CHALMERS

charmportalen

WEBBPORTAL FÖR NORDENS STÖRSTA ARBETSMARKNADSMÄSSA!

CHARMportalen
– Ett internsystem för CHARM

Kandidatarbete vid Data- och Informationsteknik

CAROLINE STRANDBERG
OSCAR CARLSSON
MIRAC GÜNES

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborg, Sverige 2012
Kandidatarbete/rapport nr 2012:17

CHARMportalen
Ett internsystem för CHARM
Caroline Strandberg, Oscar Carlsson & Mirac Günes
Handledare Christer Carlsson, examinator Sven-Arne Andreasson & Arne Linde.

Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg, 2012
Sverige

Sammandrag

Varje år arrangeras Nordens största arbetsmarknadsmässa, Chalmers Studentkårs Arbetsmarknadsmässa (CHARM), som engagerar hundratals studenter. Denna rapport avhandlar ett kandidatarbete vars syfte är att utveckla ett nytt internsystem till CHARM-kommittéen. Det nuvarande internsystemet uppfyller inte de krav på funktionalitet som ställs av kommittéen.

Studenter som vill engagera sig i mässan kan ansöka om att bli värdar. En värd har ett ansvarsområde och behöver då få tillgång till väsentlig information ur internsystemet. Internsystem skall kunna hantera värdansökningar och låta kommittéen tilldela värdroller till de studenter som skickat in ansökan. Företag som vill delta i mässan anmäler sig på en extern hemsida.

Det nyutvecklade internsystemet är en webbaserad portal med all information lagrad i en databas. Internsystemet har en väl genomtänkt arkitektur för att enkelt kunna vidareutvecklas och underhållas.

För att beskriva internsystemet har ett flertal modeller tagits fram, som bidrar till förståelse och tydliggör systemets funktionalitet. Rapporten behandlar också en utvärdering av programmeringsspråk med ramverk samt databashanterare, för att bestämma vad internsystemet skall implementeras i.

På grund av projektets storlek behandlar kandidatarbetet endast de mest grundläggande funktionerna till det nya internsystemet.

Abstract

Every year the largest career fair in the Nordic countries, Chalmers Student Unions Career Fair (CHARM), is arranged, involving hundreds of students. This thesis is devoted to the development of a new internal system for the CHARM Committee, as the current system does not fulfil the functional requirements set by the Committee.

Students who wish to be involved in the fair can apply to become hosts. A student who is made host has a responsibility and needs access to essential information from the system. The system should therefore be able to manage host applications and allow the Committee to assign host roles to students who submit an application. Companies wishing to attend the fair apply at an external site.

The system presented in this thesis is a web portal with all its information stored in a database. The system is well-designed and modeled to facilitate further development and maintenance.

In order to describe and ensure understanding of the system, several models have been designed. This thesis also discusses an evaluation of programming languages with frameworks, as well as database management systems for implementation.

Due to the size of the project, this thesis only presents the beginning of the new system.

Förord

Rapporten redovisar ett kandidatarbete som utfördes våren 2012 vid Institutionen för Data- och informationsteknik, Chalmers Tekniska Högskola. Arbetet avhandlar ett nytt internsystem åt Chalmers Studentkårs Arbetsmarknadsdagars kommittée.

Projektgruppen önskar tacka Christer Carlsson för de timmar han lagt ner på att handleda projektet i rätt riktning. Vi vill tacka IT-ansvariga i CHARMk'11 Max Sikström och Victor Jansson för alla timmar de har lagt ner för att ta fram en kravspecifikation tillsammans med oss. Samt tacka Marina Yudanov för de timmar hon lagt ner som stöd vid rapportskrivning.

Denna sida har med avsikt lämnats tom.

Innehåll

Ordlista

Nedan följer förklaringar av olika begrepp som används i rapporten. Ordlistan avslutas med en lista över olika programvarulicenser och tillhörande beskrivningar.

Active Records	Arkitekturmönster för hantering av data mot en databas. Ett objekt representerar en rad i databastabellen.
API	Application Programming Interface. Definierar funktioner och klasser.
Cookie	Temporär fil som lagras på klientens dator för att lagra det aktuella tillståndet på en webbsida.
Copyleft	Innebär att modifierade versioner av en programvara skall följa samma licensvillkor som orginalversionen. Ordvits på copyright som används i samband med öppen källkod.
CRUD	Create, Read, Update and Delete. Funktioner för databasinteraktion.
CSV	Comma-Separated Values. Ett filformat som lagrar tabelldata i ren text och kan användas vid datautbyte mellan program som har olika filformatstandarder.
ER-diagram	Entity-Relationship-diagram. Används för att modellera relationer i en databas.
MD5	Message-Digest Algorithm. Hash-funktion som används för att kryptera data.
MVC	Model-View-Controller. Arkitekturmodell för att separera programlogik från presentation i ett mjukvarusystem.

Proprietär programvara	Programvara med restriktioner för användning och modifiering. Syftar på stängd källkod.
Ramverk	Syftar på mjukvaruramverk. En kollektion av bibliotek vars funktioner är definerade i ett API.
Scaffolding	Genererar grundkoden för ett objekt med CRUD-funktionalitet.
SHA1	US Secure Hash Algorithm 1. Hash-funktion som används för att kryptera data.
Skalbarhet	Hur väl systemet är anpassat för tillägg av ny funktionalitet eller ökning i datamängd.
SQL	Structured Query Language. Standardiserat språk för att hantera data i en relationsdatabas.
UML	Unified Modelling Language. Objektorienterat generellt språk för modellering av system i form av diagram.
XML	eXtensible Markup Language. Ett märkspråk som används för att utbyta data mellan system.

Programvarulicenser

BSD	Berkeley Software Distribution. Licens som kräver öppen källkod för programvara, men inte för programvarans modifierade versioner.
CDDL	Common Development and Distribution License. Innebär öppen källkod och är framtagen av Sun Microsystems.
GPL	GNU General Public License. Kräver öppen källkod för programmet såväl som programmets modifierade versioner.
MIT	Licens som tagits fram vid Massachusetts Institute of Technology. Licensen innebär öppen källkod och tillåter återanvändning av programvara i proprietär mjukvara.
OSL	Open Software License. Copyleft med en uppsägningsklausul i det fall att användaren ansöker om patent för programmet.
PHP	Licens med öppen källkod dock inte copyleft. Tillåter inte programvara att innehålla termen PHP i sin titel.

1. Inledning

På Chalmers arrangeras årligen Nordens största arbetsmarknadsmässa, CHARM, med över hundra deltagande företag. Varje år tillsätts en kommitté, CHARMk, som får i uppdrag att styra över mässan. För att kunna genomföra ett arrangemang av denna storlek behövs många volontärer, vilket engagerar studenter på hela Chalmers. Företagen som deltar i mässan blir tilldelade en student (värd) som skall vara företagets kontaktperson.

CHARMk har ett datoriserat internsystem för att administrera CHARM-mässor. Företagen anmäler sig till en mässa via den externa sidan arbetsmarknadsdagar.se och sedan importeras företagsinformationen till CHARMks internsystem. Alla studenter kan, via internsystemet, registrera sig och ansöka om att få bli värdar på en mässa. Medlemmarna i CHARMk är administratörer i systemet och har som uppgift att para ihop varje företag med en värd.

CHARMks nuvarande internsystem uppfyller i dagsläget inte de behov som efterfrågas. Systemet har många brister och är inte anpassat för vidareutveckling och underhåll; det är till exempel komplicerat att starta nya mässor, och tidskrävande att återställa systemet mellan varje mässa – saker som kräver onödigt manuellt arbete. Systemet saknar dessutom dokumentation, samt bygger på en teknik som få inom CHARMk besitter kunskap om.

1.1 Syfte

Det generella syftet med kandidatarbetet är att leverera ett väl genomtänkt och strukturerat internsystem till CHARMk. Systemet som efterfrågas är en webbaserad portal – med all information lagrad i en databas – byggd med en modern och användarvänlig teknik. Eftersom en kravspecifikation saknas skall en sådan inledningsvis tas fram tillsammans med CHARMk.

Fokus skall ligga på att konstruera ett system innehållande de mest grundläggande funktionerna. Det nya systemet skall bland annat kunna hantera värdansökningar från studenter, lagra information om företag, och ha funktioner som tillåter CHARMk att administrera ansökningar.

Systemet skall ha en flexibel struktur och vara lätt att i framtiden vidareutveckla med ny funktionalitet. Några exempel på sådan funktionalitet är bland annat administration av CHARM-samtal, autogenerering av fakturor samt inloggning för företag.

1.2 Problem/Uppgift

Först och främst skall en kravspecifikation upprättas tillsammans med CHARMk. Kravspecifikationen skall ge förståelse för vad som efterfrågas av ett nytt internsystem.

För att framtida utvecklare enkelt skall kunna felsöka och vidareutveckla systemet skall det noggrant modelleras och dokumenteras. Modelleringen beskriver hur systemet skall utformas och hur varje användare kommer att interagera med det.

Systemet skall kunna hantera tre olika aktörer: studenter, företag och CHARMk (administratörer). Aktörerna skall interagera med systemet på olika sätt, vilket kommer att resultera i olika användaraspekter.

För att kunna påbörja implementeringen av internsystemet behöver ett programmeringsspråk samt en databashanterare att väljas. Valet kommer att göras genom att kunskap om lämpliga språk inhämtas och utvärderas. Vid valet kommer hänsyn att tas till CHARMks önskemål och komfort.

Utifrån de jämförelser som görs i samband med kunskapsinhämtningen, dras sedan en slutsats om det mest optimala programmeringsspråket som internsystemet skall implementeras i.

Användaraspekter

Systemet har, som tidigare nämnt, tre olika aktörer, varför tre olika användaraspekter modellerats; student, företag och CHARMk. Nedan följer en kort beskrivning av dessa aspekter.

Student
En student skall kunna registrera sig i internsystemet och fylla i lämplig kontaktinformation. Vid en ny mässa får studenten tillgång till att kunna skicka in en värdansökan där ansökningstiden styrs av CHARMk. När ansökningen är inskickad behandlas den manuellt av CHARMk.

Studenten skall sedan kunna bli tilldelad en värdroll. Det finns tre olika värdroller: företagsvärd, områdesvärd och övrigt. Beroende på vilken värdroll studenten blir tilldelad, kan studenten följaktligen se och ändra olika delar i systemet. Exempelvis kan en företagsvärd se och redigera information om det tilldelade företaget.

Företag
All företagsinformation skall importeras från en fil av formatet CSV, till internsystemets databas. Filen kommer från den externa sidan arbetsmarknadsdagar.se. Om ett företag finns i internsystemet sedan tidigare skall informationen endast uppdateras; detta för att undvika dupliceringar.

CHARMk
Alla medlemmar i CHARMk har samma grundläggande funktionalitet. De skall kunna:

- Administrera värdansökningar
- Uppdatera ansökningsformuläret
- Tilldela studenter olika värdroller
- Tilldela värdar till företag
- Importera företag

Ett behörighetssystem behöver utvecklas för att skilja vilka funktioner varje CHARMk medlem har behörighet till i systemet.

Behörighetsnivåer

En viktig aspekt är behörigheten för olika grupper av användare. Ur säkerhetssynpunkt är det viktigt att övriga användare inte har samma rättigheter i systemet som administratörerna (CHARMk), utan endast de mest nödvändiga. Utan hänsyn till detta riskerar systemet exempelvis dataförluster och spridning av sekretessbelagt material.

1.3 Avgränsningar

Tidsbegränsningen på kandidatarbetet leder till att det inte är möjligt att uppnå en komplett produkt. Det huvudsakliga målet blir därför att designa en grundstomme till internsystemet, innefattande allt som krävs för implementering och vidareutveckling av produkten.

För att få ett flexibelt och hållbart system läggs fokus på analys och design. Tanken är att en erfaren programmerare skall kunna, med hjälp av dokumentation och design, vidareutveckla denna grundstomme till en färdig produkt.

Att fokus ligger på analys och design gör att implemeteringen av system kommer i andra hand och den kommer därför inte att vara så omfattande. De delar som skall implementeras är endast den högsta prioritetsnivån i kravspecifikationen (Se Bilaga A).

I valet av programmeringsspråk kommer endast de mest spridda och använda programmeringsspråken att hinna utvärderas.

1.4 Metod/Genomförande

Genomförandet av projektet delades in i tre större moment: kunskapsinhämtning, modellering av systemet och implementering. Nedan följer beskrivningar av dessa moment.

Kunskapsinhämtning

Genom möten har kontinuerlig kontakt hållits med CHARMk för att ta fram en kravspecifikation. Kravspecifikationen har legat som grund för förståelse av problemet.

Med hjälp av litteratur och vetenskapliga artiklar har kunskap inhämtats och dokumenterats om programmeringsspråk, ramverk och databashanterare som projektet kan implementeras med. Med utgångspunkt från dokumentationen gjordes en utvärdering för att kunna välja programmeringsspråk, ramverk och databashanterare.

Modellering

Med hjälp av information om hur en CHARM-mässa är strukturerad har en fullskalig modell över det nya systemet tagits fram. Strukturen har avsiktligt gjorts generell och under arbetet har fokus legat på: flexibilitet, säkerhet, underhåll, skalbarhet och effektivitet. Inledningsvis definierades vilken funktionalitet systemet skulle ha och vilka relationer som skulle finnas mellan de olika delarna.

När valet av programmeringsspråk var klart, behövde modellen anpassas till det valda språket för att enkelt kunna implementeras. Eftersom systemets struktur är viktig för att det skulle bli flexibelt och lätt att vidareutveckla, prioriterades modelleringen framför implementeringen.

Implementering

Efter att ha samlat kunskap och gjort jämförelser mellan olika programmeringsspråk och tillhörande ramverk, började modellen att implementeras med den valda tekniken. En detaljplanering gjordes för att dela upp arbetet och kunna prioritera olika delar av strukturen. Dessutom skulle en djupare kunskapsinhämtning inom den valda tekniken inkluderas.

Kravspecifikationen delades upp i olika delområden, där de högst prioriterade delarna implementerades. Exempel på delområden i kravspecifikationen var flexibla ansökningsformulär och hur importen av företagsinformation skulle ske.

För att efterleva ambitionen om att all kod och dokumentation skulle följa samma standard, arbetades en mall för detta fram.

Eftersom det är fler än en utvecklare i projektet var det lämpligt att ha någon form av versionshantering. Därför skulle möjligheterna till detta ses över samtidigt som kunskapsinhämtningen genomfördes. Då miljön i vilken systemet implementeras och valet av programmeringsspråk hörde ihop, valdes dessa i samband med varandra.

2. Teoretisk bakgrund

Nedan följer en sammanfattning av hur programmering mot webben sker och en sammanställning av några viktiga säkerhetsaspekter. Därefter följer en utvärdering av ett antal presumtiva programmeringsspråk och ramverk, samt databashanterare som är tänkbara för projektet. För att göra utvärderingen objektiv har denna skett enligt en mall framarbetad utefter de krav gruppen ansåg viktiga.

Kapitlet avslutas med en jämförelse och diskussion, vilka föranleder ett teoretiskt resultat och ett val som även inbegriper CHARMks krav och önskemål.

2.1 Webbutveckling

Webbprojekt går att skriva i de flesta programmeringsspråk men det finns ett antal språk som är specifikt inriktade mot webbutveckling och kommunikation med databashanterare. Ofta har dessa programmeringsspråk också tillhörande ramverk – detta för att underlätta implementering och bidra med vissa standardfunktioner, vanligtvis för att lösa återkommande problem.

För att effektivt kunna spara stora mängder data på Internet används databaser. En databas är information samlad på ett sådant sätt att det enkelt går att lagra, söka, hämta och ändra i informationen. För att kunna hantera databaser krävs en databashanterare [1].

För att kommunicera med en databas finns det ett speciellt standardiserat språk, SQL. Interaktion med databasen sker genom att ställa frågor i SQL [1]. Inom webbutveckling utvecklas databasstrukturen oftast inte i ren SQL, utan med hjälp av något verktyg som genererar SQL-koden.

2.2 Säkerhet

I utvecklingen av en säker och professionell applikation mot webben finns det ett antal olika säkerhetsaspekter som är kända och viktiga att ta hänsyn till. Nedan följer en kort teoretisk förklaring om några viktiga aspekter.

2.2.1 Injektioner

Det är viktigt att alltid vara kritisk mot datan som skickas från användarna till servern. Om den inskickade datan gör att systemet beter sig på ett annat sätt än vad utvecklaren avsett, kallas det för en injektion. Två vanliga injektioner är SQL-injektion och Cross Site Scripting [2].

SQL-injektion

Vid lagring av data där systemet tillåter användaren att skriva in valfri information löper systemet risk för SQL-injektion. Beroende på användarens information kan söksträngen manipuleras. Det här kan resultera i att en obehörig person får tillgång till systemet, manipulering av lagrad data eller radering av information i databasen [2].

Cross Site Scripting

Cross Site Scripting betyder att en användare injicerar ett script, Javascript eller HTML, i systemets kod. Det kan resultera i avslöjande av säker data och åtkomst till autentiseringsinformation [2].

2.2.2 Lösenord i klartext

Vid lagring av lösenord i en databas är ett vanligt misstag att utvecklaren väljer att spara lösenorden i klartext. Risken är att någon obehörig kan komma in i systemet, exempelvis genom SQL-injektion [2], och få tillgång till databasen och därmed tillgång till en lista över alla användarnamn med deras respektive lösenord. De användare som använder samma lösenord till andra tjänster löper då risk för intrång även på dessa.

En lösning på problemet är att kryptera lösenorden med en envägsfunktion [3]. De vanligaste envägsfunktionerna som används vid kryptering av lösenord är SHA1 [4] och MD5 [5]. Båda dessa funktioner går att dekryptera genom att skapa en tabell över lösenord och tillhörande krypteringsvärde, likt ett uppslagsverk. En sådan attack går dock att försvåra genom att lägga till en sträng till lösenordet som krypteras. Strängen kallas för salt och fungerar som en hemlig nyckel.

2.2.3 Sessioner

För att hålla reda på temporär information om besökare på webbplatser används sessioner. En session lagras oftast i en cookie lokalt på användarens dator och innehåller information om användarens tillstånd [6].

Tjänster som autentisering kan utnyttja sessioner för att hålla reda på om användaren har fått tillträde eller ej. En säkerhetsrisk ligger i hur och vad som lagras i en session. En manipulerad session kan till exempel ge obehöriga tillträde till säkrade sidor [6].

2.3 Språk och ramverk

Ramverk är skapade för att underlätta implementeringsarbetet för utvecklaren och därmed korta ner produktionstiden och motarbeta upprepning av kod. Detta sker genom att tillhandahålla utvecklaren med färdiga och välbeprövade funktioner. Användandet av ramverk hjälper också utvecklaren att följa MVC-modellen, vilket underlättar felsökning och ökar skalbarheten vid vidareutveckling.

Nedan följer en utvärdering av några av de mest använda programmeringsspråken och ramverken som är anpassade för webbutveckling. Varje programmeringsspråk med tillhörande ramverk har utvärderats med avseende på följande mall:

Mall för programmeringsspråk och ramverk

MVC:	Inbyggt stöd för MVC-modellen
Licens:	Vilken programvarulicens som bestämmer villkor för programvaran.
Databashanterare:	Kompatibla databaser
Multipla databaser:	Om det stödjer mer än en databas
Active Records / DB Objects:	Stöd för Active Records
Operativsystem:	Kompatibla operativsystem
Dokumenterade säkerhetshål:	Hjälpfunktioner eller dokumentation
CRUD:	Create, Read, Update och Delete funktion för databasobjekt. Autogenererat eller stöd.
Moduler:	Moduler för XML-läsning och CSV-hantering.
Versioner:	Kompatibla versioner (endast ramverk)
Autentisiering:	Modul för användarautentisiering
Testning:	Säkerhet- eller funktionstestning
Scaffolding:	Autogenererar kod (endast ramverk)
Organisation:	Språket/ramverkets spridning och hjälpmedel. Exempelvis community, forum och API.

2.3.1 ASP

ASP lanserades 2002 av Microsoft och har därefter uppdaterats regelbundet, vilket är en anledning till dess stora spridning. Det ramverk som används i samband med programmeringsspråket ASP kallas .NET. ASP är ett av de bästa programmeringsspråken när det gäller skalbarhet [7]. Vid installation medföljer ingen webbserver, dock finns detta inbyggt i Windows.

ASP.NET används på många webbsidor och av många stora företag. Några exempel på webbsidor som använder ASP.NET är sökmotorn bing.com och nyhetssidan msnbc.msn.com [8].

MVC:	Ja
Licens:	Proprietär programvara (Microsoft-licens)
Databashanterare:	Alla. Standard: Microsoft SQL Server
Multipla databaser:	Ja
Active Records / DB Objects:	Ja, med tillägg
Operativsystem:	Windows
Dokumenterade säkerhetshål:	Ja
CRUD:	Ja
Moduler (XML, CSV):	XML, CSV
Autentisering:	Ja
Testning:	Ja
Organisation:	Forum, community, API

2.3.2 JSP

JSP, Java Server Pages, är ett objektorienterat programmeringsspråk som lanserades 1999 av Sun Microsystems [9]. Till språket finns flera ramverk som exempelvis Grails och Spring Roo.

För att kompilera ett JSP-projekt behöver dels ett antal standardfiler skrivas, och dels en kompilator för JSP. För att underlätta kompileringen finns det servletmotorer som kan användas, exempelvis Apache Tomcat [9]. Att det behövs både vissa speciella filer och en speciell kompilator beror på att JSP-sidor görs om till Java Servlets när de kompileras. En Java Servlet används för att utöka funktionaliteten i en server genom att acceptera förfrågningar och generera svar. Detta medför både snabbhet och skalbarhet i systemet, eftersom projektet endast kompileras en gång [9].

Oracle.com är en av de största hemsidorna som använder JSP. [10]

MVC:	Ja
Licens:	CDDL
Databashanterare:	Alla
Multipla databaser:	Nej
Active Records / DB Objects:	Nej
Operativsystem:	Windows, Linux, Mac OS X
Dokumenterade säkerhetshål:	Ja
CRUD:	Ja, genom vissa ramverk
Moduler (XML, CSV):	XML
Autentisering:	Nej
Testning:	Nej
Organisation:	Community, API

2.3.3 PHP

PHP är ett skriptspråk som lanserades 1995. Ett skriptspråk innebär att det krävs en speciell miljö för att kunna exekvera koden. Språket är utvecklat av privatpersoner men underhålls idag av företaget Zend Technologies och är ett de mest använda språken för webbutveckling [11].

För att kompilera och köra PHP-koden krävs en webbserver, vilket inte medföljer vid installation. Till språket finns många ramverk där några av de mest använda är Yii, CodeIgniter, CakePHP, Symfony och Zend [11]. En utvärdering av ramverken finns i Bilaga B, Tabell B.2.

Några stora webbsidor som delvis är skrivna i PHP är exempelvis Facebook och Wikipedia [12].

MVC:	Ja
Licens:	PHP license
Databashanterare:	Alla
Multipla databaser:	Ja
Active Records / DB Objects:	Ja
Operativsystem:	Windows, Linux, UNIX, Mac OS X
Dokumenterade säkerhetshål:	Ja
CRUD:	Ja, genom vissa ramverk
Moduler (XML, CSV):	XML, CSV
Autentisering:	Nej
Testning:	Ja
Organisation:	Forum, community, bloggar

2.3.4 Django

Django är ett ramverk som är skrivet i programmeringsspråket Python och som lanserades 2005 av Django Software Foundation. Python tillåter multipla paradigmer: objektorienterad, imperativ och delvis funktionell programmering [13].

Django är framtaget för att utnyttja Pythons egenskaper i webbutveckling och konceptet DRY, Don't Repeat Yourself, eftersträvas [14]. DRY handlar om att använda generisk kod för att kunna återanvända kod.

Vid installationen av Django ingår en fristående webbserver för utveckling av mindre projekt och enkel testning. Med tilläggspaket blir Django kompatibelt med andra kända webbservrar.

Ramverket stödjer MVC-modellen. En modelklass representerar en databastabell och en instans av klassen representerar en specifik läsning i tabellen [14]. Detta innebär att utvecklaren inte arbetar direkt mot databasen utan endast mot modelklasser.

Hemsidan till applikationen Instagram är ett av många projekt som är skrivna i Django [15].

MVC:	Ja
Licens:	BSD
Databashanterare:	Alla. Standard: PostgreSQL
Multipla databaser:	Ja
Active Records / DB Objects:	Ja
Operativsystem:	Windows, Linux, UNIX, Mac OS X
Dokumenterade säkerhetshål:	Ja
CRUD:	Ja
Moduler (XML, CSV):	XML, CSV
Autentisering:	Ja
Testning:	Ja
Organisation:	Mailinglista, Wikipedia, API, konferenser [16]

2.3.5 Ruby on Rails

Ruby on Rails är ett ramverk till det objektorienterade programmeringsspråket Ruby [17]. Ramverket lanserades 2004 och är i första hand utvecklat av en privatperson - David Hansson - men idag bidrar över 2000 personer till utvecklingen [18].

Ruby on Rails använder principerna DRY samt COC, Convention over Configuration [18]. COC handlar om att använda konventioner, exempelvis på filnamn, för att låta ramverket sköta konfigurationen. Det innebär att filer med rätt namnkonvention kopplas till varandra med hjälp av ramverket.

Ruby on Rails kommer med en egen webbserver, WEBrick, och finns tillgänglig i Rubys pakethanterare, RubyGems [19]. En stor applikation som till en början var byggd i Ruby on Rails är Twitter.com [20].

MVC:	Ja
Licens:	MIT
Databashanterare:	Alla. Standard: SQLite
Multipla databaser:	Ja
Active Records / DB Objects:	Ja
Operativsystem:	Windows, Linux, UNIX, Mac OS X
Dokumenterade säkerhetshål:	Ja
CRUD:	Ja
Moduler (XML, CSV):	XML, CSV
Autentisering:	Nej
Testning:	Ja
Organisation:	Community, mailinglistor, Twitter, Wikipedia

2.4 Databashanterare

Det finns olika typer av databashanterare, som stödjer olika databasmodeller. Ett exempel på en databasmodell är den hierarkiska modellen som bygger på en trädstruktur. En sådan struktur är väldigt effektiv att söka i men mycket ineffektiv att ändra i [1].

Den numera dominerande databasmodellen är relationsdatabaser. Denna innebär att databasen byggs upp av tabeller med rader och kolumner som relaterar data till varandra. Denna struktur innebär att det går att definiera relationer mellan tabellerna, vilket gör att komplexa strukturer kan konstrueras [21]. En relationsdatabas är anpassad för att vara skalbar.

Nedan följer en utvärdering av några av de mest använda databashanterarna. Varje databashanterare har utvärderats med avseende på följande mall:

Mall för databashanterare

Typ:	Typ av databashanterare
Licens:	Vilken programvarulicens som bestämmer villkor för programvaran.
Operativsystem:	Kompatibla operativsystem
Programmeringsspråk:	Kompatibla programmeringsspråk
Organisation:	Databashanterarens spridning och hjälpmedel. Exempelvis community, forum och API.

2.4.1 DB2

DB2 lanserades 1983 av IBM [22]. IBM har lång erfarenhet av databaser och var även med och utvecklade språket SQL [23].

Typ:	Relationsdatabashanterare
Licens:	Proprietär programvara (EULA)
Operativsystem:	Windows, Linux, UNIX
Programmeringsspråk:	Alla
Organisation:	Support, installationsguider

2.4.2 Microsoft SQL Server

Microsoft SQL Server lanserades 1989 av Microsoft och används idag, förutom av företaget själva, av exempelvis SAS och Maersk [24]. Databashanteraren är även kompatibel med Microsoft Office [25].

Typ:	Relationsdatabashanterare
Licens:	Proprietär programvara (Microsoft-licens)
Operativsystem:	Windows
Programmeringsspråk:	Begränsat, .NET-ramverk måste användas
Organisation:	Support, guider

2.4.3 MySQL

MySQL lanserades 1995 av privatpersoner men ägs numera av företaget Oracle och är en av de mest använda databashanterarna i världen. Idag används databashanteraren av många av världens största och snabbt växande företag, exempelvis Google, NASA och YouTube [26].

Arkitekturen på databasen gör att den har hög prestanda och ger exempelvis snabba hämtningar och fulltextsökningar [1]. Det finns också ett tillhörande GUI-verktyg, phpMyAdmin, som förenklar hanteringen av databasens struktur och innehåll.

Typ:	Relationsdatabashanterare
Licens:	GPL
Operativsystem:	Alla
Programmeringsspråk:	Alla
Organisation:	Support, community, dokumentation, installationguider

2.4.4 Oracle

Databashanteraren Oracle lanserades 1979 av företaget Oracle [10]. Oracle tillhör en av världens mest använda databashanterare. [27]. Oracle har en stor och lättillgänglig support samt flera extra databasapplikationer som kan läggas till vid behov [10].

Typ:	Objektorienterad relationsdatabashanterare
Licens:	Proprietär programvara
Operativsystem:	Windows, Linux, UNIX
Programmeringsspråk:	Alla
Organisation:	Support, installationsguider

2.4.5 PostgreSQL

PostgreSQL lanserades 1997 av PostgreSQL Global Development Group och är idag använt av många stora företag, däribland Skype och Sun Microsystems [28].

Databashanteraren använder en speciell lagringsteknik kallad Multiversion Concurrency Control, MVCC. MVCC är en metod som gör att flera användare kan få tillgång till databasen samtidigt och snabbt kunna hantera stora mängder data. Det finns också ett antal olika GUI-verktyg tillgängliga [29].

Typ:	Objektorienterad relationsdatabashanterare
Licens:	BSD
Operativsystem:	Alla
Programmeringsspråk:	Alla
Organisation:	Support, Wikipedia, community, installationsguider

2.5 Jämförelse

Med utgångspunkt från de utvärderingar som presenterades ovan (kapitel 2.3 och 2.4) gjordes en jämförelse mellan de olika produkterna. Syftet var att fastställa vilka verktyg som är mest lämpliga att använda i implementeringen av systemet.

För att underlätta läsningen finns i Bilaga B en översikt av utvärderingarna. I Tabell B.1 finns en sammanställning av programmeringsspråken, i Tabell B.2 en sammanställning av PHP-ramverken och i Tabell B.3 en sammanställning av databashanterarna.

Nedan följer en diskussion avseende jämförelser mellan olika programmeringsspråk och tillhörande ramverk, samt jämförelser mellan olika databashanterare som resulterar i ett val.

Utvärdering av programmeringsspråk och ramverk

Guider och andra former av lättillgänglig support var en viktig aspekt vid valet av programmeringsspråk och ramverk eftersom gruppmedlemmarna hade bristande erfarenheter inom webbutveckling. Det skulle enkelt gå att sätta sig in i syntaxen och metoderna för implementeringen.

I urvalet gick JSP bort för att det saknar många projektrelevanta funktioner som de andra programmeringsspråken har, och det var komplicerat att komma igång med. Vi tycker att språket inte verkade populärt bland utvecklare och det framstår som föråldrat. Det verkade inte populärt eftersom guiderna enbart relaterar till Oracles sidor och det var svårt att hitta hemsidor som är utvecklade i språket.

ASP.NET är ett intressant ramverk med en fullbordad autentisieringsmodul. Dock krävs en Windows-server med Microsoft-licens, vilket kostar pengar. Att det kräver en betallicens har en naturlig konsekvens: att många privatpersoner väljer bort det. Guider och hjälp finns tillgängligt, men genom Microsoft istället för oberoende utvecklare – något som gruppen uppfattade som negativt.

Ramverket Django är inte så utbrett, och det är svårt att hitta användbara guider och hjälpmedel för att komma igång. Visst stöd finns att tillgå – men inte så mycket som projektgruppen behövde. Valet stod därför mellan PHP med ramverk och Ruby on Rails.

Ruby on Rails är ett nytt ramverk som snabbt har blivit populärt, vilket märks genom att det finns många artiklar och publiceringar om ramverket. Positivt är att Ruby är objektorienterat i grunden, vilket utnyttjas i Ruby on Rails. Organisationen kring Ruby on Rails är imponerande, och de hjälpmedel som finns för att komma igång är enkla och tydliga. Det finns flera guider för nybörjare.

Ramverken till PHP (Tabell B.2) imponerar också. Många av ramverken innehåller intressant funktionalitet och professionell dokumentation, trots att de ofta är framtagna av privatpersoner. Ramverken är enkla att komma igång med och skillnaden mellan dem är utvecklingsmetoderna.

En del ramverk uppmuntrar till mer autogenererad kod än andra. Det märks på komplicerade mappstrukturer, vilket gör det svårt att förstå hur filerna hänger ihop. Det ramverk som utmärker sig positivt är CodeIgniter som ger mycket frihet vid användning av hjälpfunktionerna. Ramverket stödjer MVC med en enkel mappstruktur och Active Records för databashantering. Yii, CakePHP, Symfony och Zend är de ramverk som ger möjlighet till färdig autentisiering, men de kräver en längre inlärningskurva. CodeIgniter är det som enklast går att komma igång med och bygga en applikation i.

Efter att PHP etablerat sig har det infört stöd för objektorienterad programmering men ej hunnit stabilisera sig, vilket vi tycker märks i både dokumentation och guider. Detta gjorde det svårt att komma igång och programmera objektorienterat.

Gruppen har efter jämförelsen dragit slutsatsen att systemet bör implenteras i ramverket Ruby on Rails. Det som var avgörande för slutsatsen är att Ruby on Rails är objektorienterat i grunden, populärt och med tydlig dokumentation samt relevanta hjälpfunktioner.

Utvärdering av databashanterare

Skillnaden mellan de olika databashanterarna är inte särskilt stora, vilket kan bero på att samtliga är relationsdatabashanterare. Skillnader som kan ses mellan databashanterarna är exempelvis att några av dem har proprietär programvara och begränsningar i kompatibilitet.

MySQL är den ledande databashanteraren bland privatpersoner medan Oracle är den ledande bland företag, som vi har uppfattat det. Den databashanterare som har flest publiceringar och diskussionsgrupper är MySQL.

Till MySQL finns ett grafiskt gränssnitt, phpMyAdmin, som underlättar hantering av databasen. Vi ansåg att detta vara viktigt, eftersom erfarenhet av databaser saknas sedan tidigare. PostgreSQL har också ett tillhörande grafiskt gränssnitt, men databashanteraren är inte lika populär som MySQL. De övriga databashanterarna var väldigt snarlika, men hade inget speciellt som vi tyckte utmärkte dem positivt. Valet föll därför på MySQL som är open source, en av de mest använda databashanterarna i världen och populärt bland privatpersoner – vilket underlättar för kommittéen som byts ut varje år.

Val efter CHARMks önskemål och krav

Vår teoretiska utvärdering visar att projektet bör implementeras i Ruby on Rails med MySQL som databashanterare. Däremot kan projektet inte fortlöpa utan att samspela med kunden, CHARMk.

CHARMk hyr servrar genom Chalmers Studentkår och administrerar inte dessa själva. Deras önskemål var därför att systemet skulle implementeras i ett programmeringsspråk som inte kräver omkonfiguration. På servern finns PHP men inte stöd för Ruby on Rails. Den databas som används på servern är MySQL.

Med CHARMks åsikt som uppdragsgivare blev slutsatsen att systemet skall implementeras i PHP med CodeIgniter som ramverk samt MySQL som databashanterare.

3. Arbetsmetod

Under projektets gång har individuellt arbete kombinerats med grupparbete. För att få vägledning i rätt riktning och sätta upp delmål har möten hållits varje vecka. Delmålen har handlat om att vissa kapitel skall vara färdigskrivna i rapporten eller att en del i implementeringen skall ha nått ett visst stadium.

Inom gruppen har en rapportansvarig och en projektansvarig tillsatts. Den projektansvariges uppgift har varit att sammankalla möten, följa upp beslut och se till att projektet går framåt.

Nedan följer beskrivning av processen från första mötet med CHARMk till den färdiga produkten. Detta innefattar att ta fram en kravspecifikation, en jämförelse av olika programmeringsspråk samt implementera utvalda delar av systemet. Under arbetets gång skall all kod dokumenteras och någon form av versionshantering måste finnas.

3.1 Förståelse av problemet

För att få förståelse för projektet och systemets krav upprättades en kravspecifikation, som arbetades fram genom möten och samtal med CHARMk. I kravspecifikationen beskrivs systemets struktur och funktionalitet i form av både krav på grundfunktionalitet samt funktioner för framtida vidareutveckling av projektet. Kravspecifikationen är uppdelad i olika prioritetskategorier baserat på projektets tidsbegränsningar samt CHARMks förväntningar på projektet.

3.2 Val av implementeringsmetod

Valet av implementeringsmetod skedde genom att utvärdera och jämföra de bäst lämpade programmeringsspråken.

Tillvägagångssättet för utvärderingarna var att genom individuellt arbete, där varje projektmedlem blev tilldelad två programmeringsspråk, inhämta kunskap och utvärdera informationen. Med hjälp av en utarbetad kunskapsmall kunde utvärderingen ske objektivt och korrekt.

I jämförelsen togs hänsyn såväl till CHARMks önskemål som till gruppmedlemmarnas uppfattning av programmeringsspråken.

3.3 Implementering

Ur kravspecifikationen kan oberoende delproblem urskiljas. Implementeringen av delproblemen har gjorts individuellt av gruppmedlemmarna för att sedan kopplas samman. Nedan följer de mest grundläggande delproblemen.

3.3.1 Autentisering

Ett krav är att studenter och medlemmarna i CHARMk skall kunna logga in och se personliga sidor. Detta kräver en autentisering där obehöriga inte kan ändra eller se information.
För att kunna implementera detta behövdes kunskap om säkerhet och sessioner samt vilka funktioner ramverket erbjuder.

3.3.2 Företagsimport

Information om företagen finns lagrad i en CSV-fil. För att implementera en företagsimport, som tar hänsyn till filens utseende, behövdes lämpliga mjukvarubibliotek studeras.

3.3.3 Ansökningsformuläret

Kravet på ett flexibelt ansökningsformulär innebär att CHARMk skall kunna ändra hela formulärets utseende utan större programmeringskunskaper, vilket skulle kunna lösas genom att använda ett XML-formulär. Huvudsakligen innebar arbetet i det här delproblemet att XML-formulär och relevant funktionalitet i det valda ramverket studerades för att kunna genomföra implementeringen.

3.4 Versionshantering

Ett vanligt problem med flera utvecklare på samma projekt är att fel som uppstår kan vara svåra att spåra. För att lösa detta behövdes någon form av versionshantering. Valet föll på GitHub, eftersom det har stor spridning och är enkelt att komma igång med. Med hjälp av GitHub sparas alla versioner av de uppladdade filerna och det går enkelt att se vilken kod som är ändrad mellan två versioner. Projektet har en central lagring, på GitHub, där användare kopierar ned filer lokalt för att modifieras. Flera utveckare kan då arbeta på samma fil utan att påverka varandra.

3.5 Dokumentation

CHARMk ställer stora krav på dokumentation eftersom en ny kommitée varje år skall använda och underhålla systemet. En bra dokumentation underlättar även vid vidareutveckling.

CodeIgniter erbjuder en mall på hur deras dokumentation ser ut. Denna mall kommer att användas för att dokumentera internsystemet och se till att all dokumentation följer samma standard. Varje utvecklare ansvarar för att dennes kod är dokumenterad.

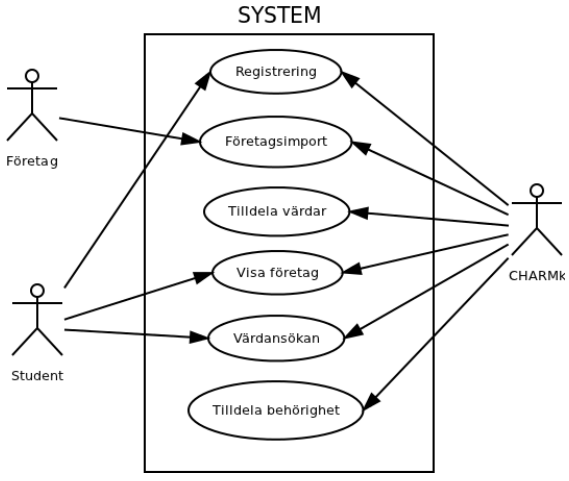
4. Modellering

För att förstå problemet och beskriva tänkbara lösningar modellerades systemet i flera steg, vilka beskrivs i detta kapitel. Även för flexibilitet och vidareutveckling var det viktigt med modeller som representerade tankesättet och idéerna bakom systemet. Dessa modeller skulle sedan fungera som en mall och underlätta vid implementeringen.

För att beskriva systemet så bra som möjligt gjordes: ett diagram över de användningsfall som finns, ett diagram för databasen samt ett klassdiagram.

4.1 Användningsfall

För att modellera hur internsystemet skall fungera i stort, skapades ett Use Case-diagram (se Figur C.1). Diagrammet representerar hur de olika användarna interagerar med systemet. Systemets funktioner är definierade, som ovaler, i boxen. CHARMk är administratörer och har därför tillgång till alla funktioner i systemet. Företagen använder inte systemet mer än att deras information importeras. Studenter har begränsad behörighet i systemet och kan därmed endast se begränsade delar.

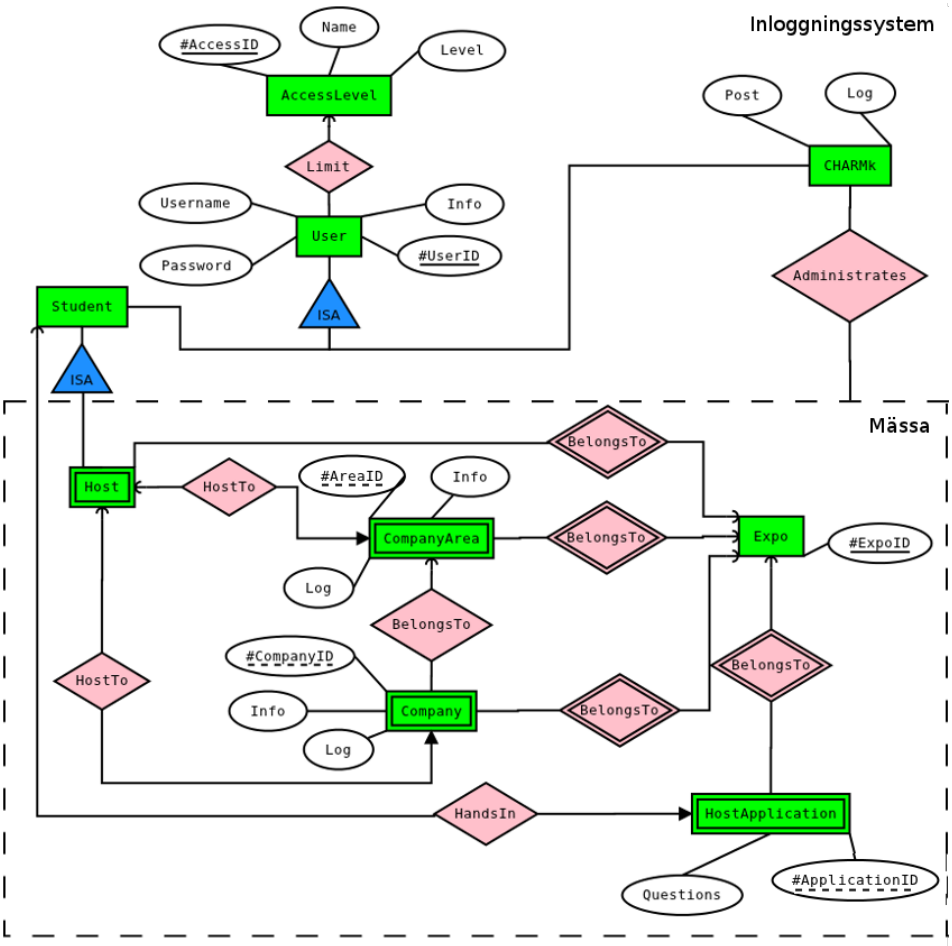


Figur 4.1: *Use Case-diagram över systemet*
För större figur, se Figur C.1

4.2 Databasen

För att kunna förstå databasstrukturen bättre och därmed underlätta implementeringen av internsystemet skapades ett Entity-Relationship-diagram, ER-diagram. ER-diagram används för att modellera relationer mellan delarna i en databas [30]. Därmed blir det enkelt att implementera de tabeller som behövs i databasen. Ett ER-diagram är uppbyggt av entiteter och relationer, vilka symboliseras i figuren genom gröna boxar respektive rosa diamanter (se Figur C.2).

Varje entitet representerar en egen tabell i databasen. En entitet kan ha flera egenskaper (ovaler i figuren) varav måste vara en nyckel; antingen en primär (understruken i figuren) eller en partiell (streckad i figuren). En nyckel krävs för att referera till ett objekt i tabellen. En entitet med partiell nyckel (en svag entitet) existerar endast i samband med en entitet med primär nyckel.



Figur 4.2: *ER-diagram av systemet*
För större figur, se Figur C.2

I systemet finns två databaser – en för att hantera inloggningssystemet och en för att hantera en mässa. Dessa beskrivs utförligare nedan.

Den stora streckade boxen representerar databasen för en mässa (se Figur C.2). Delarna i denna box (värdar/Host, företagsområden/CompanyArea, företag/Company och värdansökan/HostApplication) existerar endast i tillhörande mässa/Expo, vilket beskrivs med de inramade entiteterna som symboliserar svaga entiteter.

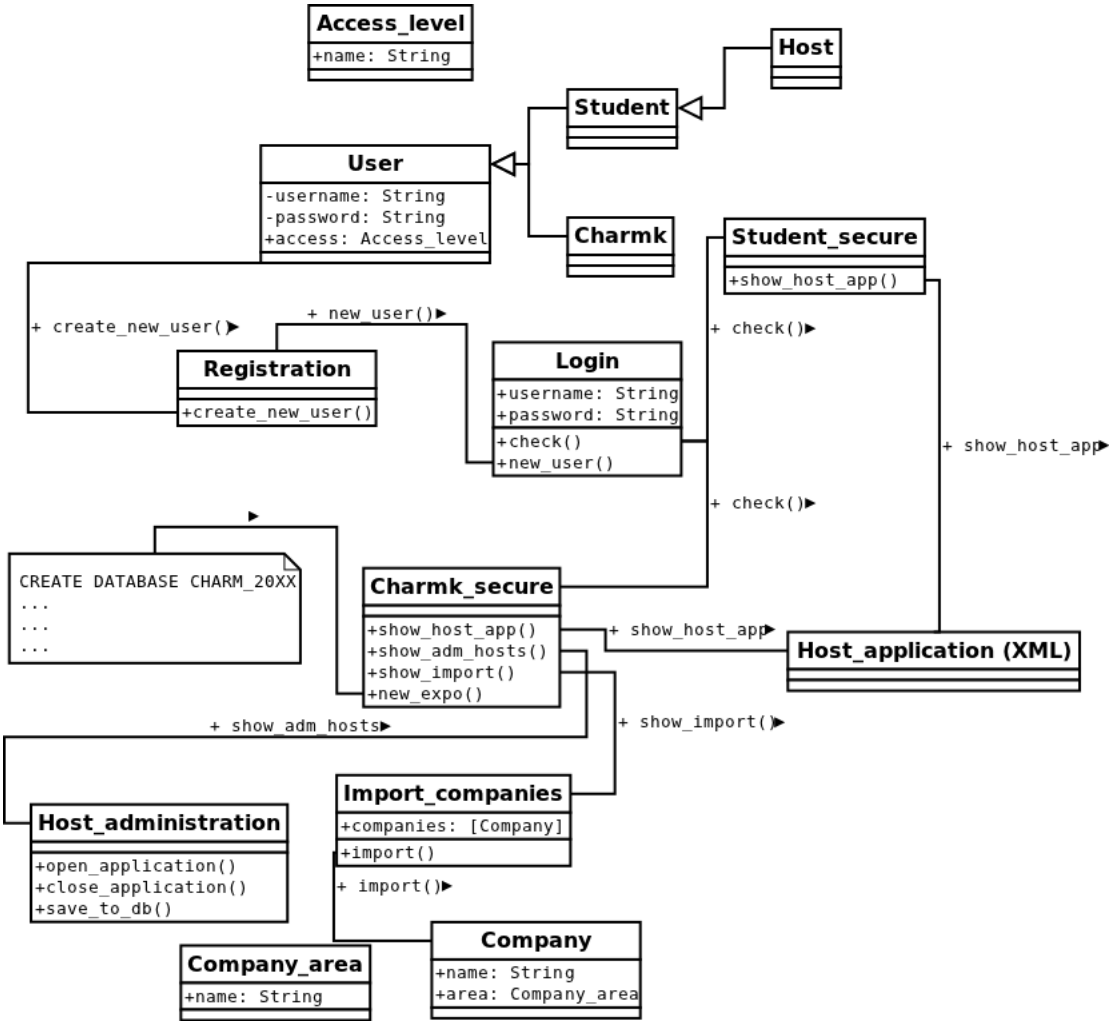
Studenter skall kunna skicka in en värdansökan som är specifik för en viss mässa. Om en student blir antagen som värd, kan denne vara värd till antingen ett företag eller ett företagsområde. Varje företag skall tillhöra ett företagsområde.

Inloggningssystemet har användare som antingen kan vara studenter eller medlemmar i CHARMk. Alla typer av användare är en form av User, vilket innebär att databasen lagrar samma typ av information för dem alla. Samtliga användare har också en behörighetsnivå för att reglera vilka sidor de skall få behörighet till.

4.3 Objektorienterad modellering

Systemet är uppdelat i klasser enligt objektorienterad PHP (se Figur 4.3). För att följa MVC-modellen representeras alla klasser av tre olika filer (Model, View och Controller). Genom denna uppdelning underlättas felsökning i systemet och det blir enkelt att vidareutveckla systemet genom att skapa nya klasser. Det är dock viktigt att tänka på att alla nya klasser måste följa MVC-modellen så att systemets uppbyggnad förblir enhetlig.

För att få tillgång till systemet registrerar användaren sig genom klassen *Registration*. Inloggning sker i klassen *Login* som beroende på användarens behörighet ger användaren tillgång till en av de säkra klasserna *Student_secure* eller *Charmk_secure*. Säkra klasser kan endast nås av inloggade användare. Klassen *Student_secure* ger tillgång till ansökningsformuläret medan *Charmk_secure* ger administratörsrättigheter och möjligheten att exempelvis importera företag.



Figur 4.3: *UML-diagram över PHP-klasser*

5. Implementering

Nedan följer beskrivningar av den utvecklingsmiljö och de metoder som har använts vid implementeringen. Kapitlet beskriver också hur de olika delarna i systemet är implementerade samt de säkerhetsaspekter som det tagits särskild hänsyn till.

5.1 Utvecklingsmiljö

Med hänvisning till slutsatsen i kapitel 2.5 valdes följande utvecklingsmiljö. All versionshantering för kod har skett genom GitHub.

Programmeringsspråk	PHP
Ramverk	CodeIgniter
Databashanterare	MySQL

5.1.1 Serverkonfiguration

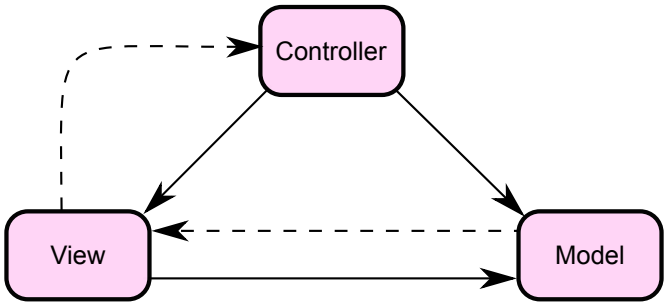
För att kunna exekvera koden krävdes en webbserver. Det konfiguerades en server som efterliknade CHARMks server enligt tabellen nedan.

Distribution:	Ubuntu Server 11.10
Webbserver:	Apache2 + phpMyAdmin
Databashanterare:	MySQL
FTP-server:	VsFTPd
Mail-server:	PostFix
Övrigt:	OpenSSH-server

5.2 MVC-modellen

Strukturen på internsystemet följer MVC-modellen som är en arkitekturmodell, vilken används för att separera programlogik från presentation i ett mjukvarusystem. MVC står för Model-View-Controller och representerar systemets uppdelning. Model är själva datan och den delen som kommunicerar med databasen. View är användargränssnittet.

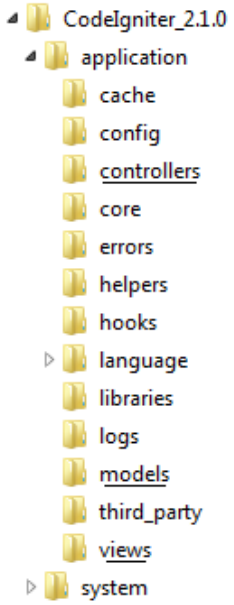
Länken mellan View och Model är Controller [31]. Detta illustreras i Figur 5.1[1] där de heldragna linjerna representerar direkta associationer och de streckade indirekta.



Figur 5.1: *Ett schematiskt diagram över MVC-modellen*

Genom att använda MVC-modellen blir systemet dels lättare att felsöka, och dels lättare för utomstående att sätta sig in i. Delkomponenterna i systemet är först indelade i klasser och dessa i sin tur uppdelade enligt MVC-modellen.

Figur 5.2 beskriver hur mappstrukturen ser ut i CodeIgniter. Delarna i MVC-modellen representeras av olika mappar (understrukna i figuren). Denna modell har följts vid implementeringen där de olika filerna läggs i rätt mapp för att MVC-modellen skall fungera.



Figur 5.2: *Mappstruktur i CodeIgniter*

[1]Figuren är hämtad från: http://en.wikipedia.org/wiki/File:ModelViewControllerDiagram2.svg

5.3 Inloggningssystem

Inloggningssystemet behöver vara säkert, samtidigt som det skall vara enkelt för studenter att registrera sig. CHARMk ställde som krav att det skall finnas olika behörighetsnivåer och att både studenter och medlemmarna i CHARMk skall kunna logga in.

Implementeringen består av två klasser: *Login* och *Registration*. Klassen *Login* har en View-fil innehållande formulär och referens till *Registration*. Nya användare skapas genom ett formulär i klassen *Registration* och användarinformationen lagras i databasen (se Figur 5.3). Användaren får också en behörighetsnivå (AccessID), beroende på om den är en student eller tillhör CHARMk.

UserID	Email	Password	FirstName	LastName	Institute	Registered	AccessID
4	coscar@student.chalmers.se	86f7e437faa5a7fce15d1ddcb9eaeaea377667b8	Oscar	Carlsson	KfKb	2012-05-03	2
0	anton@student.chalmers.se	86f7e437faa5a7fce15d1ddcb9eaeaea377667b8	Anton	Svensson	E	2012-05-06	2
6	castra@student.chalmers.se	86f7e437faa5a7fce15d1ddcb9eaeaea377667b8	Caroline	Strandberg	D	2012-05-07	2
8	max@charm.chalmers.se	86f7e437faa5a7fce15d1ddcb9eaeaea377667b8	Max	Sikström	F	2012-05-12	1
9	mirac@student.chalmers.se	86f7e437faa5a7fce15d1ddcb9eaeaea377667b8	Mirac	Günes	D	2012-05-13	2

Figur 5.3: *Användartabellen i databasen*

Vid registreringen kontrolleras att den inmatade mailadressen har ändelsen "@student.chalmers.se" eller "@charm.chalmers.se". Sedan skickar systemet ut ett mail som bekräftar för mailadressens ägare att den har blivit medlem på sidan. Användaren kan direkt efter registreringen logga in med den angivna mailadressen och tillhörande lösenord.

Till inloggningssystemet finns endast en Model-fil som har hand om all databashantering. För lagring i databasen används ramverkets Active Records-funktion. Ramverket garanterar SQL-säkrade söksträngar genom att byta ut specialtecken [32].

Användarnas lösenord lagras som SHA1-hashvärden. De räknas ut med hjälp av ramverkets krypteringsfunktion, vilken är lämplig för lagring av lösenord [32].

För att särskilja behöriga och obehöriga på de säkra sidorna används sessioner. All sessionsdata lagras i krypterat i cookies och lagras även i databasen (se Figur 5.4) för validering [32].

session_id	ip_address	user_agent	last_activity	user_data
262cbe58ca86e4489759ffad55d348f1	127.0.0.1	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.19...	1336400237	
e5e553b1eaafb991f6807d1343eddd2f	127.0.0.1	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.19...	1336322241	
ebe6bb7d0a147a12f4624a82299f7810	127.0.0.1	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.19...	1336375651	

Figur 5.4: *Sessionstabellen i databasen*

5.4 Företagsimport

Företagen anmäler sig till en mässa via arbetsmarknadsdagar.se, där företagsinformationen finns tillgänglig i form av en CSV-fil. Informationen importeras sedan till internsystemet.

View-filen representerar ett uppladdningsformulär, som endast accepterar filer av typen CSV. När användaren laddat upp CSV-filen lagras den temporärt på servern innan uppdateringen av databasen sker.

I en CSV-fil (se Figur 5.5) representerar varje rad ett företag med dess information, där olika informationdelar separeras med kommatecken. Rubrikera för företagsinformationen representeras av den första raden i filen. Exempel på rubriker är: organisationsnamn, organisationsnummer och telefonnummer. Med den färdiga hjälpklassen *CSVReader* [33] läses all data in och lagras i en Array-struktur.

```
1 Serienr,SID,Tid,Draft,Orgnamn,Orgnr,Orgtyp,Telnr,Kommentar,Stolar,Bildskarm
2 1,5,13:15,1,Foretag1 AB,012431,Ideell,0716123234,Tom,5,Typ 5
3 2,4,13:15,1,Foretag2 HB,0121235,Foretag,0716123234,Tom,4,Typ 1
4
```

Figur 5.5: *Exempel på en CSV-fil där SID är unikt ID*

I Model-filen uppdateras databasen (se Figur 5.6) med informationen från Array-strukturen. Om ett företag redan finns i databasen kompletteras endast informationen, annars läggs företaget till. Detta för att undvika dupliceringar av företag.

Serienummer	SID	Tid	Draft	Organisationsnamn	Organisationsnummer	Typ_av_organisation	Telefonnummer	Kommentar	Antal_barstolar	Bildskarm	FID
1	0	13:15	1	Foretag1 AB	012431	Ideell	716123234	Tom	5	Typ 5	8
2	1	13:15	1	Foretag2 HB	0121235	Foretag	716123234	Tom	4	Typ 1	9

Figur 5.6: *Företagstabellen i databasen*

5.5 Ansökningsformulär

Vid implementeringen av ansökningsformuläret var det viktigt att formuläret blev så flexibelt som möjligt. CHARMk skall enkelt kunna uppdatera formuläret inför varje ny mässa.

Ansökningsformuläret implementerades därför i ett XML-dokument innehållande alla frågor, samt vilken typ av HTML-formulär som skall vara kopplad till vilken fråga (se Figur 5.7 och Figur 5.8). Formuläret är flexibilet i den mån att frågor samt HTML-formulären defineras i XML-filen, och inte i systemets statiska kod.

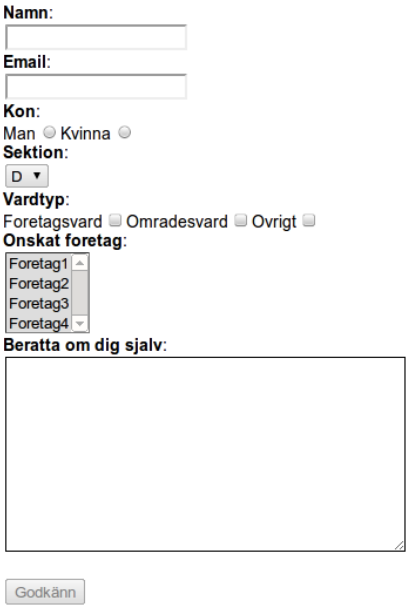
```
<question>
  <name>Vardtyp</name>
  <category>1</category>
  <option>Foretagsvard</option>
  <option>Omradesvard</option>
  <option>Ovrigt</option>
</question>
```

Figur 5.7: *Exempel på en fråga i XML-filen*

```
<category id="1">
  <type>Checkbox</type>
  <dbtype>VARCHAR(50)</dbtype>
</category>
```

Figur 5.8: *Exempel på en kategori i XML-filen*

Systemet översätter endast XML-filen till dess HTML-representation (se Figur 5.9).



Figur 5.9: *Representation av XML-fil*

När CHARMk öppnar värdansökan skapas en databastabell (se Figur 5.10) där varje fält representerar en fråga från XML-filen. När värdansökan är öppen kommer studenter att få möjlighet att skicka in ansökningar.

id	UserID	Name	Email	Kon	Sektion	Vardtyp	Onskat_foretag	Beratta_om_dig_sjalv
1	0	Anton Svensson	anton@student.chalmers.se	Man	D	Foretagsvard	Foretag1	Jobbade 2009, 2008, 2007.
3	6	Caroline Strandberg	castra@student.chalmers.se	Kvinna	D	Ovrigt	Foretag3	IT-Carro här!
5	4	Oscar Carlsson	coscar@student.chalmers.se	Man	D	Foretagsvard	Foretag3	Hej

Figur 5.10: *Tabell i databasen över alla ansökningssvar*

När en ansökan har skickats in läggs studenten även till i en annan databastabell för att kunna bli tilldelad en värdroll.

5.6 Värdtilldelning

Värdtilldelningen representerar en databastabell (se Figur 5.11) som lagrar alla studenter som har ansökt om att bli värdar för en mässa. Varje ansökan som läggs till defineras med startvärdena: status *Väntande* och värdtyp *Ej tilldelad*.

UserID	status	host_type
0	Vantande	Ej_tilldelad
4	Vantande	Ej_tilldelad
6	Antagen	Foretagsvard

Figur 5.11: *Tabell i databasen över alla tilldelningar*

De förutbestämda värdena för status är: *Väntande*, *Antagen*, *Ej Antagen* samt *Svartlistad*. De olika värdtyperna är: *Företagsvärd*, *Områdesvärd* och *Övrigt*.

View-filen representerar databastabellen grafiskt (se Figur 5.12) genom att visa studentens namn, nuvarande status och värdtyp för varje inskickad ansökan.

En tilldelning sker genom att CHARMk markerar en användare, väljer en ny status och eventuell värdtyp, samt klickar på knappen Tilldela.

Figur 5.12: *Värdtilldelningen som CHARMk ser*

6. Resultat

Nedan presenteras både de resultat som modelleringen och implementeringen givit. Implementeringen gjordes i CodeIgniter med en MySQL-databas, och resulterade i den slutgiltiga produkten. Resultatet beskrivs också i en slutgiltig modellering, som återkopplar till den ursprungliga modelleringen (kapitel 4). Även hur väl kraven från kravspecifikationen har uppfyllts presenteras här.

6.1 Slutgiltiga produkten

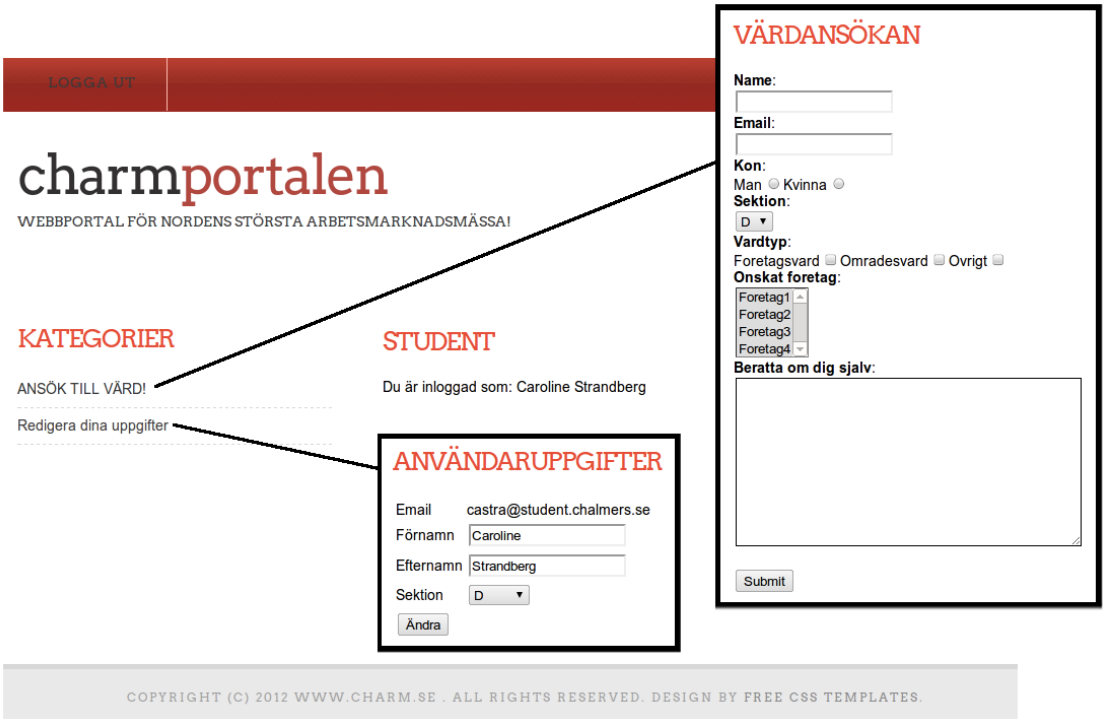
Projektet resulterade i en webbportal kopplad till en databas. Portalen har ett inloggningssystem (se Figur 6.1) där registrerade studenter och medlemmarna i CHARMk kan logga in och förflyttas till olika sidor, för att interagera med systemet på olika sätt.

Vid registrering ifylls följande information: förnamn, efternamn, institution, mailadress samt lösenord. Den angivna mailadressen kontrolleras mot databasen för att säkerställa att varje användare har ett unikt användarnamn. Om registreringen går igenom skickas ett bekräftelsemail ut.

Figur 6.1: *Inloggningssidan till portalen samt registrering*

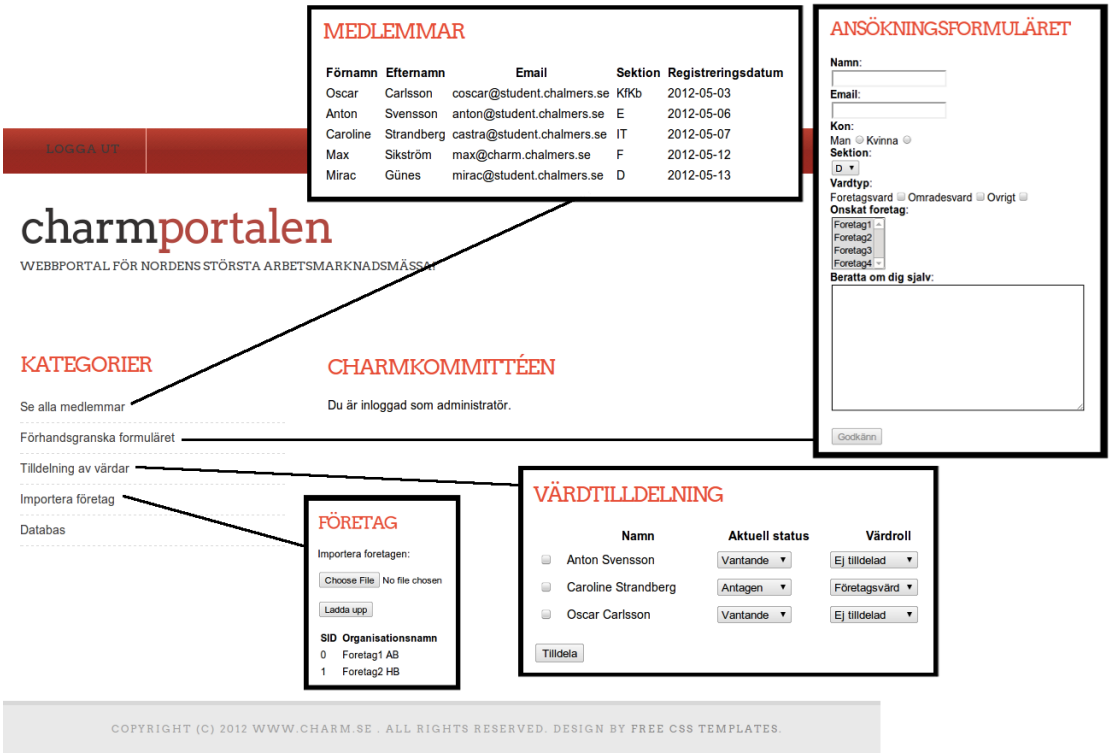
Studenten kommer efter inloggning till en sida där det förutom en länk till värdansökan även finns ett formulär för uppdatering av studentens information (se Figur 6.2).

Värdansökan består av olika frågor som är sammankopplade med varsitt lämpligt svarsformulär. De olika HTML-formulären som finns är följande: input, textarea, checkbox, radio button, dropdown list och selection list. Samtliga formulärstyper finns representerade i Figur 6.2



Figur 6.2: *Studenternas startsida samt användaruppgifter och värdansökan*

CHARMks egna sida innehåller länkar till företagsimport, tilldelning av värdar, godkännande och öppnande av värdansökningsformuläret, samt en lista över alla registrerade användare (se Figur 6.3).



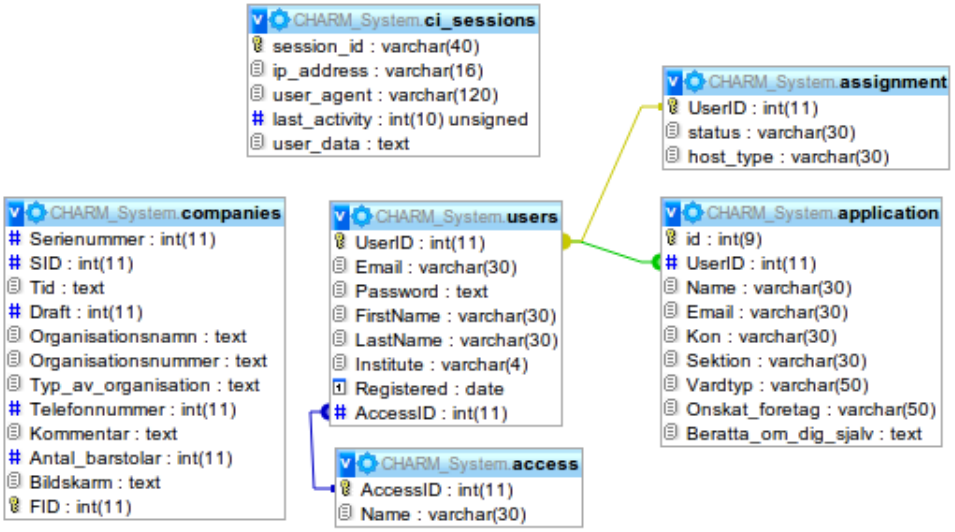
Figur 6.3: *CHARMks startsida med samtlig funktionalitet*

All kod för systemet finns publikt på: https://github.com/Kandidatgrupp17/.

6.2 Slutgiltig modellering

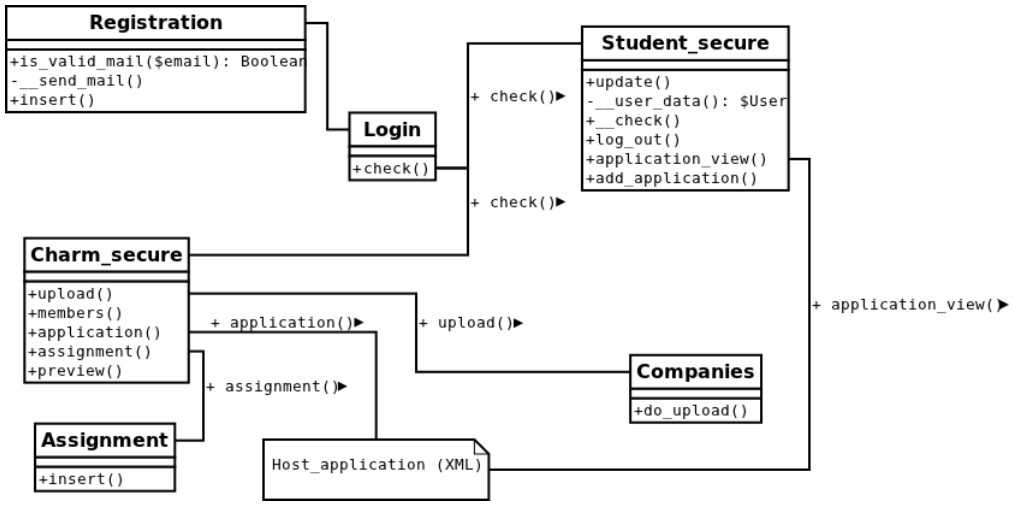
Den ursprungliga modelleringen, framtagen utifrån det genomförda förarbetet, har beskrivits och redovisats i kapitel 4. Den faktiska implementeringen krävde dock ett fåtal smärre modifikationer av modellen, vilka beskrivs nedan. En schematisk illustration över den slutgiltiga databasen återfinns i Figur 6.4, att jämföra med den ursprungligt planerade i Figur C.2.

I den faktiska implementeringen har det tillkommit en databastabell för hantering av sessioner och strukturen består enbart av en databas, istället för två databaser (se Figur C.2).



Figur 6.4: *Databasens struktur*

Det implementerade systemets klasser representeras i Figur 6.5. Allt eftersom gruppen anammade mer kunskap om webbutveckling kunde förbättringar göras vid implementeringen. Den ursprungliga modelleringen innehöll klasser som istället bättre lämpades som databastabeller.



Figur 6.5: *UML-diagram över PHP-klasser i vårt system*

6.3 Uppfyllda/icke uppfyllda krav

Arbetet har resulterat i att de största och viktigaste delarna i systemet är implementerade. Systemet har också anpassat för både vidareutveckling och underhåll genom dokumenterad kod. Vid implementering har hänsyn tagits till angivna säkerhetsaspekter och även användarvänlighet, det senare genom bland annat ett flexibelt ansökningsformulär.

Krav som uppfylldes efter implementeringen:

- Krav #1 Studenter skall kunna logga in
- Krav #2 Medlemmarna i CHARMk skall kunna logga in
- Krav #3 Importering av företag
- Krav #4 Tilldela student en värdroll
- Krav #5 Behörighetsnivåer

Krav som inte hann implementeras:

- Krav #6 Tilldela värdar till företag och företagsområden

7. Diskussion

Här diskuterar vi problem som uppstått och val vi gjort som har påverkat projektets resultat. Diskussionen kring jämförelsen finns utförlig i kapitel 2.5, med en kort sammanfattning nedan. De delar vi diskuterar är modellering, implementering och vår arbetsgång. Kapitlet avslutas med tankar om hur vårt arbete kan utmynna i ett nytt kandidatarbete och vidareutvecklas.

7.1 Jämförelsen av språk

Efter att vi hade påbörjat språkjämförelsen framkom det att det från kunden fanns outalade restriktoner angående vilket programmeringsspråk internsystemet kunde implementeras i. Om dessa varit kända från början hade vi kunnat begränsa utvärderingen till ramverk för de möjliga språken. Ruby on Rails, Django och JSP är inte möjliga att implementera i, varför istället mer utförliga utvärderingar borde ha gjorts av ramverk till PHP, då de var tänkbara för implementering. Under utvärderingen fick vi däremot bred kunskap inom olika sorters webbutveckling samt inblick i fördelar och nackdelar med olika språk. Utvärderingen var alltså lärorik och kunskapen användbar vid implementeringen.

7.2 Hur gick modelleringen?

Under modelleringsprocessen ändrades modellen av vårt system flera gånger, allt eftersom vi inhämtade ny kunskap. Det hade givetvis varit en stor fördel och även sparat tid om gruppen hade haft kunskap om databaser och modelleringsspråk vid projektets start. Däremot har vi fått möjligheten att studera helt nya områden. Mycket av materialet vi använt i modelleringsstadiet kommer från en databaskurs på Chalmers och kan med all säkerhet användas framgångsrikt i framtida arbeten.

När väl implementeringen påbörjades lärde vi oss nya fördelar med CodeIgniter, vilket gjorde att resultatet skiljer sig från vår ursprungliga modelleringsidé. De modeller som omnämns i kapitel 4 bör därför ses som en utgångspunkt för systemet och det som lagt grund till den slutgiltiga modelleringen samt det implementerade resultatet.

Att resultatet och modelleringen skiljer sig ser vi som en naturlig del av projektet. Modelleringen var lika mycket till för att underlätta vid implementering som för att förstå problemet och se svårigheter. Somliga klasser i modelleringen resulterade i databastabeller och istället för två databaser implementerades systemet med en. Framtida arbeten kan utnyttja både vårt resultat och våra modeller för att komplettera internsystemet.

7.3 Hur gick implementeringen?

Stora delar av implementeringen tog längre tid än väntat. Vi hade inte förstått hur omfattande den var, och förutsåg inte de svårigheter som uppkom. Detta tillsammans med att det tog längre tid än planerat att sätta sig in i programmeringsspråket med tillhörande ramverk, innebar att endast de mest grundläggande delarna av systemet hann implementeras. Konsekvensen av det här blev att krav #6 inte hann implementeras.

För att exekvera den implementerade koden installerade varje gruppmedlem varsin lokal webbserver på sin dator. Vi antog att institutionen skulle tillhandahålla en gemensam webbserver, men så var inte fallet. Tid som avsatts till implementering fick istället gå till att konfigurera en egen server, vilket inte var något vi tagit hänsyn till i första tidsplaneringen. Däremot fick vi kunskap om servrar och deras mjukvarukonfiguration, vilket bidrog till bättre förståelse av webbutveckling.

Vid utvärderingen av ramverk insåg vi inte fördelarna med funktionen Scaffolding, som autogenererar kod. Vi uppfattade det som något för mer inbitna programmerare och att det snarare skulle förvirra oss i vår egen kod. Efter att ha kommit igång med programmeringen kunde vi se en upprepning av kod för olika klasser, där autogenerering hade varit smidigt och sparat tid för projektet – vilket är något vi borde ha vägt in i valet av ramverk. Å andra sidan var det väldigt lärorikt att få skriva mer kod själva och det är först efter implementeringen vi ser fördelarna med Scaffolding.

Valet att använda MVC-modellen vid implementering var nästintill självklar på grund av dess stora spridning inom webbutveckling. Det har tagit tid att vänja sig vid uppdelning av kod och det tankesätt som krävs, men i efterhand har modellen visat sig ha förvånansvärt många fördelar. En aspekt som underlättades särskilt mycket var felsökning.

Objektorienterad webbutveckling i PHP är nytt, och det finns spridda åsikter om hur det skall tillämpas. Det har varit svårt att hitta en metod för implementeringen som definerar hur klasser skall användas och vad som särskiljer komponenter i MVC-modellen. Detta är något som hade undvikits genom att använda Ruby on Rails, som har stöd för objektorientering sedan sin lansering.

Eftersom vi saknade kunskap om CodeIgniters alla fördelar, är delar av implementeringen onödigt komplicerade. I efterhand känns det som att vi har fastnat i nybörjarfel, vilket är naturligt vid första projektet i ett nytt programmeringsspråk, men vi har lärt oss av våra misstag och utvecklats under projektet. I slutfasen av implementeringen kunde vi dock börja optimera vår kod, och i större grad utnyttja CodeIgniters funktioner.

7.4 Hur väl har arbetsgången fungerat?

De möten som hafts varje vecka, både med handledare och inom gruppen, har varit ett bra sätt att följa upp de mål som kontinuerligt satts och gemensamt kunna lösa problem som uppstått under projektet.

När årets CHARM-mässa inföll utnyttjade vi tiden till att studera hur CHARMk organiserar och arbetar under en mässa, samt vilka delar av det dåvarande systemet som faktiskt användes. Däremot innebar mässan att CHARMk till och från var otillgängliga och flera av de planerade möten uteblev.

Beskrivningen av kandidatarbetet var otydlig och trots att CHARMk hade åsikter om projektet framgick det inte tydligt ur vår kommunikation. Bättre kommunikation med CHARMk hade exempelvis kunnat underlätta vid valet av programmeringsspråk.

I slutet av projektet klargjorde CHARMk att ett komplett internsystem behöver vara implementerat till sommaren. Detta på grund av att det utgående system är ohållbart och att licensen inte skulle förnyas. Vi poängterade att detta var något vi inte kunde uppfylla, vilket gjorde att kraven på projektet minskade och vi inte längre behöver leverera en produkt. Däremot är CHARMk intresserade av det resultat och de slutsatser vårt arbete har givit.

Något som definitivt kunde förbättrats under projektets gång var rapportskrivandet; om rapporten hade skrivits mer löpande hade arbetsbelastningen i slutet av projektet inte varit lika oproportionerligt hög. Tidsplaneringen borde gjorts mer realistisk genom att räkna in förberedelser inför redovisningar och handledningstillfällen. Projektet och rapportsskrivandet har trots detta fungerat bra och känts mycket givande.

7.5 Lärdomar och framtida arbeten

Vi har fått erfarenhet av hur det är att arbeta mot en uppdragsgivare och fått översätta idéer och krav till tekniska lösningar. Projektet har medvetet utformats för att efterlikna ett modernt webbprojekt i industrin. Nästan all utveckling mot webben sker idag med hjälp av ramverk och den erfarenheten vi har fått från CodeIgniter kan överföras till andra ramverk. Under projektet har vi fått läras oss om serverkonfiguration mot en Linuxdistribution.

De lärdomar vi har dragit från att använda GitHub som versionshanterare är inte bara användbara vid webbutveckling. GitHub används vid många programmeringsprojekt, såväl i industrin som vid hobbyprojekt.

Projektet har lagt grund för CHARMks nya internsystem. En möjlighet är att ett nytt kandidatarbete kan utnyttja det vi påbörjat för att slutföra produkten. Det skulle innebära en vidareutveckling av det system som vi påbörjat, och kunna så småningom resultera i ett komplett internsystem.

8. Slutsats

Utifrån de uppnådda resultaten och hur arbetet har gått kan vi dra följande slutsatser:

- Alla icke-funktionella krav samt följande funktionella krav har uppfyllts: #1, #2, #3, #4, #5.
- Produkten kräver vidareutveckling för att sättas i drift.
- Språkjämförelsen resulterade i att internsystemet för CHARMk skulle implementeras i Ruby on Rails. Men när CHARMks åsikt vägdes in i valet var detta inte längre en möjlighet och det bestämdes istället att systemet skall implementeras i PHP med CodeIgniter och MySQL.
- Om gruppen hade haft mer förkunskap eller varit mer insatt i webbutveckling kunde språkjämförelsen ha komprimerats och gett mer tid till implementering. Språkjämförelsen gav däremot en bra förståelse för projektet.
- Inlärningströskeln för CodeIgniter var högre än beräknat. Vi borde ha räknat med att det skulle ta längre tid att lära sig ramverket.
- Bra kommunikation med CHARMk krävdes för att få förståelse för projektet. Ifall kunden inte har en klar idé om vad som efterfrågas gäller det att kunna hantera kundkontakt på ett professionellt sätt.

Litteraturförteckning

[1] Sheldon R, Moes G. Beginning MySQL. Indianapolis: Wiley Publishing; 2005.

[2] Johns M, Beyerlein C. SMask: preventing injection attacks in web applications by approximating automatic data/code separation. SAC '07 Proceedings of the 2007 ACM symposium on Applied computing. 2007;p. 284–291. http://dl.acm.org/citation.cfm?id=1244071 (2012-05-04).

[3] Malpani R, Ilac C, Dutta T, Schutz K. Communicating a password securely; 2008. http://appft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.html&r=1&f=G&l=50&d=PG01&s1=12038815 (2012-05-04).

[4] Eastlake D, Jones P. Us secure hash Algorithm 1 (sha1), RFC 3174; 2001.

[5] Rivest R. The md5 message-digest algorithm, RFC 1321; 1992.

[6] Liu AX, Kovacs JM, Gouda MG. Secure cookie scheme. Computer Networks. 2012;56(6):1723–1730. http://www.sciencedirect.com/science/article/pii/S1389128612000370 (2012-05-04).

[7] Volodarsky M. ASP.NET: Fast, Scalable, and Secure Session State Management; 2012. http://msdn.microsoft.com/en-us/magazine/cc163730.aspx#S7 (2012-03-19).

[8] Microsoft ASP NET Team. Home: Official Microsoft Site; 2012. http://www.asp.net/ (2012-03-19).

[9] Zambon G, Sekler M. Beginning JSP, JSF and Tomcat web development: from novice to professional. New York: Apress; 2007.

[10] Oracle Corporation. Database 11g, Oracle Database 11g, Oracle; 2012. http://www.oracle.com/us/products/database/ (2012-03-05).

[11] The PHP Group. PHP: Hypertext Preprocessor; 2012. http://www.php.net/ (2012-03-19).

[12] Shire B. PHP and Facebook; 2007. `http://blog.facebook.com/blog.php?post=2356432130` (2012-03-19).

[13] Python Software Foundation. Python Programming Language - Official Website; 2012. `http://www.python.org/` (2012-03-19).

[14] Django Software Foundation. Django | The Web framework for perfectionists with deadlines; 2012. `https://www.djangoproject.com/` (2012-03-19).

[15] Instagram Engineering; 2012. `http://instagram-engineering.tumblr.com/` (2012-03-19).

[16] DjangoCon Europe - Home; 2012. `http://2012.djangocon.eu/` (2012-03-19).

[17] Thomas D, Hansson DH. Agile Web Development with Rails. Raleigh, N.C.: Pragmatic Bookshelf; 2005.

[18] Heinemeier-Hansson D. Ruby on Rails; 2012. `http://rubyonrails.org/` (2012-03-19).

[19] rubygems.com: The Leading Ruby Gem Site on the Net; 2012. `http://www.rubygems.com/` (2012-03-19).

[20] Twitter, Inc . Twitter Engineering; 2011. `http://engineering.twitter.com/` (2012-03-19).

[21] Garcia-Molina H, Ullman JD, Widom J. Database systems: the complete book. 2nd edition. New Jersey: Pearson Prentice Hall; 2009.

[22] IBM. IBM - DB2 database software; 2012. `http://www-01.ibm.com/software/data/db2/` (2012-03-05).

[23] Allen G. Beginning DB2: from novice to professional. New York: Apress; 2008.

[24] Dewson R. Beginning SQL Server 2008 for developers: from novice to professional. New York: Apress; 2008.

[25] Microsoft Corporation. Database Management, Data Mining and Warehousing, Microsoft SQL Server; 2012. `http://www.microsoft.com/sqlserver/en/us/` (2012-03-05).

[26] Oracle Corporation. MySQL:: The world's most popular open source database; 2012. `www.mysql.com` (2012-03-05).

[27] De Haan L. Beginning Oracle SQL. New York: Apress; 2009.

[28] PostgreSQL Global Development Group. PostgreSQL; 2012. `http://www.postgresql.org` (2012-03-05).

[29] Matthew N, Stones R. Beginning databases with PostgreSQL: from novice to professional. New York: Apress; 2005.

[30] Broberg N. Databases (VT2012); 2012. `http://www.cse.chalmers.se/edu/course/TDA357/` (2012-05-09).

[31] EllisLab, Inc . Model-View-Controller: CodeIgniter User Guide; 2012. `http://codeigniter.com/user_guide/overview/mvc.html` (2012-03-05).

[32] EllisLab, Inc . Welcome to CodeIgniter: CodeIgniter User Guide; 2012. `http://codeigniter.com/user_guide/` (2012-05-04).

[33] Wiki - CodeIgniter; 2009. `http://codeigniter.com/wiki/CSVReader/` (2012-05-04).

[34] Zend Technologies Ltd . Zend - The PHP Company - Zend.com; 2010. `http://www.zend.com/en/company/` (2012-03-19).

[35] EllisLab, Inc . EllisLab - Where Ideas Hatch!; 2012. `http://ellislab.com/` (2012-03-19).

[36] Yii Software LLC . Yii Framework: Bet for Web 2.0 Development; 2012. `http://www.yiiframework.com/` (2012-03-19).

[37] Cake Software Foundation. Cake Software Foundation, Home; 2012. `http://cakefoundation.org/` (2012-03-19).

[38] Fabien Potencier. High Performance PHP Framework for Web Development - Symfony; 2012. `http://symfony.com/` (2012-03-19).

[39] Zend Technologies Ltd . Zend Framework; 2012. `http://framework.zend.com/` (2012-03-19).

A. Kravspecifikation

Kravspecifikationen är uppdelad i funktionella och icke-funktionella krav. Kraven har dessutom olika prioritet: prioritet 1 är de krav som skall implementeras i projektet och prioritet 2 är krav som implementeras vid vidareutveckling.

A.1 Funktionella krav

A.1.1 Prioritet 1

#1 **Studenter skall kunna logga in**
Alla studenter skall kunna registrera sig, med sin Chalmers-mail som användarnamn, och kunna lämna information som: namn, institution och mailadress.

#2 **Medlemmarna i CHARMk skall kunna logga in**
Alla i CHARMk skall kunna logga in i systemet och administrera endast de delar som rör dennes arbetsuppgifter. Endast CHARM-mail accepteras här som användarnamn.

#3 **Import av företag**
Företagen anmäler sig till mässan via den externa sidan, arbetsmarknadsdagar.se, och det skall därför finnas en funktion för CHARMk som importerar informationen till internsystemet.

#4 **Tilldela student en värdroll**
En student kan av CHARMk tilldelas en av tre roller: företagsvärd, områdesvärd eller övrigt.

#5 **Behörighetsnivåer**
Varje användare skall endast kunna se och ändra information som är specifikt för dennes arbetsuppgift. CHARMk är administratörer och har full behörighet.Företagsvärd skall kunna administrera sitt tilldelade företag, områdesvärd skall kunna administrera sitt tilldelade område och värdrollen övrigt har i nuläget ingen speciell behörighet.

#6 **Tilldelning av värdar till företag och företagsområden**
Alla företag och områden skall tilldelas en värd. Företagsvärden ska kunna se och ändra information om sitt tilldelade företag samt dess beställning. Områdesvärden kan se och ändra vilka företagen som tillhör dennes tilldelade område.

A.1.2 Prioritet 2

#7 **Logg-filer**
Det skall fnnas logg-filer för ändringar som varje person i CHARMk har gjort, och även för alla ändringar i varje företag.

#8 **Mässhistorik**
Det skall finnas historik över gamla mässor. När en ny mässa skapas återställs systemet och mässan arkiveras. Vid återställning skall tilldelade behörigheter försvinna. Alla företag, områden och ansökningar arkiveras men föregående mässas ansökningsformulär återanvänds som grund till den nya.

#9 **Företag skall kunna logga in**
Företag skall kunna logga in för att administrera samt schemalägga sina CHARM-samtal.

#10 **Autogenererade fakturor**
Efter en mässa skall CHARMk enkelt kunna autogenerera en faktura beroende på ett företags beställning.

#11 **CHARM-samtal**
Studenter skall kunna anmäla sig till CHARM-samtal med ett specifikt företag. Anmälan skall innehålla CV och personligt brev.

#12 **Banketten**
CHARMk skall kunna administrera banketten med platshantering och företagsanmälan.

#13 **Godshantering**
Bord, stolar och annat material som behövs inför mässan skall kunna hanteras av CHARMk. Plats och mängd av varje leverans är viktigt att ha koll på.

#14 **Check-in för företag**
Alla företag skall kunna checka in när dom anländer till mässan och hämta ut sin beställning. I beställningen skall alla artiklar vara unikt märkta för spårning. De skall även kunna återlämnas vid mässans slut.

#15 **Effektiv sökfunktion**
All information i databasen skall effektivt och enkelt gå att söka upp. Det är önskvärt att få ut delar av en tabell, exempelvis alla värdars mailadresser eller alla allergiker till banketten.

A.2 Icke-funktionella krav

#16 **Vidareutveckling och underhåll**
Systemet skall vara lätt att både underhålla och utveckla i framtiden.

#17 **Väldokumenterad kod**
Eftersom IT-ansvarig för CHARMk byts ut årligen är det viktigt att det snabbt går att få en övergripande blick om systemets uppbyggnad, varför systemet behöver vara väldokumenterat.

#18 **Säkerhet**
Informationen om företag är sekretessbelagt och säkerheten anses viktig. Lösenorden skall inte sparas i klartext, utan vara krypterade. En obehörig användare skall inte kunna se något om databasens struktur.

#19 **Flexibilitet i värdansökningsformuläret**
Det skall vara enkelt att ändra frågorna i formuläret, från år till år.

#20 **Användarvänlighet**
Systemet skall vara användarvänligt, framförallt i de delar som används av andra än IT-ansvariga i CHARMk.

B. Tabeller

Nedan följer sammanställda tabeller över informationen från utvärderingarna i kapitel 2.

	ASP.NET	**JSP**	**PHP**	**Django**	**Ruby on Rails**
MVC:	Ja	Ja	Ja	Ja	Ja
Licens:	Proprietär programvara (Microsoft-licens)	CDDL	PHP license	BSD	MIT
Databas-hanterare:	Alla. Standard: Microsoft SQL Server	Alla	Alla	Alla. Standard: PostgreSQL	Alla. Standard: SQLite
Multipla databaser:	Ja	Nej	Ja	Ja	Ja
Active Records / DB Objects:	Ja, med tillägg	Nej	Ja	Ja	Ja
Operativsystem:	Windows	Windows, Linux, Mac OS X	Windows, Linux, UNIX, Mac OS X	Linux, Mac OS X, Windows, UNIX	Linux, Mac OS X, Windows, UNIX
Dokumenterade säkerhetshål:	Ja	Ja	Ja	Ja	Ja
CRUD:	Ja	Ja, genom vissa ramverk	Ja, genom vissa ramverk	Ja	Ja
Moduler (XML, CSV):	XML, CSV	XML	XML, CSV	XML, CSV	XML, CSV
Autentisering:	Ja	Nej	Nej	Ja	Nej
Testning:	Ja	Nej	Ja	Ja	Ja
Organisation:	Forum, community, API	Community, API	Forum, community, bloggar	Mailinglista, Wikipedia, API, konferenser	Community, mailinglistor, Twitter, Wikipedia

Tabell B.1: *Programmeringsspråk och ramverk*

	CakePHP [37]	**CodeIgniter** [35]	**Symfony** [38]	**Yii** [36]	**Zend** [39]
Utvecklare:	Cake Software Foundation	EllisLab	Sensio Labs	Yii Software LLC (YiiSoft)	Zend Technologies (äger även PHP)
År:	2005	2006	2005	2008	2006
MVC:	Ja	Ja	Ja	Ja	Ja
Licens:	MIT	OSL	MIT	BSD	BSD
Databashanterare:	Alla*	Alla*	Alla*	Alla*	Alla*
Multipla databaser:	Ja	Ja	Ja	Ja	Ja
Active Records / DB Objects:	Ja	Ja	Ja	Ja	Ja
Dokumenterade säkerhetshål:	Ja	Ja	Ja	Ja	Ja
CRUD:	Ja	Ja	Ja	Ja	Ja
Versioner:	PHP4, PHP5	PHP4, PHP5	PHP5	PHP5	PHP5
Autentisering:	Ja	Nej	Ja	Ja	Ja
Testning:	Ja	Ja	Ja	Ja	Ja
Scaffolding:	Ja	Nej	Ja	Ja	Ja
Organisation:	GitHub, dokumentation	GitHub, community, API, konferenser	GitHub, dokumentation, community, support	Community, dokumentation, forum, IRC-kanal	Wikipedia, mailinglistor, support

Tabell B.2: *PHP-ramverk*

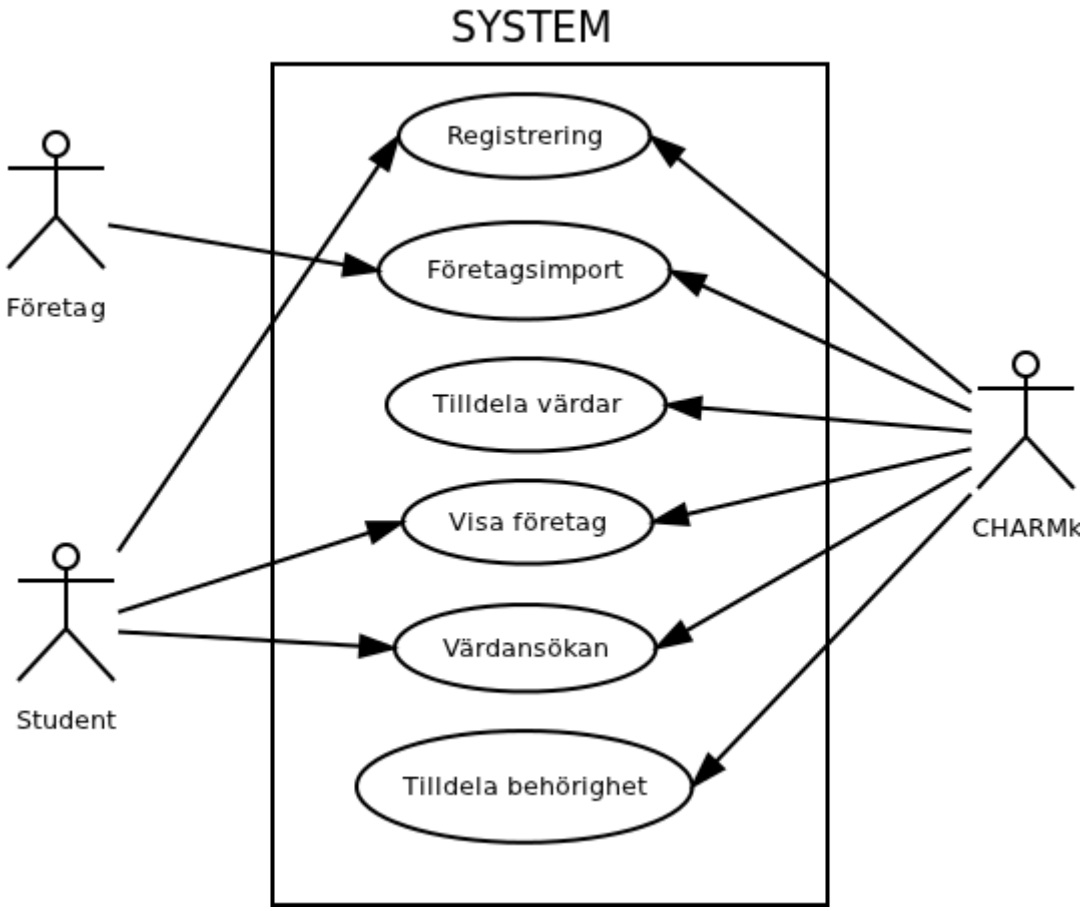
*PostgreSQL, MySQL, SQLite, Microsoft SQL Server, Oracle
Referenserna gäller för hela kolumnerna.

Tillägg: I CakePHP kopplas modelklasser till databastabeller med hjälp av konventioner.

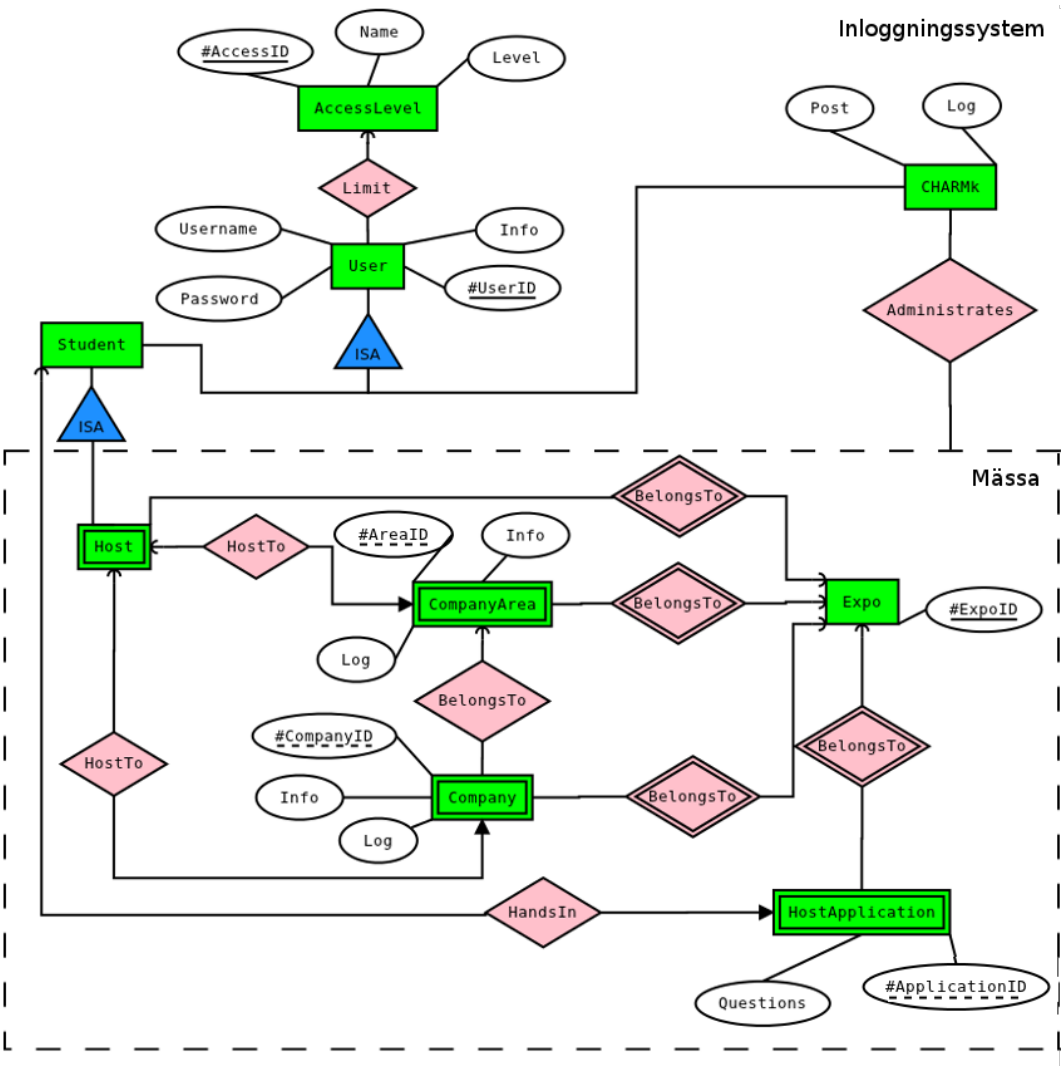
	DB2	**Microsoft SQL Server**	**MySQL**	**Oracle**	**PostgreSQL**
Typ:	Relationsdatabashanterare	Relationsdatabashanterare	Relationsdatabashanterare	Objektorienterad relationsdatabashanterare	Objektorienterad relationsdatabashanterare
Licens:	Proprietär programvara (EULA)	Proprietär programvara (Microsoft-licens)	GPL	Proprietär programvara	BSD
Operativsystem:	Windows, Linux, UNIX	Windows	Plattformsoberoende	Windows, Linux, UNIX	Plattformsoberoende
Programmeringsspråk:	Alla	Begränsat, .NET-ramverk måste användas	Alla	Alla	Alla
Organisation:	Support, installationsguider	Support, guider	Support, community, dokumentation, installationguider	Support, installationsguider	Support, Wikipedia, community, installationsguider

Tabell B.3: *Databashanterare*

C. Modelleringsdiagram



Figur C.1: *Use Case-diagram över systemet*



Figur C.2: *ER-diagram av systemet*

D. Bidragsförteckning

Nedan listas hur varje individ har bidragit till de olika delarna i projektet.

D.1 Ansvarsområden

Projektledare	Oscar Carlsson
Rapportansvarig	Caroline Strandberg
Modellering av systemet	Caroline Strandberg, Oscar Carlsson
Inloggningssystem	Oscar Carlsson
Företagsimport	Mirac Günes, Oscar Carlsson
Ansökningsformulär	Caroline Strandberg
Värdtilldelning	Caroline Strandberg
Sammankoppling av implementeringen	Oscar Carlsson
Konfiguera PHP-Server	Oscar Carlsson
Design av posters	Caroline Strandberg

D.2 Författare av avsnitt

Huvudansvarig författare av avsnitt i slutrapporten

Sammanfattning	Caroline Strandberg, Oscar Carlsson
Ordlista	Caroline Strandberg, Oscar Carlsson
1 Inledning	Alla
2 Teoretisk bakgrund	Oscar Carlsson
2.1 Webbutveckling	Caroline Strandberg
2.2 Säkerhet	Oscar Carlsson
2.3 Språk och ramverk	Caroline Strandberg, Oscar Carlsson

2.3.1 ASP	Alla
2.3.2 JSP	Caroline Strandberg
2.3.3 PHP	Alla
2.3.4 Django	Oscar Carlsson
2.3.5 Ruby on Rails	Oscar Carlsson
2.4 Databashanterare	Caroline Strandberg
2.5 Jämförelse	Caroline Strandberg, Oscar Carlsson
3 Arbetsmetod	Oscar Carlsson
4 Modellering	Caroline Strandberg
5 Implementering	Caroline Strandberg
5.1 Utvecklingsmiljö	Caroline Strandberg, Oscar Carlsson
5.2 MVC-modellen	Caroline Strandberg
5.3 Inloggningssystem	Oscar Carlsson
5.4 Företagsimport	Mirac Günes
5.5 Ansökningsformulär	Caroline Strandberg
5.6 Värdtilldelning	Caroline Strandberg
6 Resultat	Caroline Strandberg, Oscar Carlsson
7 Diskussion	Caroline Strandberg, Oscar Carlsson
8 Slutsats	Oscar Carlsson
A Kravspecifikation	Oscar Carlsson
B Tabeller	Caroline Strandberg, Oscar Carlsson

D.3 Redovisningar

Muntlig halvtidsredovisning	Caroline Strandberg, Oscar Carlsson
Muntlig slutredovisning	Alla
Muntlig opponering	Alla