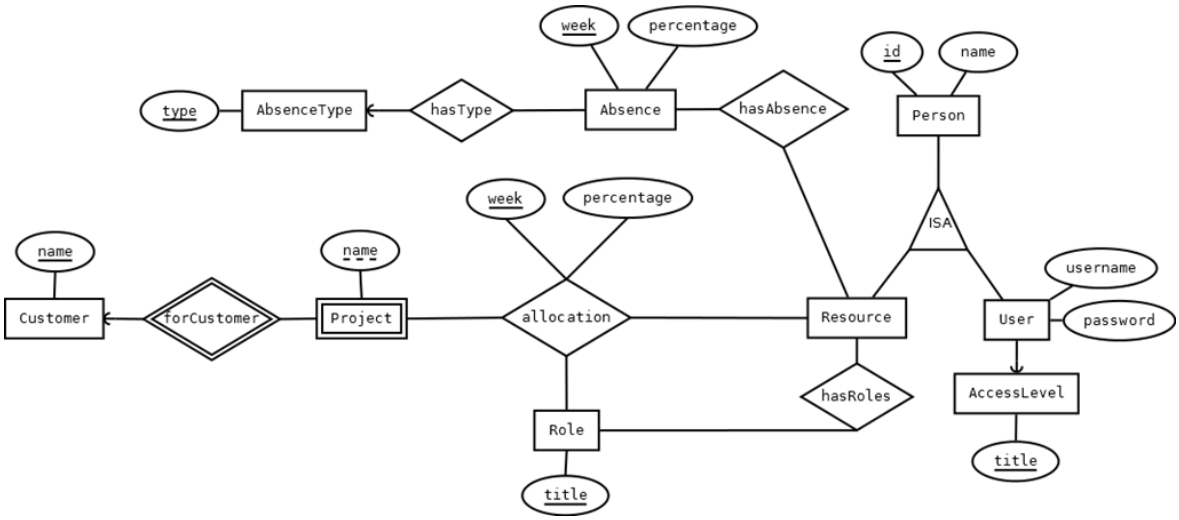
5.3.6 Systemplattform

En bärbar PC med operativsystemet Windows 7 utrustades med Microsoft SQL Server 2008 och där Microsoft IIS konfigurerades för att agera som värd för webbapplikationen. Microsoft Visual Studio 2013 valdes som utvecklingsmiljö och installerades. Denna systemplattform valdes för att fungera ihop med CGIs IT-struktur. Datorn placerades därefter på fast plats och fjärranslutning till den möjliggjordes för att kunna sköta konfiguration oavsett var man befann sig. Dessa förberedelser resulterade i att applikationen enkelt kunde driftsättas på webbservern genom Visual Studio och att den hade lokal åtkomst till databasen.

5.4 Domänmodell

Med kravspecifikationen som grund påbörjades arbetet med att modellera applikationens domän. En av de centrala frågeställningarna berörde hur man önskade representera arbetstid i modellen. Var det passande att behålla det tidigare formatet där veckobasis gällde, eller behövde man gå ner till dagsnivå och på så sätt uppnå högre upplösning? Efter en diskussion med företagets handledare togs beslutet att systemet skulle hantera veckor, då detta tillät allokeringstiden att begränsas till en enda kolumn i den framtida databaslösningen samt att det framstod som passande att presentera allokeringar per arbetsvecka för användaren.

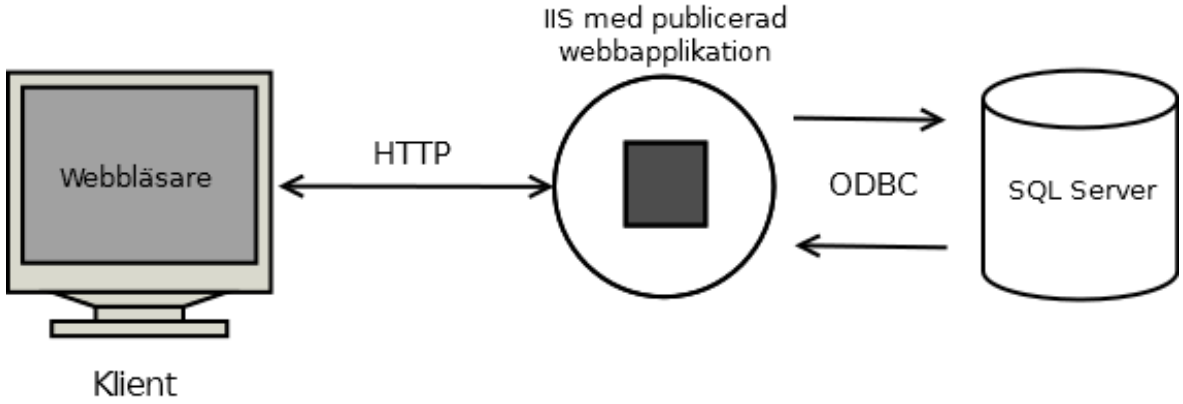
Domänmodellen som visas i figur 5.3, innefattar entiteter såsom företagskunder, konsulter (resurser) och roller som anger konsulters kompetens. Man såg helst att projektnamn inte skulle vara unika, utan att samma namn skulle kunna förekomma bland flera kunder. Ett projekt identifierades därför med ett projektnamn tillsammans med ett kundnamn. Mellan dessa entiteter kopplades relationer som beskriver deras interaktion. Den centrala relationen i domänen, allokeringsrelationen, utformades på så sätt att ett allokeringsfragment unikt identifieras av ett projekt, en konsult, en roll och en arbetsvecka. Med en tillhörande arbetsomfattning utryckt i procentenheter, bildar dessa allokeringsfragment en komplett konsultallokering med avseende på projekt, konsult och roll. Tanken var att låta varje fragment bestå av en egen rad i den framtida databastabellen för att enkelt kunna göra beräkningar på arbetsveckor och arbetsomfattningar i relation till dessa. Entiteten som representerar frånvaro modelleras med samma upplägg för att kunna kombineras med allokeringarna, varpå man genom detta potentiellt hade lyckats lösa problemet med att ha konsulternas frånvaro i åtanke och kunna beräkna deras egentliga beläggningsgrad.



Figur 5.3 *– ER-diagram över applikationens domän.*

6. Beskrivning av arkitektur

I ett driftsatt system kommunicerar webbapplikationen med klienten och databasen med olika protokoll. Kommunikationen mellan klient och webbapplikation sker med HTTP (Request - Response) medan kommunikationen mellan webbapplikation och SQL Server sker med hjälp av ODBC som illustreras i figur 6.1.

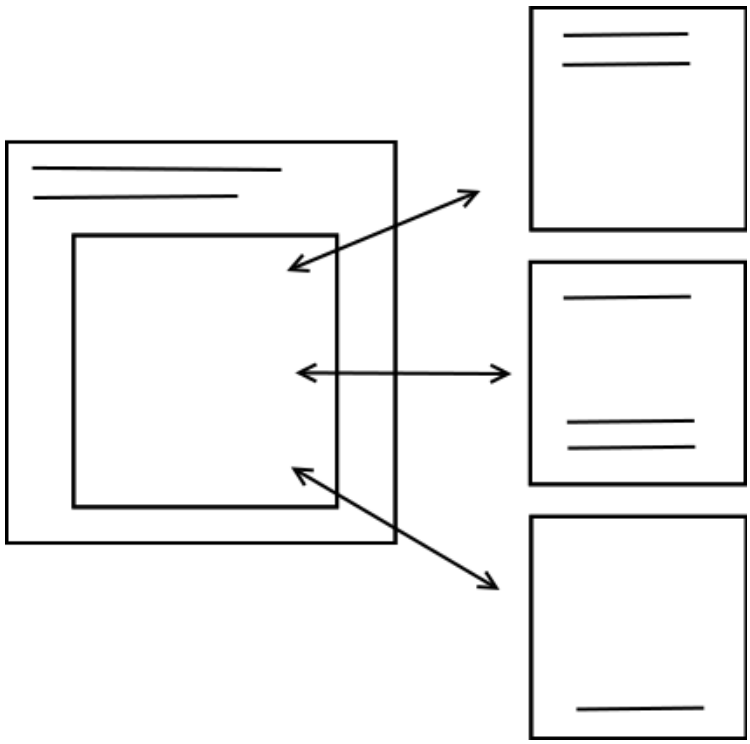


Figur 6.1 *– Översikt av arkitektur.*

7. Design

7.1 Master Page

En Master Page (huvudsida) utformades som kunde fungera som mall för samtliga Web Forms. Syftet var att ge enkel åtkomst till applikationens alla funktioner, detta genom att placera en alltid tillgänglig menyrad i sidhuvudet. I huvudsidan placerades länkarna till de inkluderade CSS- och JavaScript-filerna samt en deklaration av det webbformulär som utgör sidans innehåll. Utrymmet som var tänkt att tas upp av de webbsidor som använde huvudsidan markerades med ASP.NETs ContentPlaceHolder-komponenter. Huvudsidan kopplas med andra sidor genom att matcha komponenternas identifierare. ASP.NET vet på så sätt var innehållet skall hamna när webbsidan genereras.



Figur 6.2 *– En Web Form med Master Page där sidinnehåll kan variera inom en mall.*

7.2 Responsive Design med Bootstrap

En del HTML-element som exempelvis tabeller och rubriker framstår vanligtvis som trista och föråldrade i webbläsaren. För att göra applikationens utseende mer attraktivt och för att förhöja användarens upplevelse på klientsidan applicerades ramverket Bootstrap. Bootstrap underlättar webbutveckling med avseende på front-end genom att förse utvecklaren med färdiga CSS- och JavaScript-filer som innehåller många olika stilar och typer av funktionalitet. En stor fördel med Bootstrap är faktumet att det utvecklades med fokus på Responsive Design, vilket innebär att man tar användarens skärmupplösning i åtanke och skalar webbsidans proportioner och element automatiskt utefter detta. Möjligheten att presentera applikationens gränssnitt på det sättet är tilltalande med tanke på den stora andelen mobila enheter som används idag.

7.3 Entity Framework ORM

För att underlätta interaktion med databasen under implementeringsfasen nyttjades Microsoft Entity Framework. Entity Framework är ett Object-Relational Mapping (ORM) ramverk som låter webbutvecklaren få åtkomst till databastabeller och dess innehåll med minimalt antal rader av egen

kod. Verktyget användes för att generera modellklasser utifrån databastabellerna. Utvecklaren kan därmed enkelt hantera databasinnehåll i form av objekt.

7.4 Routing

ASP.NETs routingfunktion tillåter webbutvecklaren att själv bestämma hur URL-adresser i adressfältet skall tolkas av applikationen. I en applikation som inte använder routing hanteras en inkommande sidförfrågan vanligtvis av en fysisk fil med överrensstämmande namn på webbservern. Till exempel skulle en förfrågan om *http://myServer/myApplication/Projects.aspx* hanteras av en Web Form vid namn Projects.aspx. I många fall är det önskvärt att undvika detta; till exempel om användaren föredrar att navigera mellan applikationens sidor genom att själv skriva in specifika URL-adresser i adressfältet. Följden av det gör att navigeringen inte blir särskilt intuitiv då användaren tvingas memorera namnet på alla fysiska filer. Det kan även finnas skäl att dölja det faktum att applikationen bygger på Web Forms, något som aspx-filändelsen avslöjar. Möjligheten att kunna skriva in adresser i stil med *http://../Projects* är därför att föredra. Principen för att registrera en sådan route går ut på att vid applikationsstart kalla på en metod i .NETs RouteCollection-klass och specificera de adressmönster som stöds. Konventionen för detta är att skapa en klass med namnet RouteConfig innehållandes en metod som anropas från applikationens Application_Start-metod som registrerar de routes som definierats. Att skriva koden för registreringen direkt i Application_Start-metoden skall undvikas, eftersom den tenderar att växa i komplexitet i takt med resten av applikationen.

7.5 Användargränssnitt

I samråd med företagets handledare angående webbapplikationens utseende och arbetssätt beslöts att konceptet med projekt- och konsultvyerna var lämpliga att ta viss inspiration ifrån. Detta då de ger en god överblick som underlättar planeringen. För projektvyn valdes det att med hjälp av programmering sammanställa en lista som visar alla unika projekt. Detta till skillnad från det nuvarande systemet där varje projekt förekommer på flera rader. Användaren kan välja ett projekt i listan och erhålla relevant information genom att expandera raden. Samma princip applicerades i webbapplikationens konsultvy för att förmedla vilka projekt konsulterna är involverade i.

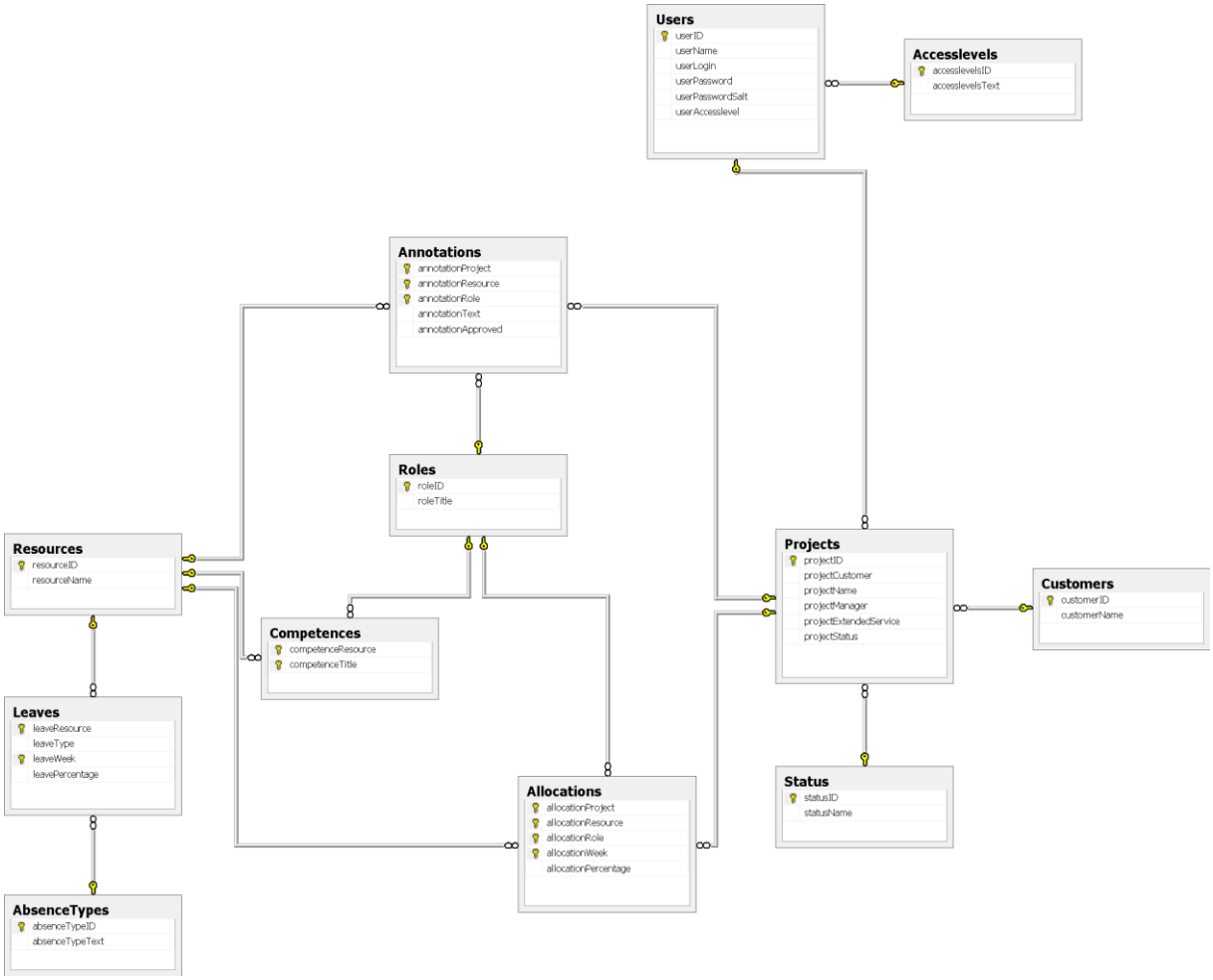
8. Implementering

8.1 Databas

8.1.1 Struktur

Databasstrukturen togs fram i ett tidigt skede. Strukturen illustrerades först i ett Entity Relationship-diagram. Första implementeringen av databasstrukturen skedde i MySQL, då MySQL var bekant sedan innan och en plattform för MySQL fanns lättillgänglig. Både modellen och strukturen av databasen justerades i flera omgångar efter diskussion med CGI för att bli så bra och flexibel som möjligt. Beslut togs bland annat om att inte ha några databärare som nycklar, utan istället komplettera med surrogatnycklar för att lätt kunna ändra olika datavärden i en tabell utan att direkt påverka de övriga tabellerna. Strukturen av databasen designades så att den inte skulle hindra vidare utveckling av applikationen med ytterligare funktionalitet, såsom loggning av händelser i systemet.

Den näst intill färdiga strukturen migrerades sedan från MySQL till Microsoft SQL Server 2008 på den avsedda plattformen som tillhandahölls av CGI. Detta gjordes med hjälp av verktyget SQL Server Migration Assistant (SSMA). Som hjälpmedel under utvecklingsarbetet installerades administrationsverktyget Adminer för att kunna komma åt databasen via ett webb-interface. Databasens slutliga struktur kan betraktas i figur 8.1.



Figur 8.1 *– Databasens struktur.*

8.1.2 Nycklar och restriktioner

Surrogatnycklar applicerades på alla tabeller som tillkom ur applikationsdomänens entiteter, i form av SQL Servers IDENTITY-egenskap. Det innebär att nya tabellrader tilldelas ett auto-inkrementerat

heltal som används som primärnyckel. De kolumner som uppstod som resultat av domänens kandidatnycklar definierades med UNIQUE-restriktion.

8.1.3 Normalisering

För att förenkla framtida utbyggnad av databasstrukturen normaliserades tabellerna. Detta möjliggör att nya tabeller kan tillkomma utan att den existerande strukturen behöver justeras. Andra fördelar med normaliseringen innefattar att tabellerna och dess relationer blir mer begripliga, samt att överflödig data inte lagras. Tabell 8.1 visar situationen som uppstod då en kommentar bifogades i samband med varje allokering:

Allokeringar

Projekt	Resurs	Roll	Vecka	Omfattning	Kommentar
Johanneberg	John	Utvecklare	10	50	Testning
Johanneberg	John	Utvecklare	11	50	Testning
Lindholmen	Alexander	Arkitekt	15	80	Preliminär

Tabell 8.1 *– Visualisering av allokeringstabell.*

Lagring av kommentarer i allokeringstabellen ledde till att de associerades med varje allokeringsfragment. För att spara utrymme normaliserades allokeringstabellen och det medförde att kommentarerna kopplades till allokeringarna som det var avsett, som i tabell 8.2.

Annoteringar

Projekt	Resurs	Roll	Kommentar
Johanneberg	John	Utvecklare	Testning
Lindholmen	Alexander	Arkitekt	Preliminär

Tabell 8.2 *– Visualisering av annoteringstabell.*

8.1.4 SQL-vyer

I databasen definieras ett antal SQL-views (vyer). Vyer är användbara för att sammanställa data från flera olika tabeller och för att dölja den underliggande datastrukturen från omvärlden. En av vyerna, OccupancyView, beräknar konsulternas faktiska beläggningsgrad och definieras enligt kodavsnitt 8.1.

```
CREATE VIEW OccupancyView AS
SELECT a.resourceID, a.allocationWeek, a.occupancy +
          ISNULL(Leaves.leavePercentage, 0) AS occupancy
FROM (
          SELECT Resources.resourceID, Allocations.allocationWeek,
          SUM(Allocations.allocationPercentage) AS occupancy
          FROM Allocations JOIN Resources
                    ON allocationResource = resourceID
          GROUP BY resourceID, allocationWeek
     ) a
LEFT JOIN Leaves ON a.resourceID = leaveResource
          AND a.allocationWeek = Leaves.leaveWeek
```

Kodavsnitt 8.1 *– SQL-kod för Occupancy View.*

Principen går ut på att i den inre SELECT-satsen veckovis summera alla konsulters respektive arbetsomfattningar och att producera en tabellrad för varje konsult och vecka. Resultatet kombineras

sedan med frånvarotabellen för att innan den slutgiltiga projektionen kunna addera eventuell frånvaro till arbetsomfattningen. På så sätt beräknas konsultarnas egentliga beläggningsgrad.

8.2 Webbapplikation

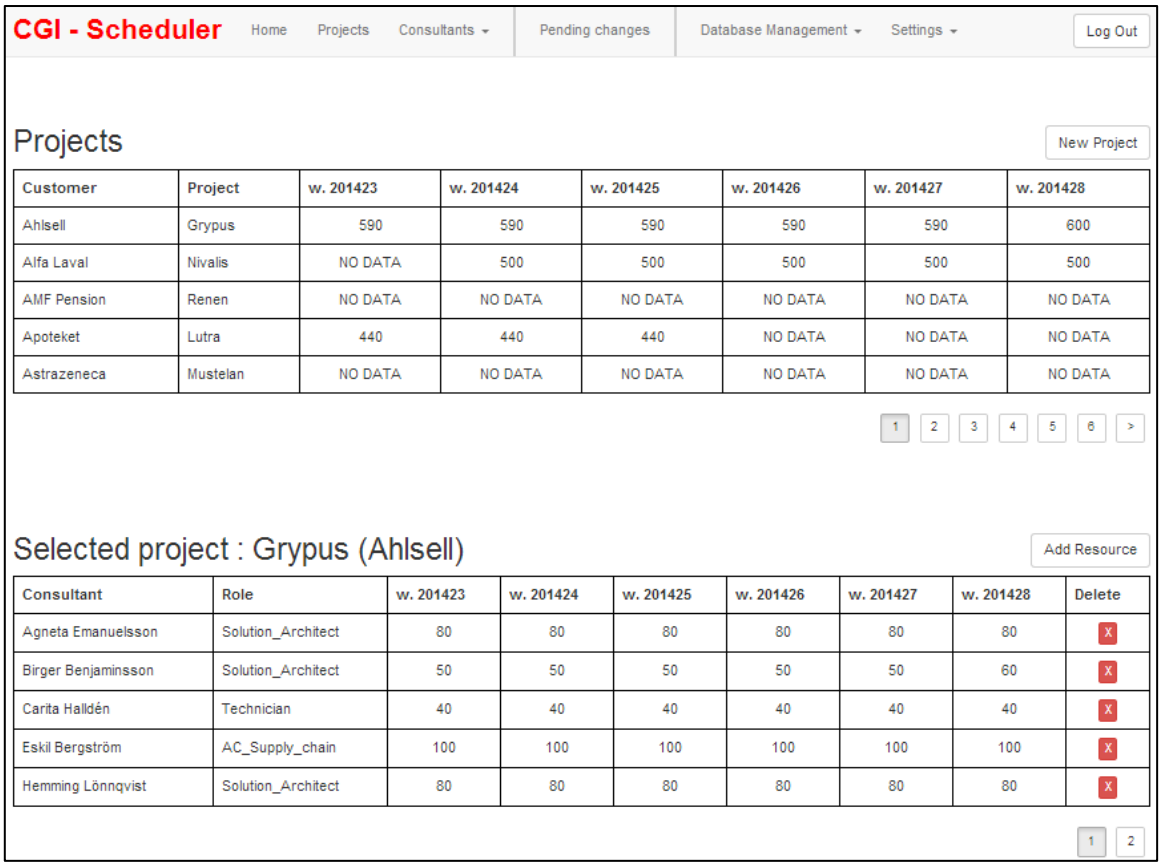
8.2.1 Inloggningsportal

En inloggningssida utformades som illustrerat i figur 8.2. All nödvändig validering implementerades och försiktighetsåtgärder mot skadlig kod hölls i åtanke. ASP.NET har en inbyggd funktion för validering av inmatade data från användaren. Denna funktion ”ValidateRequest” är aktiverad som standard för alla aspx-sidor. Utöver det kontrolleras även att de inmatade textsträngarna möter vissa krav beroende på vad det är för data användaren förfrågas att mata in. Dessutom skyddar Entity Framework-modellen delvis mot SQL-injections [19]. För ytterligare säkerhet läggs en slumpmässig textsträng till varje lösenord och krypteras med hashningsfunktionen SHA1.

Figur 8.2 *– Webbapplikationens inloggningsportal.*

8.2.2 Projektvy

Webbapplikationens motsvarighet till Excel-dokumentets projektvy utformades som en tabell där varje unikt projekt listas. Varje rad specificerar projektets kund, projektnamn och den veckovis sammanlagda arbetsomfattningen för alla involverade konsulter. Om ett projekt i listan väljs genom att klicka på motsvarande rad, så visas ytterligare en tabell med de allokerade konsulterna på projektet och deras respektive arbetsomfattning per vecka. De data som visas i tabellerna hämtas i flera omgångar från olika vyer i databasen. Detta för att kunna begränsa antalet rader som visas för att få en användarvänlig layout på sidan. Användaren kan sedan navigera genom tabellen med sidknapparna. Ett utdrag ur applikationens projektvy visas i figur 8.3.



Figur 8.3 *– Utdrag ur webbapplikationens projektvy.*

Beroende på vilken tabellrad som väljs körs en JavaScript-funktion som skickar webbsidans formulär till servern, se kodavsnitt 8.2. I back-end analyseras parametern som anger vilken tabellrad som valdes och den relevanta informationen skickas tillbaka till klienten.

```
function __doPostBack(eventTarget, eventArgument) {
    if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
        theForm.__EVENTTARGET.value = eventTarget;
        theForm.__EVENTARGUMENT.value = eventArgument;
        theForm.submit();
    }
}
function selectProject(project) {
   __doPostBack('projectSelected', project)
}
function selectConsult(consultant) {
   __doPostBack('consultantSelected', consultant)
}
```

Kodavsnitt 8.2 *– JavaScript-funktioner som körs för att skicka webbformuläret.*

8.2.3 Konsultvy

Då webbapplikationens konsultvy designades togs mycket inspiration ifrån företagets tidigare nämnda Excel-dokument. Detta valdes för att arrangemanget ger användaren en god översikt av prognosen för konsulternas beläggningsgrad och en motsvarighet kunde utan större svårigheter implementeras. Resultatet blev en tabell med villkorsstyrd formatering, där celler innehållande en beläggningsgrad tilldelas en viss färg baserat på värdet. Ett utdrag ur applikationens konsultvy visas i figur 8.4 nedan.

Figur 8.4 *– Utdrag ur webbapplikationens konsultvy.*

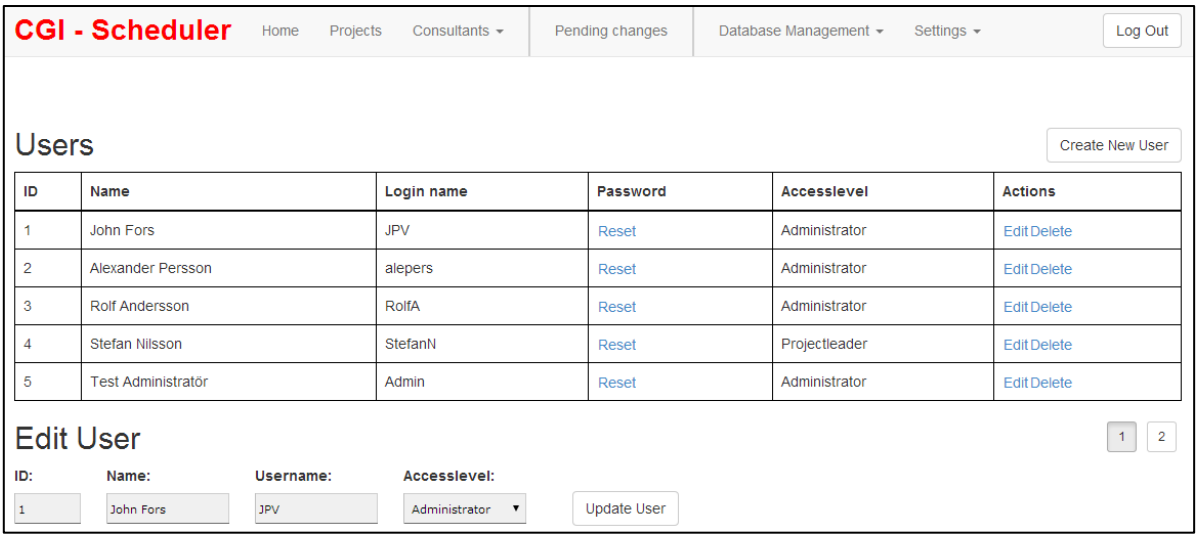
Tabellen konstrueras dynamiskt i back-end då sidan begärs. Koden som exekveras för att generera tabellen visas i kodavsnittet 8.3.

```
…
List<Resource> resources = GetResources();
foreach ( Resource resource in resources ) {

    // Skapa fältet "weekOccupancyCells" (TableCell[]):
    …
    foreach( TableCell tc in weekOccupancyCells ) {

        tc.Text = GetWeekOccupancy(resource.ID, currentWeek);
        short percentage = Convert.ToInt16(tc.Text);

        // Analysera värde och tilldela motsvarande CSS-klass..
        if( percentage == … )
        …
    }

    // Lägg till cellerna till tabellraden..
    …
    // Lägg till tabellraden till tabellen..
    …
}
…
```

Kodavsnitt 8.3 *– Kod för att konstruera konsultvyns tabell.*

8.2.4 Databashantering

En sida för hantering av systemets användare implementerades, ”Database Management - Users”. Sidan ger systemadministratörer möjlighet att hantera användare av systemet enligt kravspecifikationen. På denna sida kan en administratör skapa nya användare, ta bort användare och redigera befintliga användares namn, liksom användarnamn och användarroll samt återställa användares lösenord. Layouten för sidan, då en användare redigeras kan ses i figur 8.5**.**



Figur 8.5 *– Webbapplikationens vy för hantering av systemanvändare.*

8.3 Applikationens fil- och klasstruktur

Filstrukturen för applikationen visas i figur 8.6. ”Account”-mappen som håller filerna för inloggningssidan är tillsammans med ”Content”-mappen de enda filerna som är tillgängliga utan autentisering.

```
└──Scheduler
   ├──App_Start : Håller klasser för övergipande funktioner t.ex. routing och skripthantering.
   ├──Content : Innehåller applikationens stylesheets (CSS) och bilder.
   │  ├──CSS
   │  └──Images
   ├──Models : Håller applikationens Entity Framework-klasser, som är gränssnittet mot databasen.
   │  └──Repository
   ├──Pages : Har alla filer för de olika sidorna i applikationen.
   │  ├──Account : Inloggningssidan.
   │  ├──Admin : Administratörssidor för databashantering.
   │  └──Helpers : Hjälpklasser med generella programmeringsfunktioner.
   ├──Properties : Håller klass med information om applikationen, så som version och copyright.
   └──Scripts : Innehåller skriptfiler, för jQuery och Bootstrap.
```

Figur 8.6 *– Applikationens filstruktur.*

Applikationens klasstruktur kan ses i Appendix B, där grupperingen av applikationens klasser i olika namespaces visualiseras. Illustrationerna är inte kompletta, men visar applikationens huvudstruktur. Alla ”Properties” för DbContext-klasserna har en motsvarande klass för att representera poster i respektive tabell som objekt. Exempel på en sådan klass är klassen ”Customers”. Alla sådana klasser ligger under namespace Scheduler.Models.

9. Resultat

Arbetet resulterade i en väl fungerande webbapplikation med tillhörande databasstruktur. Webbapplikationen utvecklades i ASP.NET med Web Forms som programmeringsmodell och använder Microsoft SQL Server som databashanteringssystem. Vyer definierades i databasen för att förse applikationen med information anpassad för de implementerade funktionerna.

Systemet uppfyller samtliga icke-funktionella krav:

- Systemet skall kunna köras på CGIs befintliga IT-struktur och därmed vara tillgängligt på deras intranät.
- Systemet skall kunna beräkna (med hänsyn till frånvaro) och presentera konsulternas beläggningsgrad.
- Systemet skall kräva inloggning som styr vad en användare skall se och kunna använda.
- Flexibelt system som möjliggör vidare utveckling.

Systemet uppfyller de funktionella krav som tillhör den första iterationen:

- Lägga till, ändra och ta bort allokering.
- Lägga till, ändra och ta bort frånvaro.
- Lägga till, ändra och ta bort systemanvändare.

10. Slutsats

10.1 Kritisk diskussion

Det fanns redan från början insikt om att det är en stor och tidskrävande uppgift att utveckla ett resurshanteringssystem från grunden. Dessutom innebar den saknade erfarenheten av den valda plattformen att detta projekt kunde antas ta ytterligare tid i anspråk. Så blev fallet och medförde att iteration 2 inte implementerades på grund av tidsbrist. Tidsplanen följdes som väntat fram till implementeringsfasen. Denna fick förlängas till följd av det initiala arbetet med att bekanta sig med en för vår del ny plattform.

Då programmeringsmodellen för projektet har varit ASP.NET Web Forms, har koden för layout och funktionalitet varit svår att separera på ett bra sätt. Detta har försvårat, för att inte säga omintetgjort möjligheterna för automatiserade tester, såsom Unit-tester för den funktionella koden. Perioden som hade avsatts för testning användes istället till ytterligare implementering. Detta möjliggjorde därmed att webbapplikationen utvecklades till den grad som beskrivs i resultatkapitlet. Då någon form av testning ändå är essentiellt för att säkerställa kvalitet och funktion har detta istället utförts manuellt under utvecklingen av applikationen. Projektet är i detta fall inte beroende av automatiska tester. I ett projekt där testningen är mera kritisk hade Web Forms varit fel programmeringsmodell. För att enkelt kunna skriva automatiserade tester hade programmeringsmodellen ASP.NET MVC varit det rätta valet då det ger goda förutsättningar för detta. Om MVC valts som programmeringsmodell för projektet, hade tiden för inlärning troligtvis blivit ännu längre. På grund av tidsramen för projektet skulle det begränsat funktionaliteten i applikationen ytterligare.

Jämfört med det nuvarande systemet med Excel-dokumentet så har det webbaserade systemet flertalet fördelar. Det nya systemet kan hantera frånvaro, det ger en mer överskådlig bild över projekten och kan användas av flera användare samtidigt.

Inmatning och uppdatering sker något långsammare i det nya systemet, detta då Excel-programmets arbetssätt är optimerat för enkel och snabb inmatning och uppdatering av data. Det fanns en ambition att implementera möjligheter att filtrera informationen i tabellerna. Detta var inte prioriterat och genomfördes därför inte på grund av tidsbrist.

10.2 Teknikens roll i samhället

Skulle det webbaserade resurshanteringssystemet sättas i drift kan man uppnå förbättring med avseende på hållbar utveckling. Detta eftersom det nuvarande resurshanteringssystemet kräver att administratörer och projektledare samlas för att diskutera planeringen. Med det nya systemet kommer detta inte att behövas, då projektledarna kommer att ha möjlighet att kommunicera resursefterfrågan genom applikationen. Man kan därför i teorin reducera transportkostnader och bränsleförbrukning.

Det nya systemet ger också CGI bättre förutsättningar för att bedriva en tidseffektiv resursplanering som möjliggör att verksamheten kan utnyttja resurserna till fullo.

10.3 Vidareutveckling

Webbapplikationen har stor potential att vidareutvecklas. Under projektets gång har flertalet funktioner diskuterats, men tiden har inte räckt till för att implementera dem alla. Här följer några funktioner som inte blivit implementerade under projektet, men som skulle vara intressanta att realisera i ett senare skede och som skulle kunna förbättra applikationen:

- Lägga till hanteringssystem för samtliga tabeller i databasen under menyn ”Database Management”.
- Lägga till stöd för preliminära allokeringar (iteration 2).
- Lägga till filtreringsfunktioner i tabellerna på de olika kolumnerna.
- Lägga till fler inställningsmöjligheter för användaren, så som antalet visade tabellrader på en sida.
- Omstrukturera koden, för möjliggörande av Unit-tester.
- Implementera kommunikation av färdig planering till konsulter (iteration 3).

Referenser

[1] ”World Wide Web Consortium (W3C),” [Online]. Available: http://www.w3.org/standards/webdesign/htmlcss. [Använd 21 Maj 2014].

[2] ”Microsoft Development Network - Overview of the .NET Framework,” Microsoft, 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/zw4w595w.aspx. [Använd 23 Maj 2014].

[3] ”Microsoft Developer Network - Programming Languages,” Microsoft, 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa292164%28v=vs.71%29.aspx. [Använd 23 Maj 2014].

[4] ”Microsoft Developer Network - ASP.NET Overview,” Microsoft, 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx. [Använd 23 Maj 2014].

[5] ”Introduction to ASP.NET Web Forms,” Microsoft, 2014. [Online]. Available: http://www.asp.net/web-forms/what-is-web-forms. [Använd 27 Maj 2014].

[6] M. M. M. S. Adam Freeman, ”ASP.NET Handbok,” i *Pro ASP.NET 4.5 in C#*, New York, APRESS, 2013, p. 1228.

[7] ”ASP.NET MVC Overview,” Microsoft, 2009. [Online]. Available: http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview. [Använd 27 Maj 2014].

[8] ”What is PHP?,” PHP Documentation Group, 2014. [Online]. Available: http://www.php.net/manual/en/intro-whatis.php. [Använd 27 Maj 2014].

[9] ”JavaScript Overview - JavaScript MDN,” Mozilla, 2014. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/JavaScript_Overview#What_is_JavaScript.3F. [Använd 28 Maj 2014].

[10] I. Alshanetsky, ”Web Advent,” 14 December 2010. [Online]. Available: http://webadvent.org/2010/usage-statistics-by-ilia-alshanetsky. [Använd 28 April 2014].

[11] T. Mirzoev och L. Sack, ”Webpage Load Speed: ASP.net vs. PHP,” *i-Manager's Journal on Information Technology*, vol. 2, pp. 1-7, 2013.

[12] I. F. Amadin och E. Nwelih, ”An Empirical Comparison Of: HTML, PHP, COLDFUSION, PERL, ASP.NET, JAVASCRIPT, VBSCRIPT, PYTON AND JSP,” *Global Journal of Computer Science and Technology*, vol. 10, pp. 9-17, 2010.

[13] ”Compatibility of ASP.NET Web Forms and MVC,” Microsoft, 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd381619(v=vs.98).aspx. [Använd 13 Juni 2014].

[14] Dia Diagram Editor, [Online]. Available: http://www.dia-installer.de. [Använd 25 Maj 2014].

[15] ”Git --distributed-is-the-new-centralized,” Open source, 2014. [Online]. Available: http://git-scm.com. [Använd 26 Maj 2014].

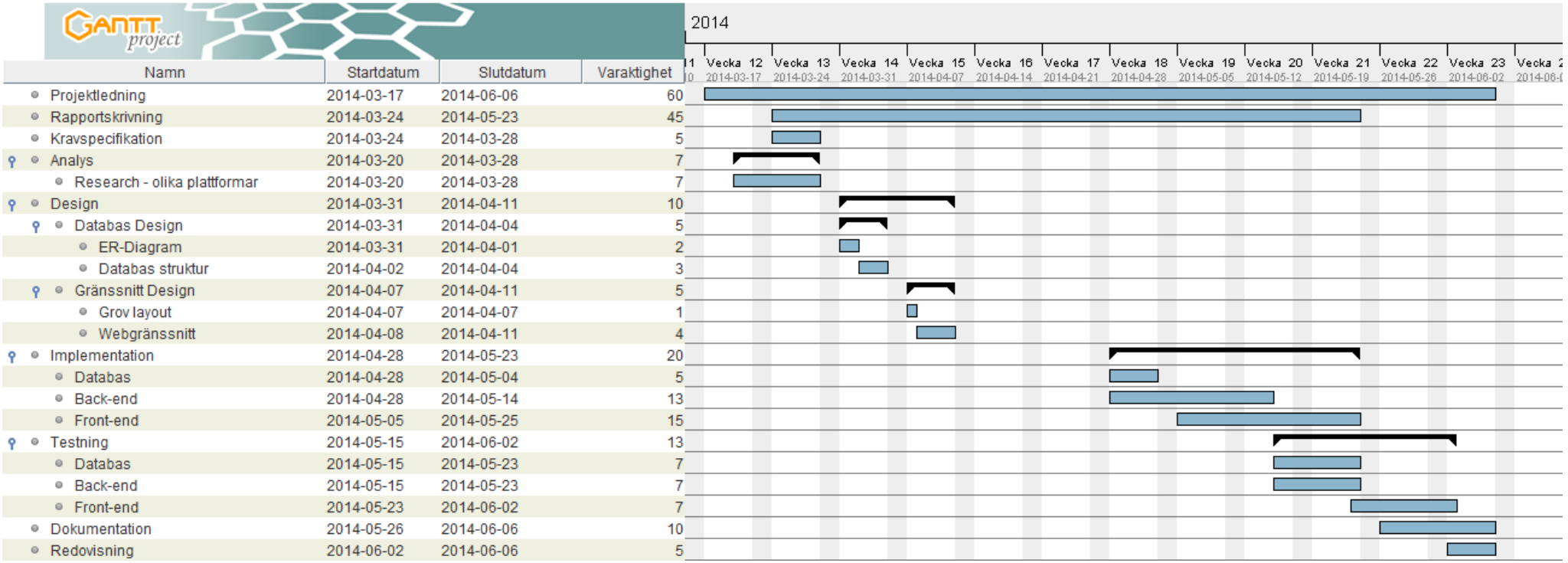
[16] ”SQL Server,” Microsoft, [Online]. Available: www.microsoft.com/en-us/sqlserver/default.aspx. [Använd 26 Maj 2014].

[17] ”Visual Studio,” Microsoft, [Online]. Available: http://www.visualstudio.com/. [Använd 26 Maj 2014].

[18] ”IIS,” Microsoft, [Online]. Available: http://www.iis.net/. [Använd 26 Maj 2014].

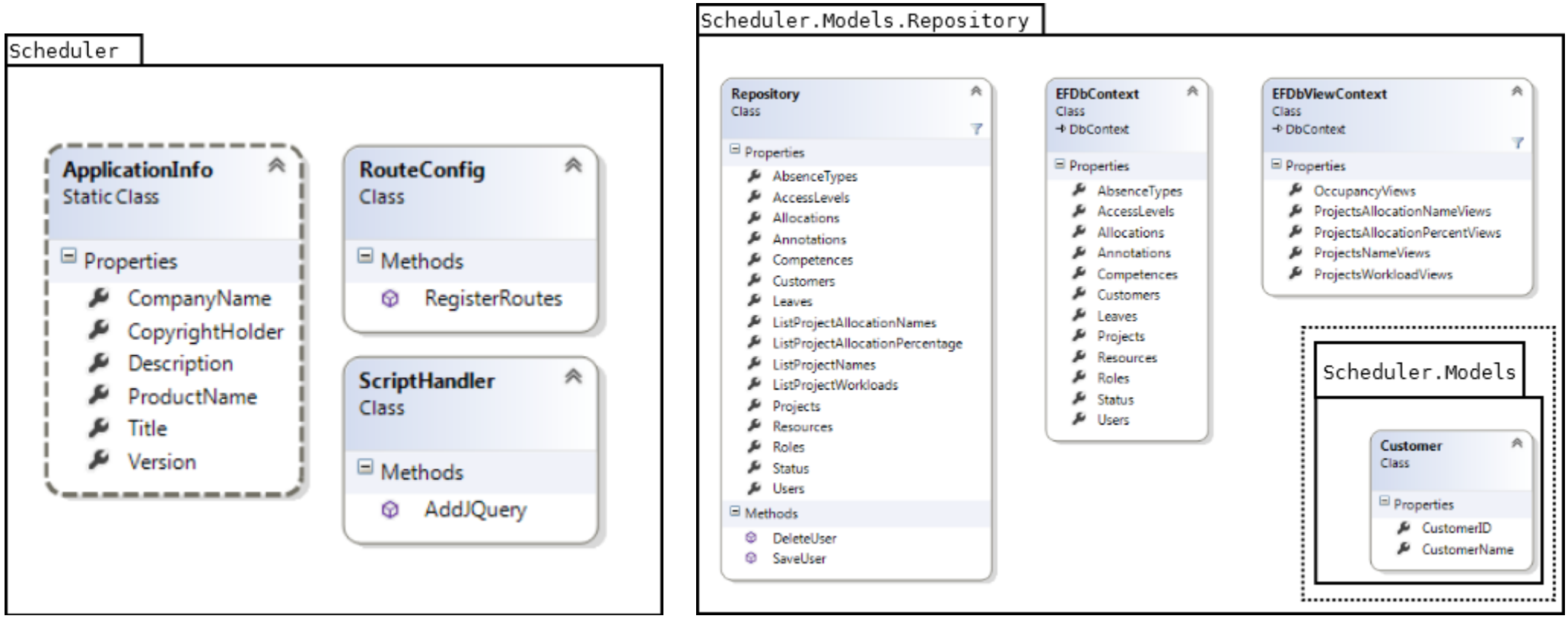
[19] ”Microsoft MSDN Library,” Microsoft, 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/vstudio/cc716760%28v=vs.100%29.aspx. [Använd 22 Maj 2014].

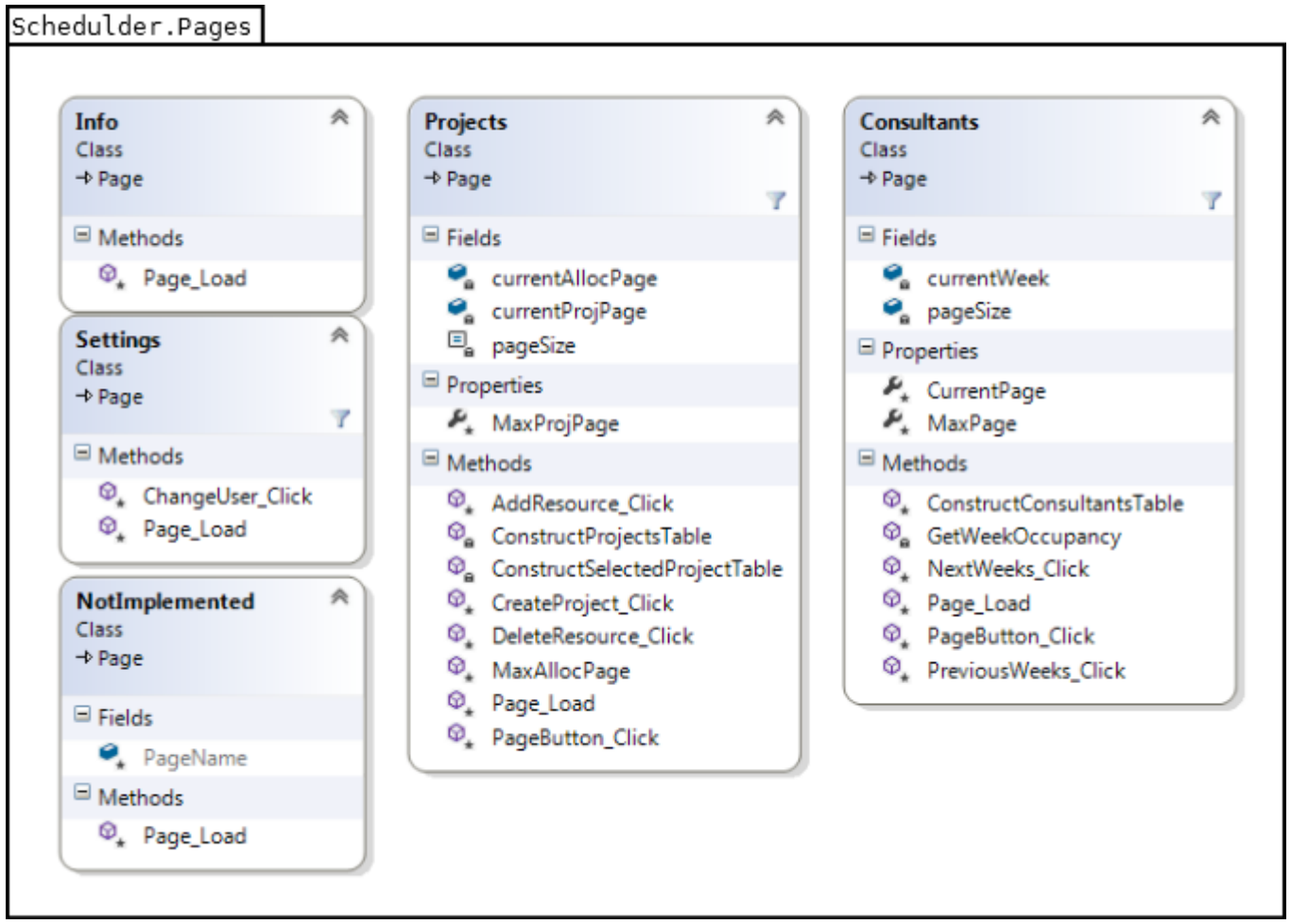
Appendix A – Gantt-schema

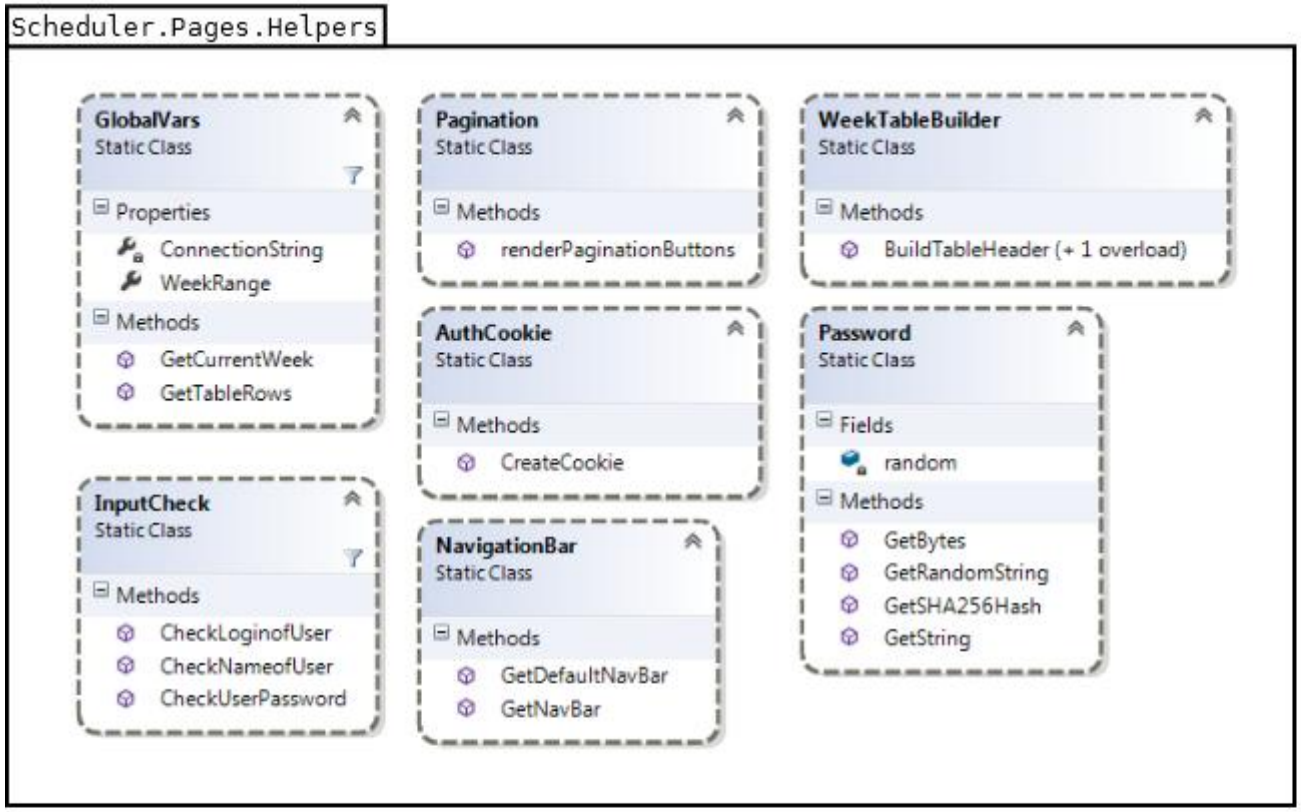


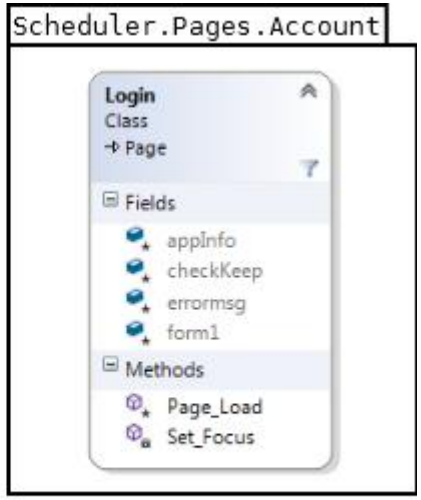
Namn	Startdatum	Slutdatum	Varaktighet
Projektledning	2014-03-17	2014-06-06	60
Rapportskrivning	2014-03-24	2014-05-23	45
Kravspecifikation	2014-03-24	2014-03-28	5
Analys	2014-03-20	2014-03-28	7
Research - olika plattformar	2014-03-20	2014-03-28	7
Design	2014-03-31	2014-04-11	10
Databas Design	2014-03-31	2014-04-04	5
ER-Diagram	2014-03-31	2014-04-01	2
Databas struktur	2014-04-02	2014-04-04	3
Gränssnitt Design	2014-04-07	2014-04-11	5
Grov layout	2014-04-07	2014-04-07	1
Webgränssnitt	2014-04-08	2014-04-11	4
Implementation	2014-04-28	2014-05-23	20
Databas	2014-04-28	2014-05-04	5
Back-end	2014-04-28	2014-05-14	13
Front-end	2014-05-05	2014-05-25	15
Testning	2014-05-15	2014-06-02	13
Databas	2014-05-15	2014-05-23	7
Back-end	2014-05-15	2014-05-23	7
Front-end	2014-05-23	2014-06-02	7
Dokumentation	2014-05-26	2014-06-06	10
Redovisning	2014-06-02	2014-06-06	5

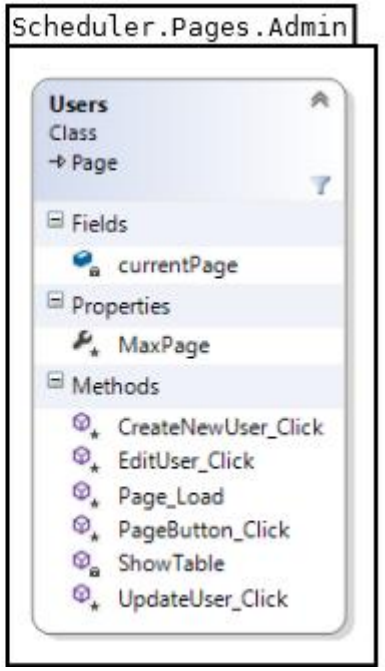
Appendix B – Klasstruktur




Schedulder.Pages
Info
Class
Page
Methods
Page_Load
Settings
Class
Page
Methods
ChangeUser_Click
Page_Load
NotImplemented
Class
Page
Fields
PageName
Methods
Page_Load
Projects
Class
Page
Fields
currentAllocPage
currentProjPage
pageSize
Properties
MaxProjPage
Methods
AddResource_Click
ConstructProjectsTable
ConstructSelectedProjectTable
CreateProject_Click
DeleteResource_Click
MaxAllocPage
Page_Load
PageButton_Click
Consultants
Class
Page
Fields
currentWeek
pageSize
Properties
CurrentPage
MaxPage
Methods
ConstructConsultantsTable
GetWeekOccupancy
NextWeeks_Click
Page_Load
PageButton_Click
PreviousWeeks_Click


Scheduler.Pages.Helpers
GlobalVars
Static Class
Properties
ConnectionString
WeekRange
Methods
GetCurrentWeek
GetTableRows
Pagination
Static Class
Methods
renderPaginationButtons
WeekTableBuilder
Static Class
Methods
BuildTableHeader (+ 1 overload)
AuthCookie
Static Class
Methods
CreateCookie
Password
Static Class
Fields
random
Methods
GetBytes
GetRandomString
GetSHA256Hash
GetString
InputCheck
Static Class
Methods
CheckLoginofUser
CheckNameofUser
CheckUserPassword
NavigationBar
Static Class
Methods
GetDefaultNavBar
GetNavBar


Scheduler.Pages.Account
Login
Class
Page
Fields
appInfo
checkKeep
errormsg
form1
Methods
Page_Load
Set_Focus


Scheduler.Pages.Admin
Users
Class
Page
Fields
currentPage
Properties
MaxPage
Methods
CreateNewUser_Click
EditUser_Click
Page_Load
PageButton_Click
ShowTable
UpdateUser_Click

B