Utveckling av ett kortbeställningssystem med publiceringsverktyget DotNetNuke

Nicolas Picasso

Examensarbete

Informationsteknik

2010

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	2792
Författare:	Nicolas Picasso
Arbetets namn:	Utveckling av ett kortbeställningssystem med publiceringsverktyget DotNetNuke
Handledare (Arcada):	Göran Pulkkis
Uppdragsgivare:	Esko Otava Oy

Sammandrag:

Examensarbetet gick ut på att planera och utveckla ett kortbeställningssystem med hjälp av gratisversioner av Microsofts utvecklingsverktyg och webbpubliceringsverktyget DotNetNuke. Projektet skulle ersätta ett gammalt kortbeställningssystem som började vara föråldrat och var svårt att vidareutveckla. Examensarbetet presenterar verktygena som användes samt berättar vilka problem begränsningarna av utvecklingsverktygenas gratisversioner medförde. Publiceringsverktyget DotNetNuke underlättade utvecklingen och gjorde det möjligt att utveckla ett modulärt kortbeställningssystem. Ett kapitel i examensarbetet presenterar publiceringsverktyget, dess funktionalitet och tekniken bakom det samt förklarar hur man kan använda det som grund för att bygga webbapplikationer. Därefter följer kapitel om projektets kravspecifikation och hur den praktiska delen av projektet realiserades. Examensarbetet berättar även kort hur webbapplikationen testades för att hitta prestandaproblem och datasäkerhetsrisker. Den sista delen av examensrapporten utvärderar det färdiga projektet och utvecklingsverktygena som användes i projektet.

Nyckelord:	Publiceringsverktyg, DotNetNuke, ASP .NET, Visual Basic .NET
Sidantal:	69
Språk:	Svenska
Datum för godkännande:	24.9.2010

DEGREE THESIS	
Arcada	
Degree Programme:	Information Technology
Identification number:	2792
Author:	Nicolas Picasso
Title:	Development of a card ordering system with the content management system DotNetNuke
Supervisor (Arcada):	Göran Pulkkis
Commissioned by:	Esko Otava Oy

Abstract:
The objective of this thesis was to plan and develop a card ordering system using the free versions of Microsoft´s development tools and the content management system DotNetNuke. The project would replace an old card ordering system that was starting to get outdated and difficult to develop further. The thesis presents the development tools used and tells about the problems the limitations associated with the free versions of the development tools brought about. The content management system DotNetNuke facilitated the development and made it possible to develop a modular card ordering system. A chapter in the thesis presents the publishing tool, its functionality and technology behind it, and explains how to use it as a basis for building Web applications. This is followed by chapters on the project specifications and how the practical part of the project was realized. The thesis also explains briefly how the web application was tested to find performance problems and security vulnerabilities. The last part of the thesis evaluates the finished project and the development tools used in the project.

Keywords:	Content management system, DotNetNuke, ASP .NET, Visual Basic .NET
Number of pages:	69
Language:	Swedish
Date of acceptance:	24.9.2010

Innehåll

1 Inledning 8

1.1 Esko Otava Oy 8

1.2 Plastkort 9

1.2.1 Beröringsfria kort, chipkort och magnetkort 9

1.3 Henkilokortti.net 10

1.4 Projektet Henkilokortti.net 2.0 och dess mål 11

2 Utvecklingsmiljö 13

2.1 Ramverket Microsoft .NET 13

2.1.1 Microsoft Visual Studio 2008 14

2.1.2 Microsoft Visual Web Developer 2008 Express Edition 14

2.2 Javascript 15

2.2.1 jQuery 15

2.2.2 jQuery UI 16

2.2.3 AJAX 16

2.2.4 ASP .NET AJAX Control Toolkit 16

3 Databasomgivning 17

3.1 SQL 17

3.2 Microsoft SQL Server 18

4 Innehållshanteringssystemet DotNetNuke (DNN) 20

4.1 DotNetNukes historia 20

4.2 Tekniken bakom DotNetNuke 20

4.3 DotNetNuke-portaler 21

4.4 DotNetNuke-användare och användarroller 21

4.5 DotNetNuke-tillägg 22

4.6 Utveckling av moduler i DotNetNuke 23

5 Kravspecifikation 27

5.1 Allmän beskrivning om HenkilöKortti .net 2.0 och dess krav 27

5.2 Användarna och användarroller 28

5.3 Funktionella krav 30

5.3.1 Användarhanteringssidan 30

5.3.2 Kortbeställningssidan 30

5.3.3 Korthållartypsidan 32

5.3.4 Korttypssidan 32

5.3.5 Beräknade fält sidan 32

5.3.6 Rapportsidan ... 33
5.3.7 Administreringssidan ... 33
5.4 Migreringen av den gamla databasen till den nya ... 33
5.5 Språkversioner ... 34
5.6 Sökfunktioner inom webbgränssnittet ... 34
6 Den tekniska lösningen ... 36
6.1 Databasen ... 36
6.2 Migreringen av den gamla databasen ... 39
6.3 Modulerna för användargränssnittet ... 40
6.3.1 Beställningsmodulen ... 40
6.3.2 Användarmodulen ... 48
6.3.3 Korttypmodulen ... 50
6.3.4 Beräknade fältmodulen ... 53
6.3.5 Rapportmodulen ... 54
6.3.6 Administreringsmodulen ... 54
6.4 Datasäkerhet inom webbapplikationen ... 56
6.4.1 Allmänt om datasäkerhet ... 56
6.4.2 Vanliga säkerhetsrisker i webbapplikationer ... 56
6.4.3 Datasäkerhetslösningar ... 59
7 Testning ... 61
7.1 Testning för säkerhetsrisker och prestanda ... 61
7.2 Resultat och åtgärder ... 63
8 Slutsatser ... 65
KÄLLOR ... 67
BILAGA ... 69

FÖRKORTNINGAR

ABS	Acrylonitrile Butadiene Styrene
AJAX	Asynchronous JavaScript and XML
CLR	Common Language Runtime
CMS	Content Management System
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
CTS	Common Type System
DAL	Data Access Layer
DBMS	Database Management System
DLL	Dynamic-Link Library
DOM	Document Object Model
FCL	Framework Class Library
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IIS	Internet Information Services
MSIL	Microsoft Intermediate Language
OWASP	Open Web Application Security Project
PET	Polyethylene Terephtalate
PETG	Polyethylene Terephtalate Glycol
PVC	Polyvinyl Chloride
RAM	Random Access Memory

RDBMS	Relational Database Management System
RFID	Radio Frequency IDentification
SQL	Structured Query Language
WVD	Visual Web Developer
XML	eXtensible Markup Language
XSS	Cross-site scripting

1 INLEDNING

Detta examensarbete utfördes för arbetsgivaren Esko Otava Oy. Projektets mål var att planera och utveckla en ny version av ett gammalt kortbeställningssystem, Henkilökortti .net. Det gamla systemet hade utvecklats för flere år sedan inom företaget, men vidareutvecklingen hade utlokaliserats till ett annat företag vilket skapade onödiga administrations- och utvecklingskostnader för arbetsgivaren. I slutet av 2008 beslöts det att en ny version av programmet skulle börja utvecklas av två personer, av vilken den ena skulle planera användargränssnittet och den andra skulle sköta om den tekniska planeringen och utvecklingen. Syftet var att utveckla en förbättrad version av det gamla systemet som skulle kunna vidareutvecklas utan större kostnader för företaget. Projektet skulle utvecklas med gratisversioner av Microsofts utvecklingsverktyg och byggas på innehållshanteringssystemet DotNetNuke för att minska på mängden kod som behövde programmeras. Innehållshanteringssystemet skulle också göra det möjligt att utveckla ett modulärt kortbeställningssystem för att lättare kunna skräddarsy produkten till potentiella kunder.

Rapporten är en teknisk beskrivning av teknikerna bakom projektet och en analys av hur publiceringsverktyget och de begränsade gratisversionerna av utvecklingsverktygena lämpade sig för projektet.

1.1 Esko Otava Oy

Arbetsgivaren som examensarbetet utvecklades för är företaget Esko Otava Oy. Företaget, som grundades år 1972 är tillverkare av plastkort och återförsäljare av plastkortsskrivare, plastkortspräglingsmaskiner, programvara för design och utskrivning av plastkort samt tillbehör för plastkort. Företaget representerar FARGO-kortutskrivare och Matica-kortpräglingsmaskiner i Finland samt är återförsäljare i Finland för kortdesign- och utskrivningsprogrammet CardExchange®. Företaget levererar även tjänster såsom skolning och service för produkterna. Företaget har cirka 15 fast anställda och är privat-

ägt. Företagets kunder är bl.a. restauranger, studerandekårer, fackorganisationer och försäkringsbolag. Företaget har sedan 2005 ett dotterbolag i Moskva, OTAVA-KART. Plastkortens användningsområde varierar, men de vanligaste plastkorten som företaget tillverkar är olika medlems-, kund-, betalnings- och identifieringskort. Den största delen av plastkorten som säljs är traditionella plastkort med eller utan magnetband, men en del är även s.k. smartkort och RFID- kort. Esko Otava Oy har även i mindre skala tillverkat kortmanglar och streckkoder.

1.2 Plastkort

Plastkort kan vara av olika storlekar, men den vanligaste storleken för plastkort är ID-1 storleken som följer den internationella standarden ISO/IEC 7810. Denna storlek kallas också för CR-80 och kortets mått är 85, 60 mm * 53, 98 mm. De vanligaste tjocklekarna för plastkort är 0,76mm, 0,50 mm och 0,3 mm. Bank- och kreditkort brukar följa ISO/IEC 7813 standarden för kortens tjocklek vilken kräver att kortets tjocklek är 0,76 mm med en tolerans av ±0.08 mm. (1, s.28)

Plastkort tillverkas oftast av PVC, ABS, PET eller PETG. Man har försökt hitta på material som skulle lämpa sig bättre och vara mera miljövänligt än plast för tillverkning av kort. Men p.g.a. kostnadsorsaker och de egenskaper kort kräver för hållbarhet och kvalitet har man inte ännu hittat något material som skulle kunna ersätta plast. Materialet av plastkort som tagits ur bruk kan dock till en del återanvändas, bl.a. för tillverkning av plaströr. (Rankl and Effing, 2003 : 41)

1.2.1 Beröringsfria kort, chipkort och magnetkort

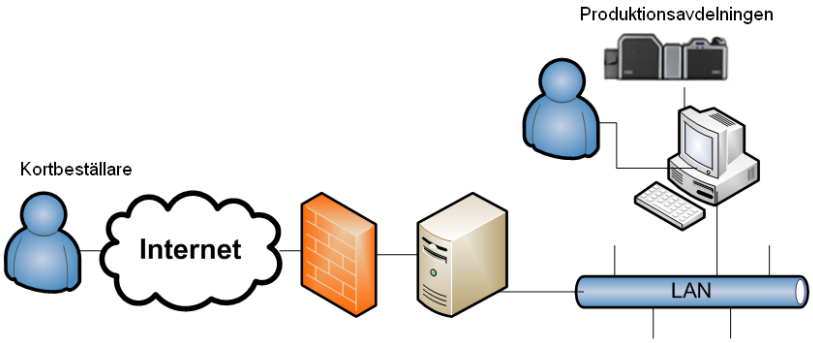
Förutom utskrivning av text och bilder på kortet finns det även andra sätt att spara information och utöka funktionaliteten av plastkort. I magnetkort lagras data på en magnetremsa på kortet. Magnetremsan består av ett ferromagnetiskt material som magnetiseras för att skapa binära koder som representerar tecken. Magnetremsan består av tre spår. Vilkas struktur är definierad i ISO-standarderna 7810, 7811 och 7813. En nackdel med magnetkort är att informationen på magnetremsan lätt kan avläsas med en magnetkortsläsare. Därför lönar det sig inte att spara känslig information på magnetremsan om

informationen inte är krypterad. Antalet tecken som kan sparas på magnetremsan är även mycket begränsat.

Chipkort eller smartkort har en egen processor med minne och en serieport för läsning och skrivning till kortet. Det finns även passiva chipkort som har minne men saknar processor. Passiva chipkort är billigare att tillverka och lämpar sig t.ex. till telefonkort. Fördelar med chipkort är att de kan programmeras att innehålla krypteringsnycklar som kan användas för att signera och kryptera. Förutom pinkod har chipkort även andra säkerhetsmekanismer i hårdvaran. Chipkort är definierade i ISO-standarden 7816. (Rankl and Effing, 2003: 20-21)

1.3 Henkilokortti.net

Personifieringsinformationen för plastkort skickar företagets kunder oftast som Excel- eller textfiler. För att få filerna till ett format som programvaran för kortskrivarna kan behandla har det oftast krävts, att filerna antingen behandlas manuellt eller att man utvecklat program som kan läsa in informationen till en databas. För att underlätta och minska det arbete som krävs, samt för att minska på möjligheterna för fel vid inläsningen utvecklade företaget webbapplikationen Henkilökortti .net, som gjorde det möjligt för kunderna att beställa kort genom ett webbgränssnitt (Figur 1). Via webbapplikationen kunde kunderna via webbgränssnittet mata in kortbeställningens information, som sparades i en databas på servern. Webbapplikationen utvecklades med programmeringsspråket PHP och använder som DBMS MySQL. Henkilokortti.net utvecklades av personer som inte mera är anställda hos företaget. Programmet har fyllt sin funktion under åren, men det har visat sig att det har begränsningar och att vidareutveckling av det är både kostsamt och svårt. Databasens struktur var dåligt planerad vilket gjorde det svårt att lägga till korttyper, kunder och användare. Programmet har saknat möjlighet för kunden att behandla bilder via webbgränssnittet. Logiken bakom hur bilderna för korten sparats har därtill gjort det svårt för produktionen att alltid använda de senaste versionerna av bilderna. P.g.a. det gamla systemets begränsningar beslöt man inom företaget att det skulle löna sig att utveckla en ny version av programmet istället för att fortsätta utvecklingen av det gamla systemet.



Figur 1. Illustrering av Henkilökortti .net:s funktionsprincip

1.4 Projektet Henkilokortti.net 2.0 och dess mål

I början av 2009 beslöts det att man inom företaget skulle börja utveckla en ny version av Henkilokortti.net. Ett företag hade visat sig vara intresserad av att köpa plastkortsskrivare och hyra användningen av webbapplikationen. Det visade sig att det skulle vara för svårt och dyrt att skräddarsy det gamla systemet till kundens behov. Det beslöts därför att två personer skulle planera och utveckla det nya systemet. Förutom ett mera modernt användargränssnitt skulle den nya versionen ha bättre rapporteringsverktyg samt förbättrade egenskaper att administrera företag, användare och korttyper. Projektet skulle planeras så att det skulle vara lätt att förbättra det med tiden samt skräddarsy det för eventuella andra kunders behov. Den nya versionen skulle använda DBMS systemet Microsoft SQL Server, och så lite som möjligt kommersiella komponenter skulle användas i projektet. Den första versionen av kortbeställningssystemet skulle göras på engelska, men systemet skulle planeras så att det skulle vara möjligt att översätta den till andra språk. För att underlätta utvecklingen beslöt man att använda gratisversionen Community Edition av innehållshanteringssystemet DotNetNuke som grund för applikationen. DotNetNuke skulle underlätta programmeringen eftersom det finns färdiga funktioner för bl.a. administrering av användare, rollhantering och lokalisering. För programmeringen av modulerna användes Microsoft Visual Studio Web Developer, som är en gratisversion av Microsofts Visual Studio IDE. Visual Studio Web Developer saknar egenskaper från de kommersiella versionerna av Visual Studio, men har alla de egenskaper som behövdes för projektet. Microsoft SQL Server 2008 Express Edition är också en gratisversion av Microsoft SQL Server med begränsningar. Efter att ha utvär-

derat begränsningarna ansåg projektmedlemmarna dock, att de saknade egenskaperna och begränsningarna inte vägde tillräckligt för att hindra användningen av gratisversionen. För att ge användargränssnittet ett mera modernt utseende än den gamla versionen skulle man använda sig av bl.a. ASP .NET AJAX Control Toolkit komponenter och färdiga Javascript-bibliotek. Efter planeringen av projektet visade det sig att utvecklingsverktygen som skulle användas var gratis, endast servern och licensen för serveroperativsystemet skulle införa kostnader.

2 UTVECKLINGSMILJÖ

2.1 Ramverket Microsoft .NET

Microsofts .NET ramverket är ett ramverk för utveckling av programvara som körs i Microsoft miljö. Ramverket består av två huvudkomponenter, the Framework Class Library (FCL) och Common Language Runtime (CLR). FCL består av närmare 13000 färdiga klasser som kan användas vid programutveckling. Base Class Library, som är en del av FCL innehåller klasser och funktioner för bl.a. databashantering, filhantering och behandling av grafik.

Den andra delen av .NET ramverket, CLR , är .NET:s körtidstjänst och sköter om exekveringen av program utvecklade med .NET. CLR sköter bl.a. om minneshantering, undantagshantering och säkerhet. Olikt många andra programmeringsspråk kompileras inte .NET kod direkt till maskinkod, utan till ett eget språk som kallas MSIL (Microsoft Intermediate Language). .NET ramverket har ett gemensamt datatypsystem Common Type System (CTS) som definierar datatyper som stöds av CLR, vilket gör det möjligt för programkomponenter i olika .NET språk att använda samma variabler med dessa datatyper.

MSIL är plattformsoberoende bytekod som vid exekvering kompileras till maskinkod, s.k. ”just-in-time” kompilering. Endast de metoder som anropas under exekveringen kompileras. På grund av att koden kompileras till MSIL är det möjligt att programmera .NET med alla programmeringsspråk som kan kompileras till MSIL. Fastän ett flertal olika programmeringsspråk kan kompileras till MSIL är de två mest använda C# och Visual Basic .Net. (Ullman m.fl., 2004 : 36-38)

ASP .NET är ett ramverk av Microsoft för utveckling av webbapplikationer och webbtjänster. ASP .NET är byggt på CLR vilket medför att webbapplikationerna kan programmeras med alla programmeringsspråk som stöder CLR. Förutom funktioner för

sessionshantering, mellanlagring av data och innehåll, användarvalidering och en utbyggbar rörledning innehåller ASP .NET tiotals färdiga kontroller för att behandla, visa och validera data i webbapplikationerna.

2.1.1 Microsoft Visual Studio 2008

Visual Studio är en programutvecklingsmiljö från Microsoft. Den första versionen, Visual Studio 97 utkom 1997 och redan följande år utkom följande version, Visual Studio 6.0. Visual Studio 6.0 har stöd för Active Server Pages, en teknik utvecklad av Microsoft för att skapa dynamiska webbsidor. År 2002 lanserade Microsoft Visual Studio .NET som introducerade Microsoft .NET ramverket och hade stöd för ASP .NET, Microsofts nya ramverk för utveckling av dynamiska webbsidor. 2003 utkom en ny version som till största delen bestod av små förbättringar av den tidigare versionen. År 2005 lanserades .NET 2.0 ramverket och samma år kom även Visual Studio 2005 ut med stöd för bl.a. ”ASP .NET 2.0” -sidor, nya projekttyper samt en egen webbserver. Följande version av Visual Studio, Visual Studio 2008, utkom i slutet av 2007. Visual Studio 2008 har stöd för .NET ramverket 3.5, Windows Vista, Office 2007, en ny designmodul för HTML/CSS och Windows Presentation Foundation samt bättre stöd för Javascript m.m.

2.1.2 Microsoft Visual Web Developer 2008 Express Edition

Microsoft Visual Studio Express är en samling gratis programutvecklingsmiljöer från Microsoft. Expressversionerna har begränsningar och saknar egenskaper från standard- och den professionella versionen av Visual Studio. Expressversionen är mera riktad till studerande och personer som programmerar som hobby. Trots att Expressversionen har begränsningar är det dock möjligt att göra kommersiella program med den. Visual Studio Express Edition består av följande produkter:

- Visual Basic Express
- Visual Web Developer Express
- Visual C++ Express
- Visual C# Express
- SQL Server Express

- Express for Windows Phone

Programutvecklingsmiljön för utveckling av ASP.NET-sidor från Visual Studio Express heter Visual Web Developer (VWD) Express. VWD stöder också andra teknologier, bl.a. Javascript, XML och CSS. Funktioner och egenskaper som saknas i VWD jämfört med den professionella versionen av Visual Studio är bl.a. :

- Versionshantering. VWD saknar möjlighet att integrera versionshanteringsprogram.
- Lokalisering. VWD saknar automatisk generering av lokaliseringsresurser.
- Begränsade avlusningsalternativ.
- VWD kan inte utökas med tredjepartskomponenter.
- Färre projektmallar

2.2 Javascript

Javascript är ett objektorienterat procedurellt skriptspråk. Den vanligaste användningen för Javascript är att skapa i webbsidor funktionalitet som inte är möjligt med HTML. Javascript gör det möjligt att bl.a. utföra beräkningar, validera input, generera aktivitet vid mus- eller tangenttryckning, identifiera vilken webbläsare används samt visa datum och tid. Det finns flera olika Javascript-bibliotek som underlättar utvecklingen av Javascript, bland de populäraste är jQuery, script.aculo.us, MooTools och Yahoo! UI Library. (Heilmann, 2006 : 4-7)

2.2.1 jQuery

jQuery är ett Javascript-bibliotek som underlättar programmering av Javascript i webbsidor. JQuery gör det möjligt att välja och gå igenom noder i HTML-sidans Document Object Model(DOM) samt göra förändringar och skapa i sidan effekter, som utförs av klientens webbläsare. jQuery har stöd för händelser, CSS manipulering, effekter, Ajax och animationer. jQuery är kompatibel med de flesta moderna webbläsare. Det är även möjligt att tillägga funktionalitet och metoder till jQuery och återanvända dem som en s.k. jQuery plug-in. (Chaffer, 2009 : 7-9)

2.2.2 jQuery UI

jQuery UI är ett Javascript-bibliotek byggt på jQuery som består av ett antal gränssnittskomponenter och interaktionsfunktioner för att förbättra användargränssnittet i webbsidor. ”jQuery UI”-gränssnittskomponenter är till stor del konfigurerbara och från ”jQuery UI”-hemsidan är det möjligt att planera egna eller ladda ner färdiga teman för gränssnittskomponenterna. ”jQuery UI”-interaktionsfunktioner gör det bl.a. möjligt att ändra på storleken, flytta, välja och sortera olika element i webbsidan. Till ”jQuery UI”-gränssnittskomponenter hör bl.a. en flikkomponent, skjutreglage, datumväljare, dialog och förloppsindikator. jQuery UI har också ett antal olika effektfunktioner, bl.a. färganimationer samt tilläggning och borttagning av CSS-klasser med animerade övergångar. (Wellman, 2009 : 1)

2.2.3 AJAX

Ajax, som är en förkortning av Asynchronous JavaScript and XML, består av en metodik som gör det möjligt att asynkront hämta data från servern. Istället för att ladda hela webbsidan på nytt gör Ajax det möjligt att vid behov uppdatera endast en del av sidan, vilket kan förbättra interaktiviteten i webbsidan och minska serverns börda i större dynamiska webbsidor. Grunden i Ajax ligger i XMLHTTPRequest-objektet, som stöds av de flesta moderna webbläsare. (Ullman and Dykes, 2007 : 2-14)

2.2.4 ASP .NET AJAX Control Toolkit

Ajax Control Toolkit är en samling programvaror med öppen källkod som består av cirka 40 anpassningsbara ”ASP .NET Ajax kontroller” och utvidgningar till kontroller. Ajax Control Toolkit kan laddas ner från Codeplex, en webbsida av Microsoft för att hysa programvara med öppen källkod. Ajax Control Toolkit kommer med färdiga lösningar för Visual Studio för att man skall kunna anpassa och vidareutveckla kontrollerna till sina egna behov.

3 DATABASOMGIVNING

En relationsdatabas är en databas där informationen sparas i tabeller som består av kolumner och rader. Tabellerna kan ha relationer till varandra, eventuellt med restriktioner mellan relationerna. Tabellerna kan refereras till varandra på tre olika sätt, en till många, många till många eller en till en. För att lätt kunna identifiera rader i tabeller definierar man en kolumn som primärnyckel. Informationen i kolumnen som definieras som primärnyckel måste vara unik, d.v.s. det får inte finnas flera rader som med samma värde i primärnyckelkolumnen. För att skapa relationer mellan tabeller använder man främmande nycklar. En främmande nyckel innehåller inte information om raden, utan den fungerar som referensvärde till en annan tabell. Data som kopplats samman med nycklar måste vara av kompatibel datatyp. (Silberschatz m.fl., 2006 : 11-18)

Database Management System (DBMS), eller databashanterare, är programvara som sköter om organisering, lagring, hantering och hämtning av data i en databas. Databashanterare för relationsdatabaser kallas för RDBMS. Exempel på RDBMS är MySQL, Microsoft Access, Microsoft SQL Server och DB2.

Relationsdatabaser har ett datahanteringsspråk (DML), ett språk för att hämta och modifiera data i databasen, och ett datadefinitionsspråk (DDL) för hantering av databasens struktur. De flesta RDBMS stöder Structured Query Language (SQL), som är ett standardiserat språk för att hämta och modifiera data i databasen. SQL fungerar både som DML och DDL.

3.1 SQL

SQL utvecklades av IBM under början av 1970 under namnet Sequel. Sequel utvecklades vidare och namnet byttes till SQL. SQL språket består bl.a. av följande delar.

- DDL. SQL har kommandon för att skapa, ta bort och modifiera relationer.
- DML. SQL har kommandon för att skapa, ta bort och modifiera rader i tabeller.
- Integritet. SQLs DDL har kommandon för att specificera integritetsbegränsning-

ar som information som sparas i databasen måste uppfylla.

- Autentisering. SQL: DDL har kommandon för att specificera åtkomsträttigheter till relationer och vyer

Fastän SQL är standardiserat har flera tillverkare av RDBMS utvidgat eller gjort sina egna realiseringar av SQL-språket, t.ex. Oracles P/L SQL och Microsofts T-SQL. (Silberschatz m.fl., 2006 :75-77)

3.2 Microsoft SQL Server

Microsoft SQL Server är ett RDBMS som först utvecklades till operativsystemet OS/2. Eftersom OS/2 kommersiellt inte blev en framgång utvecklades SQL Server för att börja stöda Windows 3.0. Efter att Microsofts och IBM samarbete slutat fortsatte Microsoft att utveckla SQL Server till Windows NT. År 2000 utgav Microsoft en ny version som hade bl.a. stöd för Windows 2000, möjlighet att visa resultatet av en SQL sats i XML format och en integrerad avlusare.

2005 utkom följande version av SQL Server . SQL Server 2000 hade kritiserats för sina datasäkerhetsrisker och hade bl.a. blivit utsatt för Internetmasken Slammer som utnyttjade en buffertöverskridningsbugg i mjukvaran. SQL Server 2005 hade stora förändringar i säkerhetsmodellen jämfört med tidigare versioner. Användning av lösenordspolitik, separering av användare, DDL-utlösare samt bättre granularitet av rättigheter gjorde SQL Server 2005 betydligt mera säkert än den tidigare versionen. Förutom förbättrad datasäkerhet hade SQL Server 2005 flera andra förbättringar. Den nya databasmotorn hade förbättringar i indexeringen av databaser, tabellstrukturen, en utvidgad version av T-SQL och förbättringar i säkerhetskopieringen av databaserna. Ett nytt GUI för administrering av databaserna, SQL Server Management Studio (SSMS), introducerades i SQL Server 2005 och ersatte Enterprise Manager, Query Analyzer och Analysis Manager i den tidigare versionen.

Den nyaste versionen av SQL server, SQL Server 2008, bygger upp på SQL Server 2005 med nya datatyper, bättre datakompression, förbättrade indexeringsalgoritmer, bättre optimerad databasspegling och IntelliSense-stöd för förfrågningseditorn.

SQL Server 2008 kommer i flera olika versioner riktade till olika målgrupper. De olika versionerna är:

- Enterprise Edition
- Standard Edition
- Workgroup Edition
- Web Edition
- Express Edition
- Express Advanced Edition
- Developer Edition
- Compact Edition

De olika versionerna skiljer sig beträffande egenskaper, hårdvara och kostnader. Express-versionen som är en gratis version kan t.ex. endast utnyttja en 1Gb RAM minne, endast en processor och databasens storlek får inte överskrida 10Gb. Express versionen saknar också avancerade egenskaper såsom t.ex. datakompression och genomskinlig kryptering av databasen. I Tabell 1 jämförs viktiga egenskaper i de olika versionerna. (Nielsen m.fl., 2009 : 20-26)

Tabell 1. Jämförelse av egenskaper i SQL Server 2008:s olika versioner

Jämförelse av egenskaper i SQl Server 2008:s versioner	Version				
	Enterprise	Standard	Workgroup	Web	Express
Antal processorer som kan utnyttjas	8	4	2	4 (endast för webbens arbetsbelastning)	1
Mängd RAM som kan utnyttjas	2 TB	64 GB	4GB	Operativsystemets begränsning	1 GB
Databasens storlek	524 PB	524 PB	524 PB	524 PB	10 GB
Datakomprimering	Ja	Nej	Nej	Nej	Nej
SQL Server Agent	Ja	Ja	Ja	Ja	Nej
Database Tuning Advisor	Ja	Ja	Ja	Ja	Nej
Import And Export Wizard	Ja	Ja	Begränsad funktionalitet	Begränsad funktionalitet	Begränsad funktionalitet
SQL Profiler	Ja	Ja	Ja	Nej	Nej

4 INNEHÅLLSHANTERINGSSYSTEMET DOTNETNUKE (DNN)

Ett webbinnehållshanteringssystem (CMS) är en applikation för att förenkla utveckling och hantering av webbsidor och portaler. Webbsidorna administreras och redigeras via CMS systemets webbgränssnitt. CMS systems egenskaper kan variera men de flesta större CMS brukar ha bl.a. följande egenskaper: (Christianson and Cochran, 2009 : 6-7)

- Separering av innehåll och layout
- Design av layouten via mallar
- Hantering och kategorisering av sidor
- Automatisk uppdatering av navigeringskontrollerna för sidorna
- Användarhantering
- Publicering av artiklar
- Händelsehantering

4.1 DotNetNukes historia

Då Microsofts .NET ramverk version 1.0 ännu var i betaversion gav Microsoft ut kod- och projektexempel för att befrämja utvecklare att lära sig ramverket .NET. Ett av projektexemplen var IBuySpy Portal som Microsoft utvecklat med Vertigo Software. Fast-än IBuySpy Portal utvecklats till att vara exempel för programmerare, hade det många egenskaper likt andra innehållshanteringssystem med öppen källkod. IBuySpy Portal gjorde det möjligt att skapa en webbsajt med ett obegränsat antal sidor, och sidorna hade behållare för att dynamiskt tillägga innehåll till sidorna i form av moduler. DotNetNukes grundare, Shaun Walker, fortsatte att bygga vidare på projektet och i början av 2003 fick projektet namnet DotNetNuke. DotNetNuke har fortsatt att vara ett innehållshanteringssystem med öppen källkod och har vid tiden av detta examensarbete varit uppe i version 5.1. (Walker m.fl., 2009 : 2-3)

4.2 Tekniken bakom DotNetNuke

DotNetNuke körs på Microsoft .NET ramverk och dess kärnkod är skrivet i Visual Basic .NET. Den nyaste versionen stöder egenskaper av .NET ramverket 3.5. DNN bygger på en treskiktad arkitektur som består av presentations-, affärslogik- och dataåtkomst-

skikten. DNN är databasdrivet, dvs. det lagrar största delen av informationen i en databas som körs på Microsoft SQL Server. DNN körs på en webbserver som stöder ASP.NET, t.ex. IIS eller den inbyggda webbservern i VWD Express(s.223). DNN version 4 och nyare kräver att ramverket .NET 2.0 är installerat. (Hammond and Renner 2009 : 4-5)

4.3 DotNetNuke-portaler

En av DotNetNukes fördelar är att man med en installation kan hysa flera portaler. Administratorn kan skapa huvudportaler eller underportaler till installationen. En fördel med detta är att flera portaler kan köras med samma kodbas och databas. Det betyder att om man har en portal med URL:n http://www.portalnamn.net så kan man skapa underportaler till den som man kommer åt genom att lägga till underportalens namn till URL:n, t.ex. http://www.portalnamn.net/portal1 och http://www.portalnamn.net/portal2. En fördel med detta är att vid uppgraderingar räcker det med att man uppdaterar modulerna eller DotNetNukes bibliotek för en installation. Nackdelar är att alla underportaler körs under samma programpool så att om en sajt faller så faller alla. Moduler som är dåligt programmerade kan leda till problem för alla underportaler beroende på felen, vilket är en sak att ta i beaktande då man bestämmer om man skall köra flera portaler på samma installation. (Sellers, 2009 : 35-36)

4.4 DotNetNuke-användare och användarroller

Vid en installation av DNN skapas två användare, värd och administrator. Värden eller Superuser har fulla administrativa rättigheter och är den enda användaren som har rättighet att skapa nya portaler. Superuser kan installera nya tillägg, administrera användare och innehåll i portalerna. Administrator-användaren är bunden till en portal. Administratorn har rättighet att behandla innehållet till portalen han hör till, samt har rätt att utnyttja de moduler Superusern har installerat till portalen. Eftersom moduler och skin kan innehålla kod som kan påverka alla portaler i en DNN installation har endast Superusern rättighet och ansvaret för installation av nya tillägg. (Sellers, 2009 : 33-34)

DotNetNuke har inbyggd administrering av användare. Från användaradministrationspanelen kan administratorn skapa nya användare, administrera användare och deras roller. DotNetNuke har fyra olika typer av användarregistrering, ingen, privat, offentlig och verifierad. Ingen betyder att endast administratörer kan skapa nya användare. Privat registrering betyder att användaren har möjlighet att registrera sig till webbsajten men har inte möjlighet att logga in till sajten förrän en administrator har godkänt registreringen. Offentlig registrering som är det förinställda värdet betyder att vem som helst kan registrera sig till sajten efter att ha registrerat sig och fyllt i den obligatoriska informationen om sig själv och godkänns därefter som användare utan administratorns bekräftelse. Det sista alternativet, verifierad, betyder att efter att ha fyllt i registreringsuppgifterna får användaren en e-post som innehåller en verifieringskod användaren måste fylla i första gången han loggar in till sajten. Med verifieringskoden försäkrar man sig om att användarens e-post adress är giltig. (Hammond and Renner, 2009 : 78-7)

En användarroll är ett sätt att gruppera användare som har samma ändamål. DotNetNukes användarroller gör det möjligt att begränsa åtkomsten till portalen för användarna. Efter en DotNetNuke-installation finns det färdigt fyra olika roller. Oautentiserade användare är användare som inte har loggat in till sajten. Registrerade användare är alla användare som har ett konto till sajten. Förutom oautentiserade och registrerade användare skapas även rollgrupperna värd och administrator vars uppgifter beskrevs tidigare i kapitlet. Förutom de förinställda användarrollerna kan administratörer skapa och radera användarroller. En användare kan tillhöra flera användarroller, och flera användare kan tillhöra samma roll. (Washington and Lackey, 2010 : 52-54)

4.5 DotNetNuke-tillägg

En av orsakerna till DotNetNukes flexibilitet är dess tillägg. De två vanligaste tilläggen för DotNetNuke är skal och moduler. Skal är paket som innehåller de komponenter som behövs för portalens design. Ett skal innehåller även information om modulbehållare. Ett skal kan innehålla flera olika modulbehållare med unika namn till vilka administratören för webbsajten kan placera moduler. Modulbehållarna har egenskaper som gör det möjligt att bestämma bl.a. på vilka sidor modulen visas, start och utgångsdatum för mo-

dulen, rollrättigheter, om modulen kan förminskas eller printas, samt om modulen skall ha en rubrik och sidfot.

Skal liksom andra tillägg installeras med s.k. paket, komprimerade zip-filer som innehåller tilläggens filer samt en fil i XML-format med information om tillägget och filerna i paketet. Det är möjligt att använda samma skal för hela portalen, eller använda olika skilda skal för varje sida. (Walker m.fl., 2009 : 515-516)

Innehållet och funktionaliteten i en DNN-webbsajt presenteras via moduler som man lagt till sidorna. Med DNN kommer det ett flertal grundmoduler, som man kan installera vid installationsskedet eller senare. En av kärnmodulerna i DNN är text/HTML-modulen, som gör det möjligt att redigera och visa innehåll i HTML-format inom modulen. Administratörer kan bestämma vilka användarrollgrupper har rätt att redigera innehållet, och vilka som har rätt att se modulens innehåll. DNN-moduler gör det möjligt för utvecklare att expandera DotNetNukes funktionalitet genom att utveckla nya moduler. Moduler kan utvecklas med alla .NET språk, fastän DotNetNuke och dess kärnmoduler är gjorda med Visual Basic .NET. Moduler som har lagts till sajten har ett eget idnummer vilket gör det möjligt att visa olika data för varje enskild realisering av modulen. (Sellers, 2009 : 31-32)

4.6 Utveckling av moduler i DotNetNuke

En DotNetNuke-modul är en ASP .NET användarkontroll som består av en komponent för användargränssnittet och kod som kan placeras i användargränssnittkomponentens inline-kod eller i en skild code-behind fil.

Förutom .NETs bibliotek har DotNetNuke-utvecklare tillgång till DotNetNukes kärnbibliotek vilket gör det möjligt att utnyttja DotNetNukes API för att underlätta programmering av moduler. DotNetNukes ramverksfunktioner och klasser är organiserade i

en logiskt organiserad namnrymd. Till DotNetNukes namnrymd hör: (Sellers, 2009 : 241-242)

- DotNetNuke.Common: Innehåller funktioner och klasser som används i hela DotNetNuke applikationen.
- DotNetNuke.Data: Innehåller klasser för datatillgångsskiktet.
- DotNetNuke.Entities: Innehåller klasser för de fem enheter en portal består av, host, portaler, flikar, användare och moduler.
- DotNetNuke.Framework: Innehåller klasser och funktionalitet för grundfunktioner i DotNetNuke .
- DotNetNuke.Modules: Innehåller klasser för organisering av moduler.
- DotNetNuke.Security: Funktioner och klasser för DotNetNukes auktorisering och autentisering av användare. I denna namnrymd finns bl.a. funktioner för att hämta och lagra rättigheter om portalens sidor, moduler och användarrollernas rättigheter.
- DotNetNuke.Services: Innehåller den funktionalitet DotNetNukes kärnkod stöder för moduler. Till dessa hör bl.a. lokalisering, sökfunktioner, undantagshantering och personalisering av modulerna.
- DotNetNuke.UI: Funktioner för gränssnittet, bl.a. skal och modulbehållarna.

DNN kommer i fyra olika paket som kan laddas ner. Starter Kit är ett Visual Studio Installer Package (.vsi) paket som efter att ha installerats lägger till DotNetNuke mallar till Visual Studio. DotNetNuke Web Application Framework innehåller hela DotNetNuke lösningen, även kärnkoden för DotNetNuke. ”DotNetNuke Compiled Module”-mallen skapar ett ”Web Application Project (WAP)”-kompilerat modulprojekt. WAP-modulprojektet skapar alla nödvändiga element som behövs för modulen, bl.a. datatillgångsskiktet (DAL) för kommunikation med databasen samt tre användarkontroller. Det sista paketet, Dotnetnuke Dynamic Module, skapar ett Web Site Project (WSP) som innehåller till stor del samma element som WAP. WAP och WSP innehåller dock skillnader i projektets strukturering och hur det kompileras. (Sellers, 2009 : 4-6)

I ASP .NET 1.x skapas webbapplikationerna i webbsidans katalog. Efter kompilering kopieras den nya DLL-filen till bin-katalogen. WAP-projekt innehåller både affärslogiken och användargränssnitt. Ett av problemen med denna projektmodell var att man måste manuellt koppla avlusaren till IIS:s arbetsprocess för att kunna avlusa applikationen. Under avlusningen var filerna låste vilket krävde att man efter varje förändring av koden måste kompilera koden och manuellt koppla avlusaren till arbetsprocessen på nytt, vilket var arbetsamt och tidsfördrivande.

ASP .NET 2 introducerade en ny dynamisk kompilationsmodell. Med denna modell placeras affärslogikens filer och användargränssnittets filer i skilda kataloger. Denna modell gör det möjligt att dynamiskt kompilera all kod som inte hör till kataloger reserverade för ASP.NET. Om man gör förändringar i sidan kompileras sidan på nytt först följande gång en förfrågan till sidan görs. Efter det lagras den kompilerade sidans sammanställningsfil i ett cache och sidan kompileras på nytt endast om man gör förändringar i sidan. Om man gör förändringar i ASP .NET:s reserverade kataloger eller ändrar på webbsajtens konfigurationsfil web.config leder det till att hela webbsajten kompileras på nytt.

WAP-modulprojektet följer ASP .NET 1.x projektmodell och behandlar modulerna som skilda projekt, medan WSP följer den nya dynamiska modell i ASP .NET 2 där modulerna är en del av hela lösningen och inte skilda projekt. Båda projektmallarna har sina fördelar. Från distribueringsperspektiv visade det sig att WAP-modellen är lämpligare för Henkilökortti .net 2.0 eftersom man inte behöver ge källkoden till kunden då modulernas kod förkompileras till en DLL biblioteksfil. Vid utvecklingen av Henkilökortti .net 2.0 användes inte versionshantering, men i projekt som använder versionshantering är WAP-modellen lämpligare. Med WSP-modellen måste hela DotNetNuke projektet checkas in och ut medan WAP-modellen möjliggör att endast modulprojektet checkas in och ut. WSP-modellen gör det möjligt att avlusa DotNetNuke moduler med Microsofts Visual Web Developer eftersom webbsajten kan köras med VWD:s egen webbserver. WAP modellen kommer inte med stöd för avlusning med VWD eftersom man med

VWD inte kan koppla avlusaren till andra processer vilket gör det betydligt svårare att söka efter fel i koden. (Sellers, 2009 : 12-13)

Eftersom Henkilökortti .net 2.0 skulle utföras med VWD beslöt man efter att ha övervägt fördelarna och nackdelarna av projektmodellerna att bygga projektet med båda modellerna. Utvecklingen skulle ske med WSP-modellen för att underlätta avlusning, och då koden testats och visat sig fungera skulle den flyttas över till WAP-projektet.

5 KRAVSPECIFIKATION

5.1 Allmän beskrivning om HenkilöKortti .net 2.0 och dess krav

För att den Henkilökortti .net 2.0 skulle uppnå arbetsgivarens och kundens krav beslöt man att en anställd skulle ansvara för planeringen av användargränssnittet och funktionaliteten och en för planeringen av den tekniska lösningen. Efter flera möten med kunden och palaver hos kunden lyckades man få fram en kravspecifikation som skulle tillfredställa kundens och arbetsgivarens behov. Istället för att planera det nya systemet från början analyserade man det gamla och gjorde en lista över nya egenskaper och förbättringar det nya systemet skulle ha.

En av de största nackdelarna med det gamla systemet är att det inte är flexibelt. Om man vill lägga till korttyper eller användare är man tvungen att göra förändringar i koden och databasen. Det nya systemet skall vara mera dynamiskt och modulärt. Företag, användare, korttyper och korthållare skall kunna läggas till och raderas av administratören från webbgränssnittet. I motsats till det gamla systemet skall en installation av webbapplikationen fungera för flera företag. Med det gamla systemet är det möjligt att flera företag använer samma installation och databas, men det är inte möjligt att definiera företag och företagets användare. Det enda sättet att definiera till vilket företag användaren hör till är genom att begränsa vilka korttyper han kan beställa, och det sker genom att manuellt göra förändringar i databasen. Informationen som används för att personifiera korten och fältens namn är hårdkodade i den gamla versionen av Henkilökortti .net. Med det nya systemet skall det vara möjligt att definiera antalet fält samt fältens namn för varje korttyp. Antalet fält som används i det gamla systemet per korttyp är oftast mellan tre till fem, så det beslöts dock att sätta en maximigräns på 10 fält per korttyp till Henkilökortti .net 2.0. Administratören skall kunna sätta begränsningar på hurdan information som kan sparas i fälten. Förutom största och minsta antalet tecken per fält skall det också vara möjligt att bestämma om fältet tillåter numeriska, alfabetiska eller alfanumeriska tecken. Förutom de vanliga fältena beställarna kan fylla i data skall det också vara möjligt att skapa fält vars innehåll genereras automatiskt av webbapplikationen. I den första versionen av Henkilökortti .net 2.0 skall detta realiseras genom att kunna definiera s.k.

beräknade fält. Beräknade fält har en egen tabell som sparar värden, och vid följande beställning hämtar applikationen det senaste värdet och adder en etta till det. Förutom själva räknevärdet skall det även vara möjligt att kunna definiera ett prefix och suffix till räknevärdet som läggs till automatiskt då kortet beställs. I framtida versioner skall det också vara möjligt att kunna räkna kontrollsummor till fälten.

Varje korttyp skall ha ett fält som kan definieras som ett bildfält. Administratören kan välja vilket vält, t.ex. efternamnet, vars innehåll namnger bilden som sparas på servern.

5.2 Användarna och användarroller

Användarna till webbapplikationen skall bestå av en Superuser, en portaladministrator, administratörer för företaget och kortbeställare (Figur 2). Alla användare skall ha ett eget användarkonto till webbapplikationen. Vid skapandet av användare skall applikationen kontrollera att informationsfälten om användaren är korrekt fyllda, och meddelar om den inmatade informationen är bristfällig. De nödvändiga fälten är förnamn, efternamn, användarnamn och e-post adress. Användarkontot skall genast vara användbart efter att det skapats och en e-post skall automatiskt skickas till användaren med användarnamnet och lösenordet vid skapandet. Vid borttagning av användare blir kundens beställningar kvar i systemet, och skall i senare skede kunna överflyttas till en annan beställare. Efter att kontot raderats får användaren en e-post som meddelar att hans konto raderats. Alla uppgifter om användaren skall kunna användas förutom användarnamnet.

Superuser

Superusern är den första användaren i systemet och skapas vid installationen av DotNetNuke. Superusern skall ha fulla rättigheter att administrera webbapplikationen och användarna. Administrationen skall ske via webbgränssnittet. Till Superuserns uppgifter hör till skillnad från de andra användarna inte att beställa kort, men han skall ha rättighet att flytta över en användares beställningar till en annan. Superusern har åtkomst till alla sidor i webbapplikationen förutom kortbeställningsmodulen.

Eftersom Superusern är den person som har mest rättigheter är det hans uppgift att definiera nya företag till webbsajten och skapa den första administratören till företaget. Superusern skall också ha rätt att definiera virtuella kataloger till vilka bilderna från kortbeställningarna sparas till. Den virtuella katalogen skapas från IIS, men Superusern kan från användargränssnittet definiera vilka företag har rätt att använda dem.

Portaladministrator

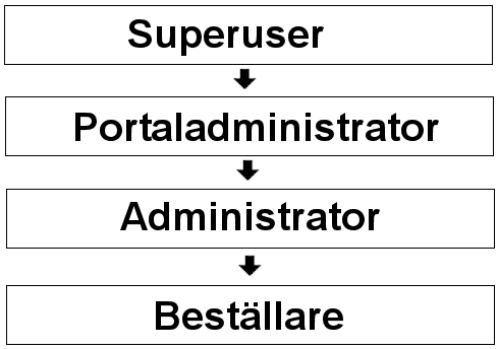
Portaladministratorn har samma rättigheter som Superusern till kortbeställningsapplikationen. Han är dock bunden till en portal och har inte ett användarkonto till andra portaler i samma installation.

Administrator

Administratörer är oftast överordnade i företag. En administratör skall ha rätt att kunna lägga till nya administratörer och beställare till företaget. Administratören kan från skilda sidor administrera korthållartyper, korttyper och beräknade fält. Administratören har rätt att bestämma vilka korttyper beställarna har rätt att beställa, och kan göra det vid skapandet av nya användare eller i ett senare skede. Administratören har åtkomst till rapportsidan från vilken han kan se rapporter om sitt företags användare och beställningar.

Beställare

Beställarna är de användare med minst rättigheter i systemet. Beställarna har bara rättighet att beställa de korttyper administratörerna har gett dem rättighet till. Beställarna har rättighet att se alla kortbeställningar för de korttyper de getts rättighet till, men har bara rättighet att modifiera och återbeställa de kort de själva beställt.



Figur 2. Användarhierarkin i webbapplikationen

5.3 Funktionella krav

5.3.1 Användarhanteringssidan

I användarhanteringssidan visas företagets användare i två tabeller. Den ena visar företagets administratörer och den andra beställarna. Superusern och portaladministratorn skall från en flervalstagg kunna välja vilket företags användare visas, administratörerna ser enbart sitt eget företags användare. Från sidan skall det vara möjligt att skapa nya användare, radera användare och ändra på användarnas uppgifter.

5.3.2 Kortbeställningssidan

Då kortbeställningssidan öppnas skall användaren kunna se tidigare beställningar. Beställningarna skall visas i en tabell som visar en beställningar för en korttyp på gången. Genom en flervalstagg kan väljaren välja vilken korttyps beställningar visas. Om korttypen har ett bildfält definierat skall tabellen visa en bild som berättar om kortet har en bild på servern. Genom att flytta muspekaren på bilden öppnas ett fönster som visar en förhandsbild av kortets bild.

Användaren har möjlighet att beställa nya kort från sidan för den korttyp som valts i flervalstaggen. Då användaren beställer ett nytt kort skall ett modalt fönster visas med de fält som definieras för korttypen. Då användaren fyllt i uppgifterna för kortet och

trycker på beställningsknappen skall programmet kontrollera att fältens innehåll uppfyller kraven. Om de uppfyller kraven sparas beställningen och en text visas för användaren att beställningen är gjord. Om inte visas bredvid textfälten en text som berättar för användaren varför informationen inte uppfyller kraven. Efter att ha sparat beställningen skall användaren ha möjlighet att kunna ladda upp en bild till beställningen om kortet har ett bildfält definierat. Beställaren skall också ha möjlighet att ladda upp en ny bild till beställningen ifall en bild redan laddats upp. I så fall skall användaren bekräfta att han vill radera den gamla bilden förrän han kan ladda upp en ny.

Kortbeställningstabellen skall ha knappar för att skriva ut information om kortet, ladda upp en bild till kortet, ändra på kortets uppgifter, visa en förhandsbild av kortet samt en knapp för att radera kortbeställningen. Om användaren inte har rättighet att ändra på beställningar, d.v.s. han är inte den som beställt kortet skall bilderna för att ladda upp bild, ändra på kortets uppgifter och raderingsknappen vara ersatta med en bild som visar att funktionerna inte är tillgängliga. Kortbeställningstabellen skall visa de fält som är definierade i korttypen, i vilket tillstånd beställningen är samt beställningsdatum och utskriftsdatum.

Behandling av bilder via webbgränssnittet

Eftersom de flesta kortutskrivningsprograms (t.ex. CardExchange och Cardfive) kortformat kräver att bilderna är sparade i samma bildformat stöder programmet endast bilder i JPG format eftersom det är det vanligaste bildformatet för fotografiska bilder och stöds av de flesta webbläsarna. Efter uppladdningen av bilden har beställaren möjlighet att behandla bilden på servern. De bildmanipuleringsfunktioner som implementeras i den första versionen är:

- Rotering av bilden i 90 grader medsols eller motsols
- Förminskning av bilden i procentgrader
- Beskärning av bilden
- Konvertering till gråskala

I senare versioner skall det vara möjligt att definiera minimi- och maximiresolution för bilderna, samt kunna välja bildformatet för bilden.

5.3.3 Korthållartypsidan

Från korthållartypsidan kan administratören skapa nya, ta bort och modifiera informationen om korthållartyper. Sidan innehåller en tabell med sökfunktioner för existerande korthållartyper. Informationen som behövs för att lägga till korthållartyper är korthållarens namn, beskrivning och riktning(horisontellt eller vertikalt).

5.3.4 Korttypssidan

Från korttypssidan kan administratören skapa nya, radera och ändra korttyper. Sidan skall innehålla en tabell med sökfunktioner för existerande korttyper. Informationen som behövs för att lägga till korttyper är korttypens namn, beskrivning och riktning(horisontellt eller vertikalt). Då en ny korttyp skapas skall fälten för korttypen automatiskt skapas till databasen. Fältena skall vara inaktiverade efter att korttypen skapats, och om administratorn inte definierat och aktiverat fält för korttypen skall det inte vara möjligt att beställa kort för den korttypen. Istället visas en varningstext som berättar att korttypen inte är i bruk. För att aktivera fält bör administratören mata in alla obligatoriska fält för fältet. De obligatoriska uppgifterna för ett textfält är fältets namn, godkända tecken och minsta och största antalet tecken som krävs för fältet. För fält som definieras som ett beräknat fält är de obligatoriska uppgifterna fältets namn och vilket av de beräknade fälttyperna som kopplas till fältet.

5.3.5 Beräknade fält sidan

Från beräknade fält sidan kan administratörer lägga till och modifiera beräknade fält. Den information som behövs för ett fält är namn, beskrivning, och längd. Dessutom kan fältet ha ett prefix, men det är inte obligatoriskt.

5.3.6 Rapportsidan

Rapportsidan skall bestå av knappar med vilka administratörerna kan generera rapporter. Till den första versionen räcker det med två knappar, en för att generera en rapport om beställningarna för de olika korttyperna och en för att generera en rapport om användarna. I framtiden skall det vara möjligt att kunna lägga till knappar för andra rapporter vid behov.

5.3.7 Administreringssidan

Administreringssidan skall endast vara synlig för Superusen och skall visa en tabell med portalernas företag. Från sidan skall det vara möjligt att skapa nya och modifiera företag, lägga till användare till företagen och administrera virtuella kataloger. För varje företag skall det vara möjligt att bestämma vilka kataloger de kan använda för att ladda upp bilderna till korten. Om katalogen inte existerar eller webbapplikationen inte har tillgång till den skall en varningsbild visas bredvid den.

5.4 Migreringen av den gamla databasen till den nya

Eftersom den gamla versionen av HKNET innehåller information som måste överföras till det nya systemet krävs det att ett program utvecklas som kan läsa in informationen från den gamla databasen och modifiera det så att det kan skrivas in i den nya. Programmet skall vara en Windows applikation utvecklas med Visual Basic .NET. Eftersom migreringen är en engångsföreteelse skall programmet endast ha de funktioner som behövs för att flytta över informationen. Programmet skall som inputparametrar ta den gamla databasens tabell, korttypens ID nummer, information för att skapa kontakt med den nya databaserna, den nya tabellen och det nya identifieringsnumret för korttypen i det nya systemet.

5.5 Språkversioner

Eftersom en del av kundens beställare är bosatta utomlands skall webbapplikationens huvudspråk vara engelska, men det skall i framtiden vara möjligt att översätta applikationen så att användarna kan välja vilket språk de använder. Administratörerna skall vid skapandet av användare kunna bestämma vilket språk är användarens förinställda språk, och användaren själv skall ha möjlighet att ändra det vid behov. Förutom ASP.NETs egna lokaliseringsfunktioner bör det även finnas en databas eller XML fil till vilken man kan lägga till ord och meningar för andra språk. Språkversionerna utvecklas dock först efter att applikationen varit i bruk en tid för att försäkra sig om att applikationen funktionalitet uppfyller kundens krav.

5.6 Sökfunktioner inom webbgränssnittet

Kortmodulen, användarmodulen, korttypmodulen, korthållarmodulen och beräknade fält modulen skall innehålla sökfunktioner för att användaren snabbare skall hitta den information han söker efter. Sökfunktionen och fälten skall placeras på samma sida där själva informationen visas utan att användaren måste flytta sig till en skild sida. Varje sökfunktion består av flera sökfält, och sökfunktionens förfrågan skapas dynamiskt av den information användaren gett. Sökfälten skall alla bestå av textfält, förutom kortbeställningssidans beställnings- och utskriftsdatum som är textfält i datum format och en flervalstagg för i vilket skede beställningen är. Då användaren klickar på datumfälten skall en kalender visas varifrån användaren kan välja datumet, men det skall också vara möjligt att skriva datumet för hand. Sökresultatet skall innehålla endast de rader som uppfyller kraven sökfältens information skapat. Om ett sökfält är tomt betyder det att sökfunktionen inte skall beakta det fältet.

Ovanför varje tabell skall det finnas en flervalstagg från vilken användaren skall kunna välja antalet sökresultat som visas samt knappar för att förflytta sig till en annan sida av sökresultatet. Värden för flervalstaggen skall vara 10, 25, 50 och 100 för alla sidor, och det valda värdet då användaren kommer till sidan skall vara 10.

Användaren skall även ha möjlighet att sortera sökresultatet i stigande eller fallande ordning genom att klicka på rubriken under sökfältet.

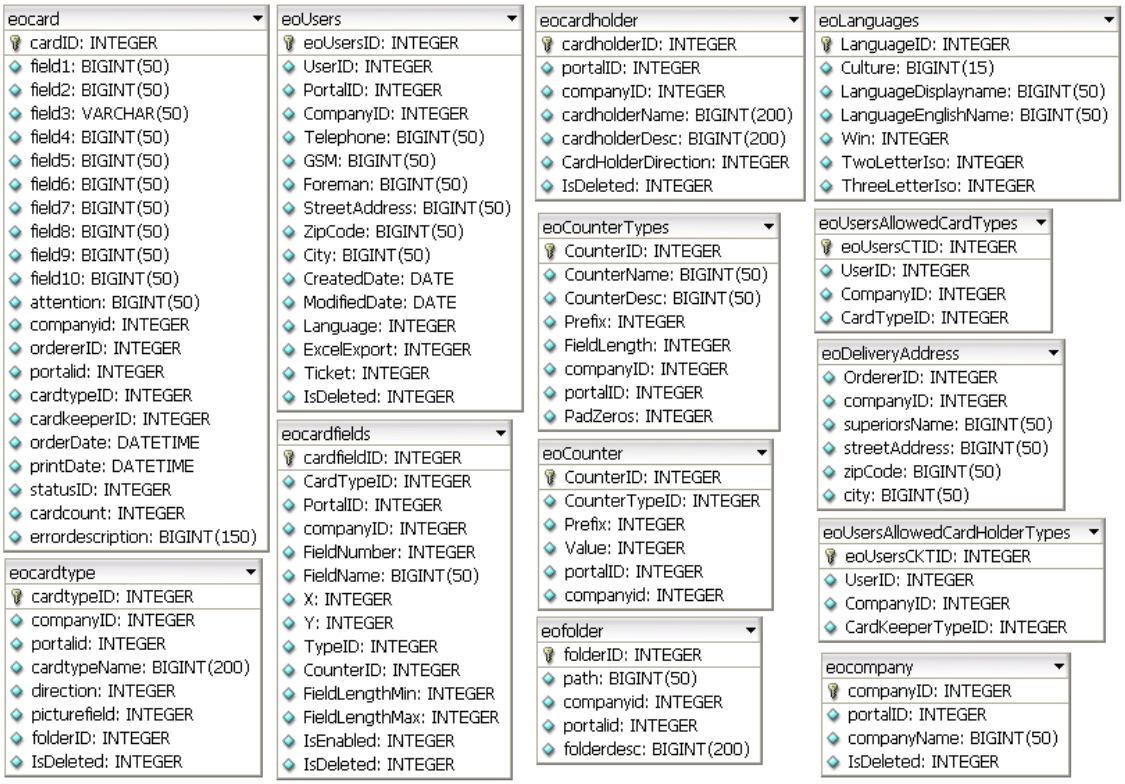
6 DEN TEKNISKA LÖSNINGEN

6.1 Databasen

Till projektet valdes som RDBMS Microsoft SQL Server 2008 Express eftersom det var den nyaste versionen som stöds av DotNetNuke och eftersom det var meningen att projektet skulle utföras så långt som möjligt med gratisversioner av utvecklingsverktyg. Efter att ha utvärderat kraven för databasen till kortbeställningsprogrammet visade det sig att Express versionens begränsningar dessutom var tillräckligt låga för att inte medföra problem.

Databasen består av DotNetNukes egna tabeller som skapas vid installationen samt 13 tabeller som innehåller data för själva kortbeställningsprogrammet(Tabell 2). Kortbeställningsprogrammets tabeller är relaterade till varandra och alla tabeller innehåller ett unikt identifieringsnummer som skapas då nya rader läggs till tabellen.

Tabell 2. Kortbeställningprogrammets tabeller



Användartabellen innehåller information om användarna, d.v.s. administratörerna och beställarna. Informationen består bl.a. av kontaktuppgifter, användarens språk och till vilken portal och företag användaren hör till. Användartabellen är relaterad till DotNetNukes användartabell som bl.a. innehåller användarens användarnamn, inloggningstider och e-post adress.

Korttabellen innehåller information om kortbeställningar. Tabellen har 10 fält reserverade för information som skall personifieras på korten. Korttyp ID-numret är relaterad till korttyptabellen och berättar vilken korttyp det är fråga om. Korttabellen har tre fält med information om beställaren, användarens ID, företagets ID och ett ID nummer till vilken portal beställaren hör till. Korttabellen innehåller också fält som berättar i vilket tillstånd beställningen är samt datum då kortet beställts och skrivits ut och hur många gånger kortet beställts. Tillståndnumret är 20 vid en ny beställning och byts till värdet 30 av utskrivningsprogrammet då kortet skrivits ut. Förutom tillståndskolumnen ändrar utskrivningsprogrammet även printdate kolumnens värde efter utskrivningen.

Korttyptabellen innehåller fält för att lagra information om korttyperna. Förutom fält som berättar till vilket företag och portal korttypen hör till har tabellen också fält för korttypens namn, kortets riktning (horisontellt eller vertikalt), vilket fält innehåller namnet på bilden samt till vilken katalog bilden lagras i.

Företagstabellen innehåller endast företagets namn och till vilken portal företaget hör till.

Till kortfälttabellen sparas information om korttypernas fält och vilken typ av information som kan sparas i kortfälten. Ett fält berättar vilket av de 10 kolumnerna fältet är, och vad fältet heter. Tabellen har koordinatfält för att möjliggöra att i framtiden kunna utveckla systemet och göra det möjligt att designa layouten på kortet via webbgränssnittet. För att kunna begränsa vad för information beställarna fyller i fälten har kortfältta-

bellen tre fält med information som applikationen använder för att begränsa och kontrollera informationen som lagras i korttabellen. De tre fälten är lägsta samt högsta antalet tecken samt ett ID nummer som definierar vilken sorts tecken är möjliga att skriva in. Om ett kortfält är länkat till ett beräknat fält sparas den beräknade fälttypens ID nummer i kortfälttabellen. Eftersom alla korttyper inte behöver 10 fält har kortfältstabellen även ett fält som berättar om kortfältet är aktiverat och syns för beställarna.

Korthållartabellen innehåller information om olika korthållare beställarna kan beställa till korten. Förutom korthållarens namn har tabellen ett fält för beskrivning om korthållaren och korthållarens riktning.

Beräknade fälttyp tabellen innehåller information om de beräknade fältens namn, en beskrivning om fältet och hur värdet för fältet beräknas. För beräkningen av fältets värde har tabellen tre kolumner, övre gränsen för värdet, hur många nollor läggs till förrän värdet samt en kolumn för ett statiskt värde eller en textsträng som läggs till förrän värdet.

Beräknade fält tabellen innehåller förutom ID nummer för att relatera tabellen till andra tabeller även själva värdet på fältet som skrivs in i databasen då kort beställs. Då ett nytt kort beställs som använder sig av samma beräknade fälttyp hämtas det senaste värdet från tabellen och ett nytt värde beräknas utgående från det senaste värdet.

Två tabeller finns till för att spara information om vilka korttyper och korthållare beställarna kan beställa. Dessa tabeller innehåller endast ID nummer för företaget, beställaren och korttypens och korthållarens ID.

I katalogtabellen sparas de katalogernas adresser till vilka kortbeställningarnas bilder sparas i. Tabellen har också identifieringsnummer för portalerna och företagen för att det skall vara möjligt att begränsa vilka företag har rättighet till katalogerna.

Leveransadress tabellen är en tabell till vilken man kan spara olika leveransadresser för företagens beställare. I ett senare skede av projektet beslöt man dock att inte ta denna funktion i bruk i den första versionen, men tabellen sparades för framtida versioner.

Språktabellen är en tabell som inte heller togs i bruk men som sparades för följande version. Tabellen innehåller fält i vilka man kan spara information om språk som applikationen i framtiden kan översättas till.

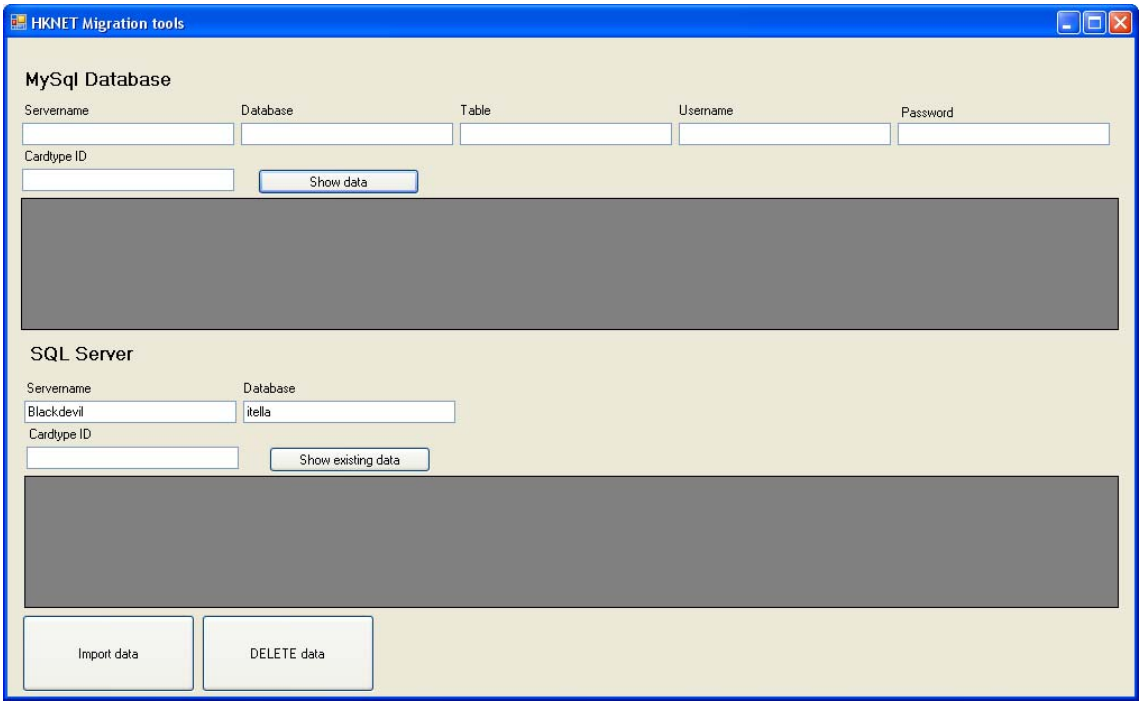
Förutom tabellerna innehåller databasen även lagrade procedurer innehållshanteringssystemet och kortbeställningsprogrammet behöver för att fungera. För kortbeställningssystemet skapades dryga sjuttio lagrade procedurer för att skriva in, radera, modifiera och hämta information från tabellerna.

6.2 Migreringen av den gamla databasen

Eftersom den gamla databasens struktur och business logiken i det gamla systemet skiljer sig märkbart från den nya kom man fram till att det lättaste sättet att flytta över informationen från det gamla systemet till det nya var att utveckla en Windows-applikation som skulle underlätta migreringen av datan. Applikationen skulle hämta informationen från de gamla databaserna, processera det och därefter skriva in informationen till den nya databasens tabell. Applikationen gjordes med Visual Basic .NET Express edition, och utnyttjade gratis komponenten MySQL connector/NET för att kommunicera med MySQL databaserna. MySQL connector är en ADO.NET drivrutin skriven i C# och utvecklad av företaget MySQL.

Migreringsverktyget består av ett användargränssnitt (Figur 3) från vilken man kan flytta över informationen från den gamla databasen till den nya. Användaren fyller i infor-

mationen som behövs för att skapa kontakt med databasen samt korttypens identifieringsnummer varefter användaren har möjlighet att flytta över informationen. Från applikationen är det även möjligt att se innehållet i tabellerna, och om överföringen misslyckas är det möjligt att radera informationen.

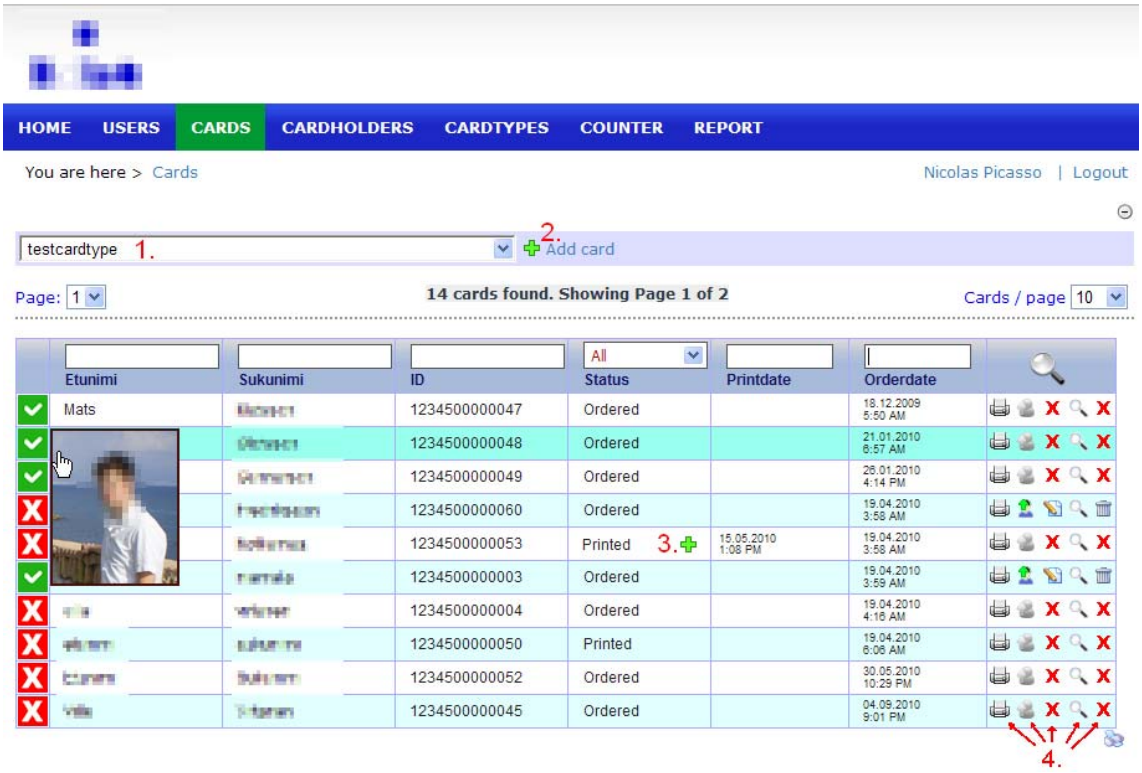


Figur 3. Migreringsverktygets användargränssnitt

6.3 Modulerna för användargränssnittet

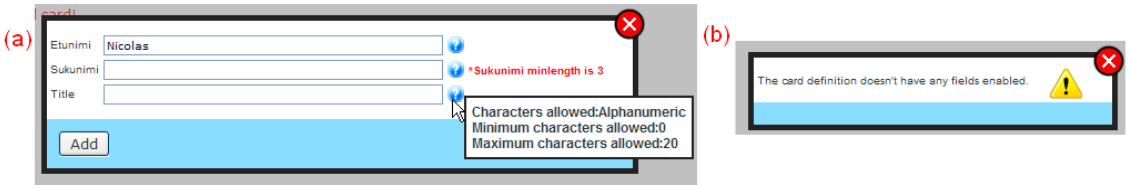
6.3.1 Beställningsmodulen

Kortbeställningsmodulens huvudvy (Figur 4) består av en tabell som visar de beställda korten och knappar för att göra nya och radera kortbeställningar, ändra gamla kortbeställningar, ladda upp bilder till korten, förhandsvisa korten och göra en utskrift av kortet. Överföringen av bilder och utskriftsfunktionen visas i skilda sidor, all annan funktionalitet visas i modala fönster.



Figur 4. Kortbeställningsmodulens huvudvy

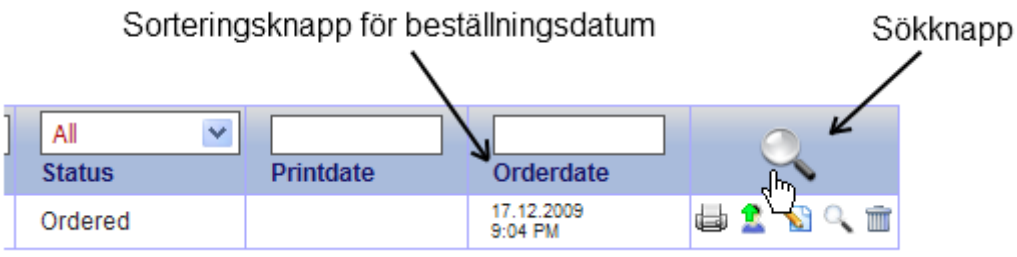
Från flervalstaggen (Figur 4, 1) kan beställaren välja korttyp. Flervalstaggen visar endast de korttyper användaren har rättighet att beställa. Då användaren byter korttyp från flervalstaggen skickas automatiskt en förfrågan till servern och tabellen under flervalstaggen uppdateras och visar den valda korttypens beställningar. Med ”Add Card” knappen (Figur 4, 2) kan användaren beställa ett nytt kort. Då användaren trycker på knappen visas ett modalt fönster (Figur 5, a) med ett formulär för att fylla i kortbeställningens information. Formuläret ligger i en Ajax uppdateringspanel, och förrän den visas för användaren uppdateras formulärets gränssnittskomponenter och information för den valda korttypen. Ifall administratören skapat en korttyp men inte aktiverat kortfält visas en modal varning för användaren(Figur 5, b).



Figur 5. Ett modalt fönster för att beställa kort (a) och ett varningsfönster (b)

Bredvid textfälten finns bilder som föreställer frågetecken. Då användaren flyttar muspekaren på bilden visas en inforuta som berättar kraven för fältets data. Efter att ha fyllt i fälten kan användaren trycka på ADD knappen för att spara beställningen. Förrän informationen sparas i databasen görs dock en kontroll att informationen i fälten fyller de krav som specificeras för korttypen. Om kraven inte uppfylls visas en röd varningstext vid sidan om fältet. Eftersom formuläret ligger i en AJAX uppdateringspanel behöver inte hela sidan laddas om vid kontrollen av informationen, utan endast innehållet i det modala fönstret uppdateras. Om informationen fyller kravena sparas informationen i databasen och den modala pop-upen stängs.

Ovanför tabellen finns gränssnittskomponenter (Figur 6) som visar antalet beställda kort för korttypen samt navigeringskomponenter (Figur 7) för att förflytta sig till en annan sida. En etikett visar antalet beställda kort och med två flervalstaggar kan användaren välja hur många kort som visas per sida samt förflytta sig till en annan sida.

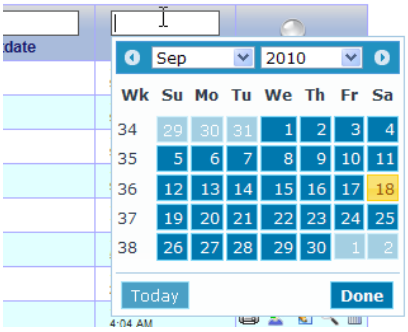


Figur 6 Gränssnittskomponenterna för sökfunktionen

Lägst nere i modulen finns tabellen som visar kortbeställningarna. Ovanpå tabellen finns textrutor för att kunna fylla i sökord för modulens sökfunktion(Figur 7). Då användaren trycker på bilden som föreställer ett förstoringsglas utför servern en sökfunktion och tabellen fylls med de kortbeställningar som fyller sökkravet. Under textrutorna finns en kolumnrubrik. Genom att trycka på den kan användaren välja enligt vilket fält tabellen skall sorteras, och ifall den skall sorteras i stigande eller fallande ordning.

Figur 7. Navigeringskomponenter för tabellen

Två av textfälten är datumfält. Då användaren trycker på datum-textfält visas en kalender från vilken användaren kan välja ett datum. Det är också möjligt att manuellt skriva datumet. Kalendern är realiserad med ”jQuery UI”-gränssnittskomponenten Datepicker (Figur 8).



Figur 8. ”jQuery UI”-Datepicker kalenderkontrollen

”jQuery UI”-gränssnittskomponenten Datepicker är konfigurerbar och har flera egenskaper andra kalenderkontroller saknar, bl.a. möjlighet att bestämma i vilken form datumet visas, lokalisering av kalendern, möjlighet att förflytta sig i kontrollen med tangentbordet samt möjlighet att visa flera månader samtidigt.

Kalendern initialiseras i jQuery händelsen $(document).ready() genom att kalla på funktionen InitCal. $(document).ready() funktionen körs då sidans DOM är färdigt, förrän sidan och dens innehåll har laddats. InitCal funktionen skapar jQuery objekt från sidans DOM av de element som har CSS-klassen datePicker och fäster en kalenderkontroll till dem. InitCal funktionen innehåller även inställningar för kalenderns egenskaper.

```
$(document).ready(function() {

    InitCal();

});

function InitCal() {
    $(".datePicker").datepicker
                        ({
                            changeMonth: true,
                            changeYear: true,
                            dateFormat: 'dd.mm.yy',
                            showOtherMonths: true,
                            showStatus: true,
                            showButtonPanel: true,
                            showWeek:true
                        });
}
```

Textrutorna för utskrivningsdatumet och beställningsdatumet behöver bara ges CSS-klassen datePicker i ASCX filens markup för att textrutorna skall få kalenderfunktionaliteten:

```
<asp:TextBox ID="tbPrintDate" CssClass="datePicker" runat="server" />
```

Om korttypen har ett bildfält definierat finns det längst till vänster i tabellen en bild som berättar om en bild har laddats upp till servern för kortet. En grön bild visar att kortbeställningen har en bild och en röd att bild saknas. Om användaren för muspekaren på en grön bild visas en förhandsbild i ett poppuppfönster bredvid muspekaren (Figur 9). Poppuppfönstret stängs automatiskt då muspekaren förs bort från bilden.

Figur 9. Förhandsbild för kortbeställningens bild

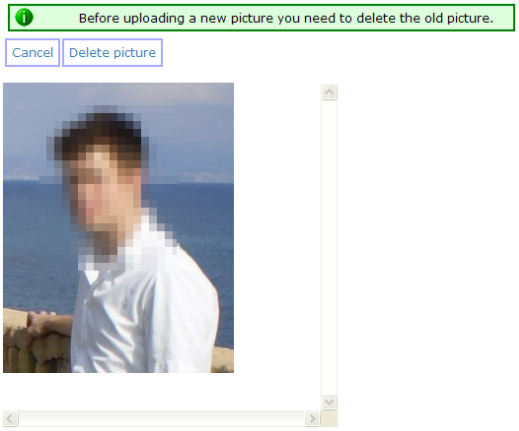
Genom att klicka på förhandsvisningsbilden laddas bilden i sin riktiga storlek och visas i ett modalt poppupfönster (Figur 10). Detta fönster har inga andra funktioner än att visa bilden. Fönstret är en ”jQuery UI”-dialog gränssnittskomponent. Fönstret använder sig av ”jQuery UI”-interaktionsfunktionerna Draggable och Resizable. Funktionerna gör det möjligt att flytta på fönstret samt ändra dess storlek.

Figur .10. ”jQuery UI”- Dialog för visning av kortets bild

Efter att ett kort från en kortbeställning har skrivits ut ändras kortets status från ”ordered” till ”printed”. Om kortet är i ”printed” status och användaren är beställaren av kortet finns det en bild av ett plus (Figur 4, 3) bredvid statustexten. Genom att trycka på plustecknet har användaren möjlighet att beställa kortet på nytt. Efter att ha tryckt på knappen visas en modal bekräftelseruta för användaren för att försäkra sig om att användaren vill beställa kortet på nytt (Figur 11).

Figur 11. Bekräftelseruta för nybeställning

Till höger om tabellen finns fem knappar(Figur 4, 4). En knapp för att visa kortets information i en utskriftsvänlig funktion, en för att ladda upp en bild till kortet, en för att ändra på kortets information, en för att visa en förhandsversion av kortet och en för att radera kortet. Om användaren inte är den ursprungliga beställaren av kortet har han inte möjlighet att ladda upp en bild för kortet, ta bort kortet eller modifiera kortets uppgifter. I så fall är bilderna för funktionerna ersatta med ett rött kryss. Om beställaren trycker på bilduppladdningsknappen gömmer modulen alla andra element i modulen och visar gränssnittet för uppladdningen av bilden. Om en bild redan har laddats upp till servern har användaren möjlighet att radera den gamla bilden och ladda upp en ny bild (Figur 12).



Figur 12.

Efter att ha raderat den gamla bilden eller om kortbeställningen inte hade en bild på servern visas kontroller (Figur 13) med vilken användaren kan ladda upp en bild till ser-

vern. Ifall användaren laddar upp en fil som inte är en JPG-fil visas en text för användaren som meddelar att filen är i fel format. Efter att ha laddat upp en bild visas en panel med gränssnittskomponenter för att kunna rotera och förminska bilden på servern En funktion för att kunna beskära bilden realiserades med jCrop komponenten, men togs inte i bruk i första versionen av Henkilökortti .net 2.0.

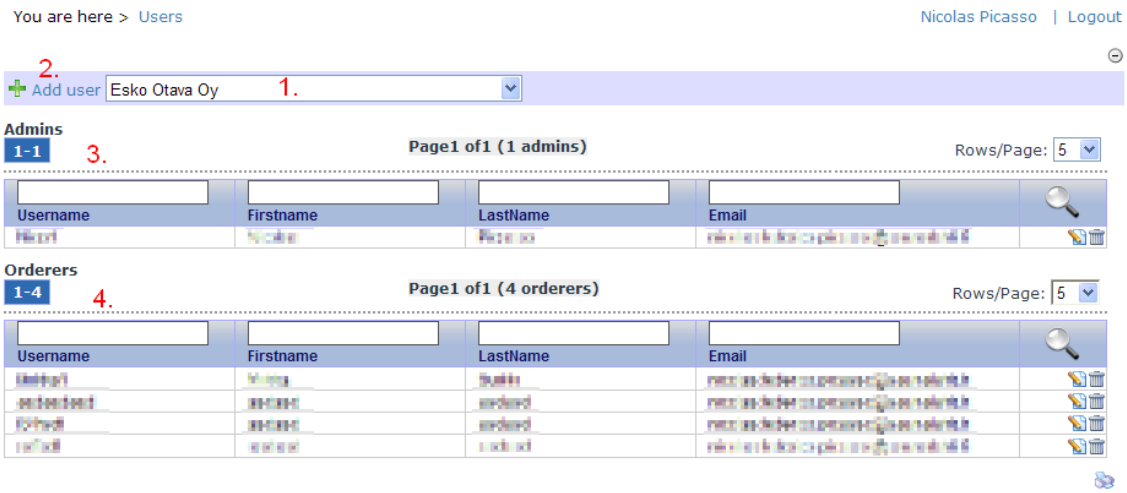
Figur 13. Användargränssnittets bildbehandlingsfunktioner före (a) och efter rotering och förminskning av bilden (b)

Två knappar finns till för att rotera bilden 90 grader medsols eller motsols. Bredvid knapparna för att rotera bilden finns en flervalstagg och en knapp för att förminska bilden. Från flervalstaggen kan användaren välja procentgraden bilden skall förminskas. Procentvärdena går från 30 till 90 med steg på tio. Efter bildens uppladdning och behandling kan användaren spara bilden eller avbryta, varefter kortbeställningsmodulens huvudvy visas.

Knappen för modifiering av kortets information visar en modal popup som ser likadan ut som med den man gör en ny beställning. Skillnaden är att kortets information fylls i formulärets textfält varefter användaren kan ändra på informationen och spara den nya informationen i databasen genom att trycka på ”Save” knappen.

6.3.2 Användarmodulen

Användarmodulens huvudvy består av två paneler och två tabeller som visar företagets användare (Figur 14).

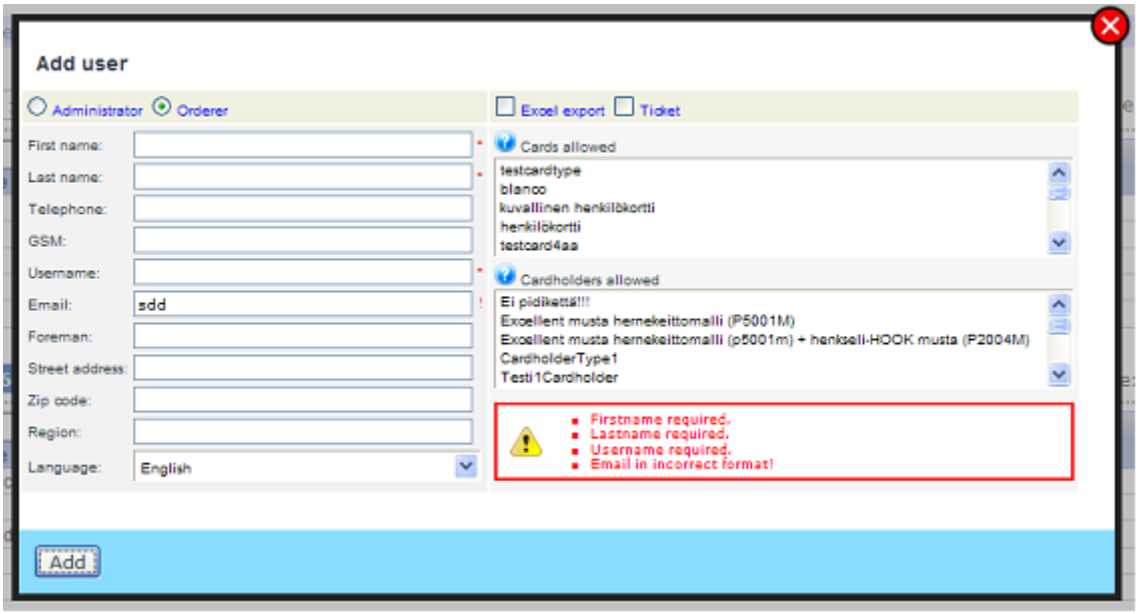


Figur 14. Användarmodulens huvudvy

Ifall användaren är portalens Superuser finns det i panelen en flervalstagg (Figur 14,1) med portalens företag, från vilken Superuser användaren kan välja vilket företags användare han vill administrera. Om användaren är en administratör ser han endast sitt eget företags användare. Huvudvyn innehåller två tabeller, en som visar företagets administratörer (Figur 14, 3) och en som visar företagets beställare (Figur 14, 4). Båda tabellerna har gränssnittskomponenter för sidans sökfunktion som fungerar på samma sätt som kortmodulens sökfunktion. Sökfältena består av användarnamnet, förnamnet, efternamnet och e-post adressen.

Från panelen är det möjligt att skapa nya användare genom att trycka på ”Add user” knappen (Figur 14, 2) varefter ett modalt popuppfönster visas för användaren. Popupen innehåller ett formulär med textfält för att fylla i användarens personuppgifter och flervalstaggar för att välja användarens språk samt vilka korttyper och korthållartyper han kan beställa. Från panelen kan man via två envalsknappar välja om användaren hör till rollgruppen beställare eller administrator. Från flervalstaggarna för tillåtna korthållare och korttyper kan man välja flera alternativ genom att hålla i ctrl-tangenten då man väljer en rad. Panelen innehåller valideringskontroller för de fält som är obligatoriska. Om användaren glömmer att fylla i fältena eller informationen i fältena inte uppfyller valideringsfunktionernas krav visas en tabell med de felaktiga informationsfält som bör korri-

geras för att skapa användaren. Om användaren har fyllt i informationen korrekt och trycker på ”Add” knappen kontrollerar applikationen om användarnamnet redan existerar. Om det existerar visas en text för användaren att användarnamnet redan är i bruk och ber användaren att ge ett nytt. Om det inte existerar skapas ett konto för den nya användaren (Figur 15) och en e-post skickas till användaren med hans användarnamn och lösenord.



Figur 15. Poppupfönster för att skapa nya användare.

I den sista kolumnen i användartabellerna finns det två bildknappar ifall användaren är en administrator och tre bildknappar ifall användaren är Superuser.

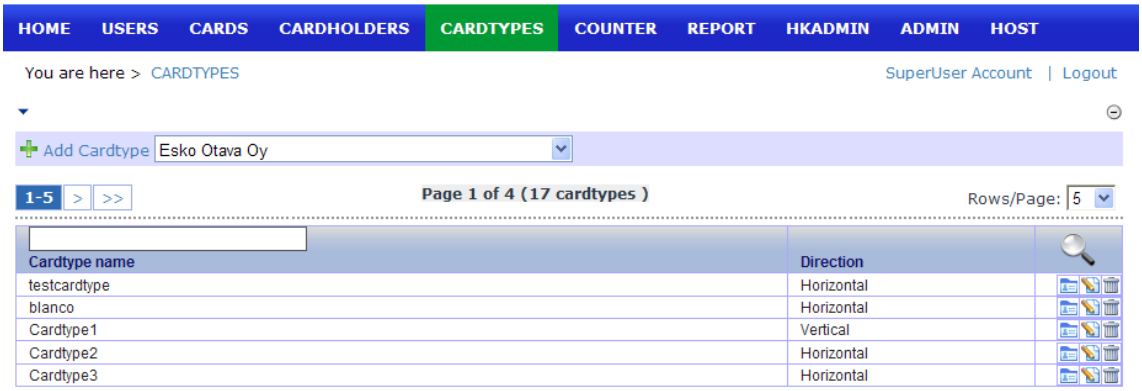
Figur 16. Knappar för flytta över beställningar, redigera användaruppgifter samt radera användare

Superusern har möjlighet att flytta över en användares beställningar till en annan användare (Figur 16). Genom att trycka på bilden av en grön pil öppnas ett poppupfönster från vilken Superusern kan välja till vilken användare beställningarna flyttas. Bilden av en penna och ett papper öppnar en popup från vilken Superusern eller administratören kan ändra på personens uppgifter. Popupen ser ut som formuläret för att skapa användare

men har användarens uppgifter färdigt laddade. Användarnamn-fältet är ersatt med en etikett eftersom det inte skall vara möjligt att ändra på användarnamnet. Genom att klicka på bilden av en skräpkorg kan man radera användaren.

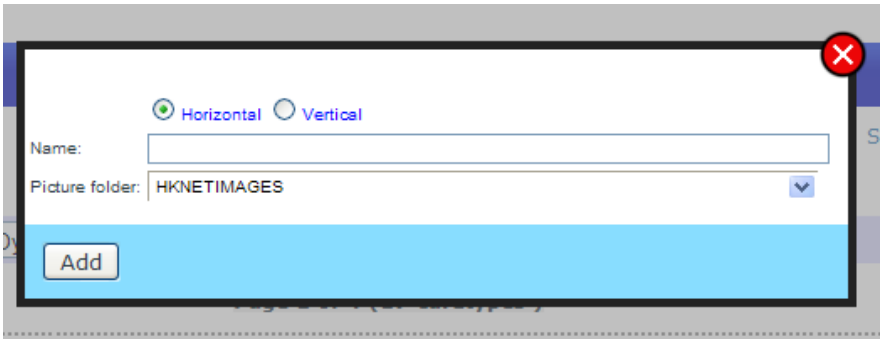
6.3.3 Korttypmodulen

Från korttypmodulen kan administratörer och Superusern skapa nya och modifiera gamla korttyper. Modulens huvudvy (Figur 17) består av en knapp för att skapa en ny korttyp och en tabell som visar existerande korttyper för företaget.



Figur 17. Korttypsmodulens huvudvy

Genom att trycka på ”Add Cardtype” öppnas ett modalt fönster för att lägga till korttyper (Figur 18). Fönstret innehåller två envalsknappar för att välja kortets riktning, en textruta för korttypens namn samt en flervalstagg för att välja till vilken katalog korttypens bilder sparas till.



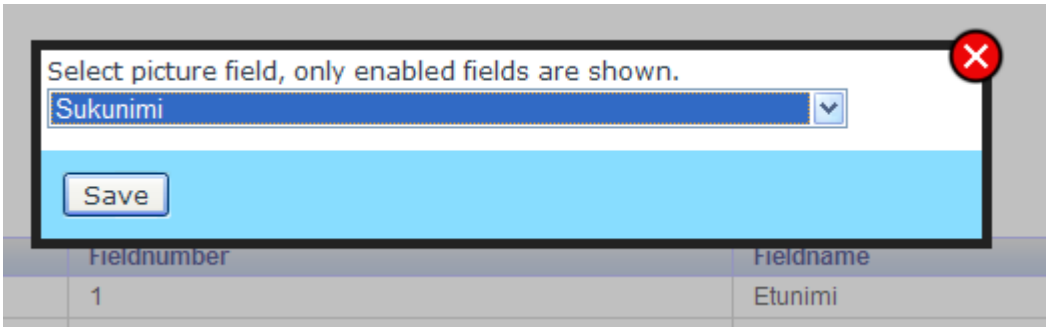
Figur 18. Formuläret för tilläggning av korttyper

Korttypstabellen kan sorteras enligt namn eller riktning, och sökfunktionen är begränsad till korttypens namn. Korttypstabellens rader har tre knappar. En för att radera korttypen, en för att modifiera korttypen och en för att definiera korttypens fält. Då användaren trycker på kortfältsknappen visas en sida med korttypens fält (Figur 19).

Enabled	Fieldnumber	Fieldname
☑	1	Etunimi
☑	2	Sukunimi
☐	3	Counter
☑	4	IDNumber
☑	5	TEXT
☐	6	
☑	7	Counter2
☐	8	
☐	9	
☐	10	Field10

Figur 19. Sida för hantering av korttypens fält

Högst uppe på sidan finns en knapp för att bestämma vilket fält fungerar som bildfält. Om man trycker på knappen visas ett modalt fönster med en flervalstagg (Figur 20). Flervalstaggen innehåller alla fält för korttypen som är aktiverade.

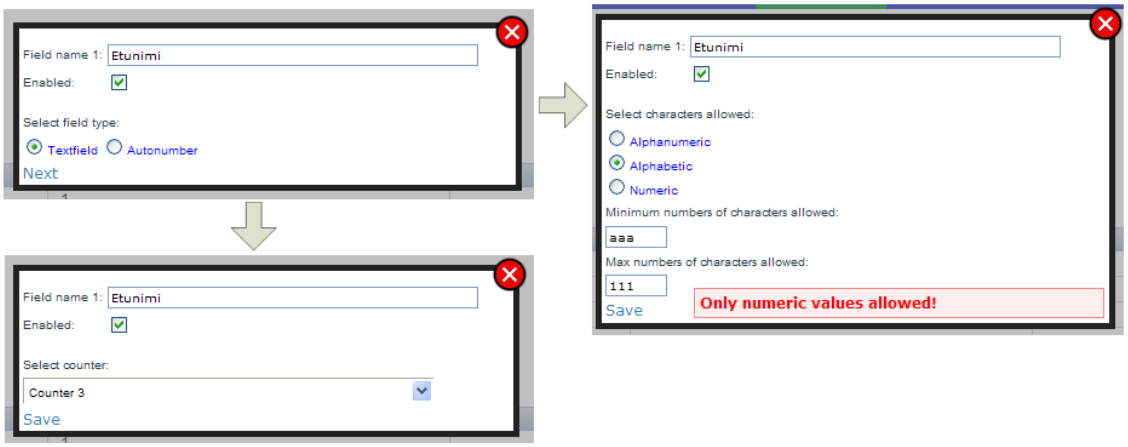


Figur 20. Fönster för att välja korttypens bildfält

Efter att ha valt ett fält och tryckt på ”Save” knappen sparas informationen i databasen. Då bilder laddas upp till servern via kortbeställningsmodulen namnges bildfilens enligt den information som finns i det fält som definieras som bildfält.

I tabellen för kortfältena finns en knapp för att kunna välja om fältet är aktiverat samt vilka begränsningar och krav fältet har. Knappen öppnar ett modalt fönster (Figur 21) från vilket man kan ge namn åt fältet och bestämma om det är aktiverat. Från två envalsknappar kan man välja om fältet är ett textfält eller beräknat fält.

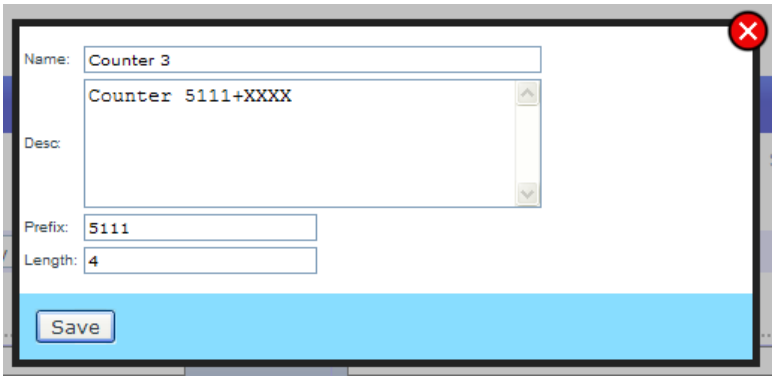
Om man väljer textfält och trycker på ”Next” uppdateras fönstret varefter man från tre envalsknappar kan bestämma om innehållet för fältet kan vara alfanumeriskt, alfabetiskt eller numeriskt. Två textrutor tar indata för största och minsta antalet tecken fältet tillåter. Väljer man beräknat fält uppdateras rutan och från en flervalstagg kan man välja vilken beräknade fält definition man använder för fältet. Webbapplikationen kontrollerar att informationen som getts är valid och visar en varningstext ifall informationen som getts är bristfällig eller fel.



Figur 21. Definiering av fälttyp

6.3.4 Beräknade fältmodulen

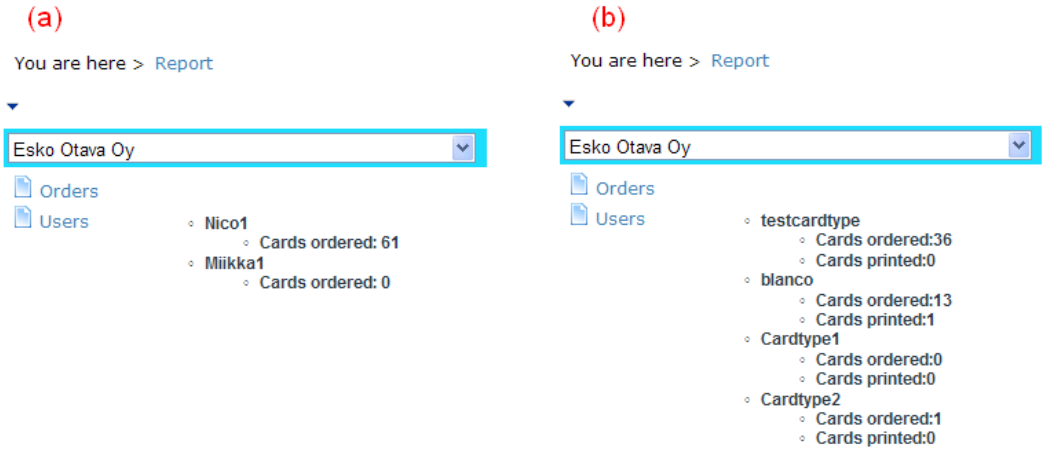
Gränssnittet för beräknade fält modulens är likt de andra modulerna. En tabell visar de beräknade fält som redan finns, och med knappen ”Add counter” kan man skapa nya beräknade fält. Sök- och sorteringsfält för modulens tabell är det beräknade fältets namn och beskrivning. Popp-upfönstrer för att skapa ett nytt eller ändra på ett färdigt beräknat visas i Figur 22. Data som krävs för att skapa ett beräknat fält är namn och längd. Valfri data är en beskrivning om fältet samt ett prefix för fältet. Längden på fältet berättar hur många siffror fältets värde skall innehålla. Ifall fältets värde inte innehåller tillräckligt många siffror lägger programmet nollor före värdet för att längden skall uppnås. Det första värdet som genereras med ett beräknat fält med prefixen 5111 och längden 4 skulle vara 51110001. I den första versionen börjar räknarna alltid från ett, men i framtida versioner av webbapplikationen skulle en funktion för att göra det möjligt att bestämma vilket värde som är det första som genereras vara praktisk.



Figur 22. Fönster för modifiering av ett beräknat fält

6.3.5 Rapportmodulen

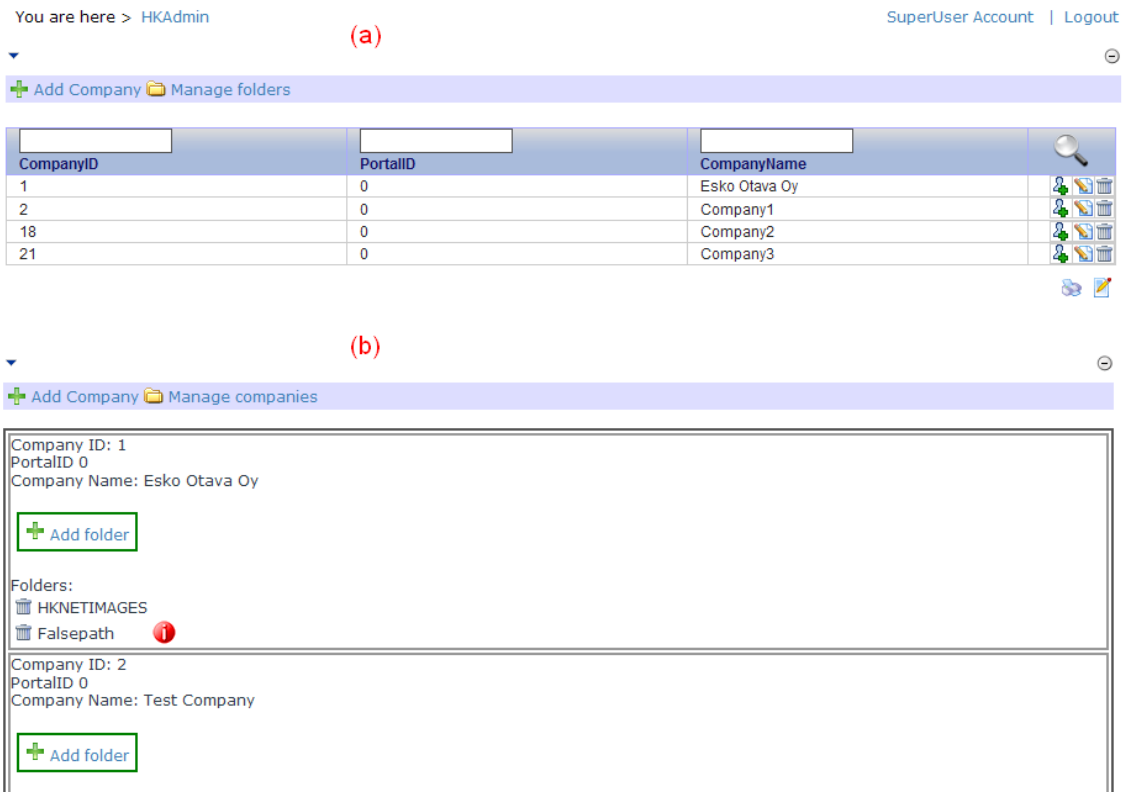
Rapportmodulen i den första versionen av Henkilökortti .net 2.0 består endast av två rapportbotten (Figur 23). Användarrapporten visar företagets användare samt hur många kort de beställt. Beställningsrapporten visar hur många kort beställts av varje korttyp, samt hur många av dem som redan skrivits ut.



Figur 23. Rapportmodulen

6.3.6 Administreringsmodulen

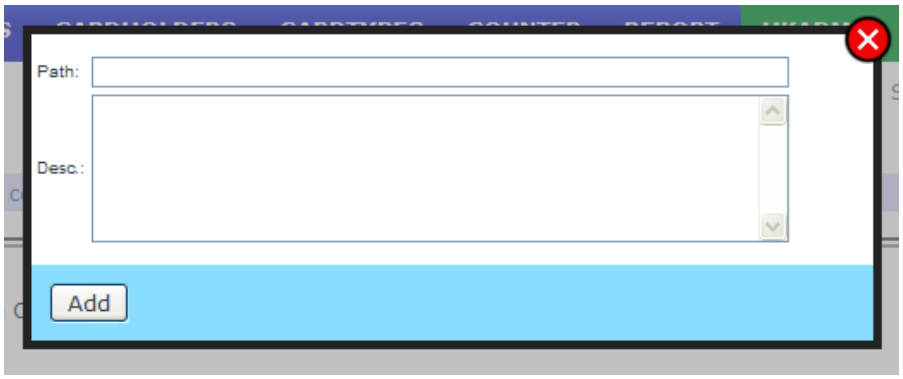
Administreringsmodulen är den enda modulen endast Superuser har tillgång till. Modulen är också den första som används i kortbeställningssystemet eftersom man med den skapar det första företaget och användaren. Modulen består av två sidor, en för att administrera företag (Figur 24, a) och en för att administrera kataloger (Figur 24, b).



Figur 24 Administreringssidan för företag (a) och kataloger (b)

Sidan för administrering av företag visar en tabell med kortbeställningssystemets företag. Administratorn har möjlighet att skapa användare till företaget genom att klicka på bilden som föreställer ett ansikte i kolumnen längst till höger. Formuläret för att skapa en ny användare ser likadant ut som formuläret i användarmodulen.

Katalogadministreringssidan visar alla företag i en tabell samt alla kataloger företagen har rättighet att ladda bilder till. Ifall katalogen inte existerar visas en varningsbild bredvid katalognamnet. Knappen ”Add folder” öppnar ett formulär i ett modalt fönster (Figur 25). Från formuläret kan administratorn skriva in sökvägen för virtuella kataloger han skapat i IIS. Genom att klicka på bilden av en soptunna bredvid katalogerna kan administratorn radera kataloger.



Figur 25. Fönster för att lägga till virtuella kataloger

6.4 Datasäkerhet inom webbapplikationen

6.4.1 Allmänt om datasäkerhet

Med ordet datasäkerhet hänvisar man till skydd av data, system, tjänster och kommunikation. Det har debatterats om vilka krav som skall uppfyllas för att ett datasystem skall vara säkert. Ofta räknas de tre kraven i CIA (confidentiality, integrity, availability) modellen om sekretess, integritet och tillgänglighet som kärnprinciperna inom datasäkerhet:

- Sekretess: informationen skall inte vara tillgänglig till obehöriga personer.
- Integritet: informationen i datasystemet skall vara riktiga och inte förändrade av obehöriga
- Tillgänglighet: informationen måste vara tillgänglig när den behövs

Datasäkerhet är ett mycket brett ämne och i detta kapitel berättas kort om datasäkerhet inom webbapplikationer.

6.4.2 Vanliga säkerhetsrisker i webbapplikationer

Datasäkerhetsrisker relaterade till webbapplikationer brukar delas in i två grupper, datasäkerhetsrisker i webbservern och datasäkerhetsrisker i webbapplikationens logik. Vanliga sårbarheter i webbservern kan vara:

- Exempelkod för webbservern som inte raderats från produktionsservern
- Möjlighet att upptäcka applikationens källkod
- Servertillägg med sårbarheter
- Buffertspill attacker

(McClure m.fl., 2009 : 544-546)

Open Web Application Security Project (OWASP) är en organisation inriktad på att förbättra säkerheten i programvara. OWASP uppehåller en topp 10 lista över datasäkerhetsrisker inom programvara. I deras lista för 2010 toppar bl.a. följande sårbarheter i webbapplikationer:

- Injektionssårbarheter
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)

Injektionsattacker

Injektionsattacker är attacker var man utnyttjar sårbarheter i valideringen av indata för att infoga kod som ändrar logiken hur applikationen behandlar indata. En vanlig injektionsattack är SQL-injektion. SQL-injektioner går ut på att attackeraren kan ändra SQL satsen som körs genom att injicera kod i indata. Indata kan bestå av data från t.ex. textrutor eller parametrar i URL:ens frågesträng. Ett exempel på en sårbar sats är

```
Dim myCommand As New SqlDataAdapter("SELECT name FROM users WHERE socnumber = "' + textbox.Text & "'", myConnection)
```

Denna kod skapar en SQL fråga som bildas dynamiskt och är sårbar för SQL-injektion. Satsen förväntar sig en parameter som hämtas från en textruta i webbsidan. Om textrutan skulle innehålla texten ”' ; DROP DATABASE tabellnamn --” skulle SQL satsen avbrytas efter ’ tecknet och ”DROP DATABASE users” kommandot skulle utföras, vilket skulle radera databasen users. Genom SQL-injektioner är det möjligt att utföra at-

tacker med olika resultat. Förutom att radera databaser är det möjligt att komma åt information som inte borde vara tillgängligt, kringgå autentisering eller i värsta fall få full tillgång till DMBS systemet eller webbservern. (McClure m.fl., 2009 : 573-576)

Cross-Site Scripting

Cross-Site Scripting eller XSS är attacker som utnyttjar sårbarheter i valideringen av indata och utdata i applikationen. XSS attackernas mål riktar sig sällan mot själva applikationen utan mot andra användare av applikationen. Då en användare laddar en sida som har blivit infogad med uppsåtlig kod utförs koden i användarens webbläsare. Koden kan göra det möjligt för attackeraren att t.ex. stjäla användarens sessions, installera en trojansk häst på användarens maskin eller dirigera användaren till en annan sida. (McClure m.fl., 2009 : 571-573)

Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF) är datasäkerhetsrisk som utnyttjar webbapplikationers förtroende för användaren. Första delen av attacken går ut på att attackeraren lyckas få offret att besöka en sida som innehåller koden för att genomföra CSRF attacken. Detta kan ske genom att infoga kod i en webbsajt användaren brukar besöka, eller genom att lura användaren till att klicka på en länk i en e-post som för användaren till en sida med den farliga koden. Ifall offret besöker sidan exekveras koden och en förfrågan skickas till en webbapplikation, t.ex. en webbaffär, som har en CSRF sårbarhet. Om användaren är inloggad eller affären använder ihållande cookies kan förfrågan utföra en funktion i affären, t.ex. köpa produkter. Eftersom själva koden kan vara dold i t.ex. en bildtagg eller i en Javascript funktion märker offret inte att hans webbläsare utfört handlingen. CSRF kräver inte att attackeraren vet någonting om användaren. Efter att ha lyckats skapa CSRF kod som utnyttjar webbapplikationens sårbarhet räcker det med att han lyckas injicera koden till en webbsajt eller skickar e-post som innehåller CSRF koden via en spam server. Alla användare som sedan besöker webbsajten eller klickar på e-postens länk och som är inloggade eller har en ihållande cookie i sin dator kan bli offer för CSRF attacken. (McClure m.fl., 2009 : 576-578)

6.4.3 Datasäkerhetslösningar

Att göra webbapplikationer säkra kan vara en svår uppgift. Ju större interaktionsmöjligheter applikationen har desto större blir attackytan för att utnyttja sårbarheter. Vid utveckling av webbapplikationer är det viktigt att redan tidigt börja bygga in säkerhet. Alla funktioner i applikationen som användaren kan interagera med skall planeras bra och testas för att hitta brister och sårbarheter. En viktig regel för att skydda sig mot attacker är att aldrig lita på användarna och att alltid kontrollera den indata de skickar. Nedan följer några exempel på hur man kan skydda sig mot vanliga datasäkerhetsrisker.

Injektionsattack-lösningar

Begränsningar och sanering av indata minskar risken för injektionsattacker. Alla tecken som kan vara del av ett skript skall förkastas eller saneras om möjligt. Farliga tecken kan vara t.ex.: < > ( ) # & ”. Ibland kan dock farliga tecken vara en del av giltig indata. Att skriva funktioner som analyserar och sanerar indatan korrekt kan därför vara en krävande uppgift. Egenskaper av indatan som lönar sig att kontrolleras är:

- Är indatan av rätt datatyp? Fält som endast borde innehålla siffror skall inte innehålla bokstäver eller specialtecken.
- Är längden av indata korrekt? Genom att begränsa antalet tecken indata kan innehålla minskar man på risken att datat innehåller skadlig kod.

För att förhindra SQL-injektioner lönar det sig att använda av parametriserade frågor och parametriserade lagrade procedurer istället för att skapa SQL-satserna dynamiskt. Parametriserade frågor och lagrade procedurer behandlar indata som parametrar och förhindrar att användaren kan köra egen SQL kod på servern. Ifall valideringen av indata inte lyckas och ett fel sker i programmet bör man undvika att ge detaljerade felmeddelande. Detaljerade felmeddelanden kan avslöja information till användaren han kan utnyttja för att hitta potentiella säkerhetsrisker.

Cross-Site Request Forgery-lösningar

Ett sätt att skydda sig mot CSRF i webbapplikationer är att implementera det s.k. ”Synchronizer Token Pattern” mönstret i sidorna. Lösningen går ut på att webbapplikationen genererar slumpmässiga värden som sparas i sessionen och webbsidorna. När en förfråga skickas till servern skickas även det slumpmässiga värdet och webbapplikationen kan kontrollera att värdet existerar i användarens session. Om värdet saknas eller inte stämmer kan applikationen be att användaren autentiserar sig på nytt.

7 TESTNING

7.1 Testning för säkerhetsrisker och prestanda

Eftersom projektet ännu var vid testfasen då detta examensarbete skrevs hann man inte utföra en mera utförlig test för applikationen. Dock utfördes tester i mindre skala för att försäkra sig om att det inte fanns större säkerhetsrisker eller prestandaproblem.

Testning av säkerhetsrisker utfördes genom att kontrollera att all användarindata granskas innan det sparas i databasen. Genom att manuellt försöka utföra SQL-injektioner och XSS attacker på alla sidor varifrån indata skickas till servern och sparas i databasen kunde man analysera resultatet och finna möjliga brister.

För att analysera och förbättra applikationens prestanda analyserades tre delar av projektet, databasens förfrågningar, källkodens funktioner och HTML koden som genereras av webbservern till användaren.

SQL Serverns express version saknar tilläggsprogrammet Profiler som gör det lätt att analysera förfrågningarna som körs på servern. För att kunna spåra förfrågningarna som kördes under användningen skapades ett T-SQL skript som startade en spårning på SQL servern och skrev ut resultatet i en loggfil (Bilaga).

För att få resultat som skulle påminna om resultaten från en produktionsserver genererade man testdata till tabellerna. Efter att spårningen pågått en tid och applikationens funktioner använts körde man ett T-SQL skript som läste loggfilens resultat till en tabell. Från tabellen kunde man se vilka av applikationens lagrade procedurer körts och hur länge exekveringen av förfrågningarna tog.

För att analysera källkodens prestanda gjordes funktioner för att mäta tiden det tog för funktioner att utföras. Loggfunktionerna kallades före och efter funktionernas anrop och skrev i en loggfil start- och sluttidpunkten. Om det visade sig att något funktionsanrop tog betydligt längre än andra funktionsanrop gick man igenom koden och kontrollerade om det berodde på att funktionen verkligen utförde mera komplexa beräkningar eller om det fanns kod som var dåligt optimerad.

HTML koden som genererades av webbservern från ASP .NET sidorna och skickas till användaren försöktes få så kompakt som möjligt. För att åstadkomma detta användes bl.a. ASP .NETs egen spårningsfunktion. ASP .NETs spårningsfunktion skriver i slutet av webbsidan en rapport som bl.a. innehåller information om webbsidans kontroller och hur många byte de tar (Figur 26). Rapporten innehåller även information om sidans Viewstate, som är en datakatalog som sparar objekt för att kunna spara bl.a. sidans visuella tillstånd. Från rapporten kan man se vilka objekt på sidan som tar mycket utrymme och möjligtvis bör optimeras.

Request Details

Session Id:	ah4grjav0z2ewa45qjymoj45	Request Type:	GET
Time of Request:	9/16/2010 1:50:33 AM	Status Code:	200
Request Encoding:	Unicode (UTF-8)	Response Encoding:	Unicode (UTF-8)

Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	3.04507975175616E-05	0.000030
aspx.page	Begin Init	5.92254043460831E-05	0.000029
aspx.page	End Init	0.00951405835099154	0.009455
aspx.page	Begin InitComplete	0.00958250280412734	0.000068
aspx.page	End InitComplete	0.00960932185515198	0.000027
aspx.page	Begin PreLoad	0.00963027423876498	0.000021
aspx.page	End PreLoad	0.00968866154776655	0.000058
aspx.page	Begin Load	0.00971240758252795	0.000024
aspx.page	End Load	0.025316346071917	0.015604
aspx.page	Begin LoadComplete	0.0253861873506627	0.000070
aspx.page	End LoadComplete	0.0255026826035153	0.000116
aspx.page	Begin PreRender	0.0255330060384769 6	0.000027
aspx.page	End PreRender	0.0549108895124939	0.029381
aspx.page	Begin PreRenderComplete	0.0549810101563188	0.000070
aspx.page	End PreRenderComplete	0.0733945236056538	0.018414
aspx.page	Begin SaveState	0.153868789856239	0.080473
aspx.page	End SaveState	0.155687105484077	0.001819
aspx.page	Begin SaveStateComplete	0.155743537237275	0.000056
aspx.page	End SaveStateComplete	0.155767842002266	0.000024
aspx.page	Begin Render	0.155790470576568	0.000023
aspx.page	End Render	0.30403724495711	0.148247

Control Tree

Control UniqueID	Type	Render Size Bytes (Including children)	ViewState Size Bytes (excluding children)	ControlState Size Bytes (excluding children)
__Page	ASP.default_aspx	157688	0	0
skinDocType	System.Web.UI.WebControls.Literal	121	180	0
Head	System.Web.UI.HtmlControls.HtmlHead	2165	0	0
ctl09	System.Web.UI.LiteralControl	459	0	0
ctl00	System.Web.UI.HtmlControls.HtmlMeta	69	0	0
ctl01	System.Web.UI.HtmlControls.HtmlMeta	67	0	0
ctl02	System.Web.UI.HtmlControls.HtmlMeta	59	0	0
MetaRefresh	System.Web.UI.HtmlControls.HtmlMeta	0	20	0

Figur 26. Utskrift från ASP .NET spårningsrapport

7.2 Resultat och åtgärder

Efter att ha testat och analyserat webbapplikationens kod visade det sig att det inte fanns några större delar av koden som skulle behöva optimering. Eftersom dagens servrar är kraftfulla och webbapplikationen inte utförde mera komplexa beräkningar ansågs det att källkodens beräkningsfunktioner fungerade tillräckligt effektivt. Däremot visade SQL serverns spårning att onödiga anrop till databasen gjordes, och genom att analysera koden kunde man optimera koden och kombinera anrop för att minska på antalet databasanrop som måste göras.

Spårningen av SQL servern visade också att kortbeställningsmodulens anrop kunde optimeras. ASP .NET:s kontrollen GridView är en kontroll för att rendera information i ett rutnät med kolumner och rader. GridView kontrollen har automatiskt stöd för sidbrytning och sortering samt stöd för deklarativ databindning. Problemet med GridView kontrollens egen sidbrytning och sorteringsfunktion är att den kräver att all information som visas i kontrollen laddas på en gång, och den information som inte visas på sidan sparas i sidans ViewState. Detta medför att resultatet från SQL frågan måste hämta alla rader på en gång som GridView komponenten skall visa, och att onödig data sparas i sidan som skickas till användaren. Om det är fråga om tusentals eller tiotusentals rader vilket korttabellen kan innehålla kan det leda till att SQL servern belastas i onödan och att stora mängder onödig information skickas till användaren. För att lösa detta problem utvecklade man en anpassad sidbrytnings- och sorteringskontroll. SQL frågan för att hämta kortbeställningarna ändrades så att den tog emot parametrar för antalet kort som skall visas och från vilket radnummer den skulle börja. Den anpassade kontrollen förminskade söktiderna och datamängden i HTML sidan drastiskt. Dock genomförs det flera databasförfrågningar eftersom en ny databasförfrågning sker vid varje sidbyte, men eftersom kortbeställningsmodulen har en sökfunktion har de flesta användarna sällan behov av att gå igenom alla sidor.

HTML koden som genererades av modulerna visade sig vara dåligt optimerad efter testerna. Eftersom det var meningen att utseendet på att alla element i modulen skulle be-

stämmas i CSS filer för att lättare kunna skräddarsy produktens utseende till kundens brand hade alla element egna CSS klasser. Genom att förbättra på stilmallen och användarkomponenternas struktur lyckades man minska på HTML sidornas storlek.

8 SLUTSATSER

Detta examensarbete visar hur projektet Henkilökortti .net 2.0 utvecklades från planeringsskedet till slutdokumentationen. Fastän projektet ännu var i testningsskedet då detta examensarbete skrevs uppfyllde projektet till största del de krav som satts för det. Kravspecifikationen ändrades till en del under projektets lopp, men målet att endast använda gratisversioner av utvecklingsverktyg samt de funktionella kraven uppfylldes. Henkilökortti .net 2.0 har ännu utrymme för förbättringar, men ett av målen med projektet var att det lätt skulle kunna vidareutvecklas och förbättras i framtiden. Ett av kravspecifikationens krav som inte uppfylldes var att det skulle vara lätt att kunna lägga till rapportbotten.

Rapportmodulen är olik de andra modulerna som gjordes för projektet. Den är den enda modulen med all funktionalitet hårdkodad. I framtida versioner skulle det vara bra om det skulle vara möjligt att skapa egna rapportbotten, möjligtvis i XML-format för att lätt kunna konvertera det till HTML via en XSL-transformation. En del funktionalitet som realiserades till projektet, t.ex. beskärning av bilder, inaktiverades vid utvecklingsskedet då kravspecifikationen ändrades. Källkoden finns kvar i projektet och kan vid behov tas i bruk efter grundlig testning.

Gratisversionerna av de utvecklingsverktyg som användes i projektet visade sig mer än väl duga för projektet. Dock medförde begränsningarna att alternativa sätt för bl.a. avlusning måste användas. Det man sparade på kostnaderna för de kommersiella utvecklingsverktygena förlorade man i att utvecklingen tog längre tid. Vid större projekt med flera utvecklare skulle dock begränsningarna och saknaden av möjlighet att integrera versionshanteringsprogram kunna visa sig vara för stora hinder för att använda gratisversionerna.

Innehållshanteringssystemet DotNetNuke visade det sig vara en relativt stabil plattform att bygga webbapplikationer på. Då projektet startade hade version 5 av DNN just kommit ut. Den visade sig ha en del buggar, men de flesta fixades snabbt i kommande

uppdateringar. DotNetNukes färdiga moduler och möjlighet att utveckla egna gör det till en bra plattform för webbsidor eller mindre webbapplikationer.

KÄLLOR

Chaffer, Jonathan and Swedberg, Karl 2009. Learning jQuery 1.3. Packt Publishing. ISBN 978-1-847196-70-5

Christianson, Curt and Cochran, Jeff 2009. ASP.NET 3.5 Content Management System Development. Packt Publishing Ltd. ISBN 978-1-847193-61-2

Cross, Michael 2007. Developer's Guide to Web Application Security. O'Reilly Media, Inc. ISBN-13: 978-1-59749-061-0

Hammond, Christopher and Renner, Patrick 2009. DotNetNuke®5 User's Guide: Get Your Website Up and Running. Wiley Publishing, Inc. ISBN: 978-0-470-46257-7

Heilmann, Christian 2006. Beginning JavaScript with DOM Scripting and Ajax From Novice to Professional. Apress. ISBN-10: 1-59059-680-3

McClure, Stuart, Scambray, Joel and Kurtz, George 2009. Hacking Exposed: Network Security Secrets and Solutions, Sixth Edition. McGraw-Hill Osborne Media. ISBN: 978-0-07-161375-0

Nielsen, Paul, White, Mike and Parui, Uttam 2009. Microsoft® SQL Server® 2008 Bible. Wiley Publishing, Inc. ISBN: 978-0-470-25704-3

Rankl, Wolfgang, Effing, Wolfgang 2003. Smart Card Handbook. Third Edition. John Wiley & Sons, LTD. ISBN 0-470-85668-8

Sellers, Mitchel 2009. Professional DotNetNuke® Module Programming. Wrox. ISBN: 978-0-470-17116-5

Silberschatz, Abraham, Korth, Henry, and Sudarshan, S. 2006. Database System Concepts, Fifth Edition. McGraw-Hill . ISBN-13: 978-0072958867

Ullman, Chris and Dykes, Lucinda 2007. Wiley Publishing, Inc. Beginning Ajax. ISBN: 978-0-470-10675-4

Ullman, Chris, Kauffman, John, Hart, Chris, Sussman, Dave and Maharry, Daniel 2004 . Beginning ASP.NET 1.1 with Visual C#® .NET 2003. Wiley Publishing, Inc. ISBN: 0-7645-5708-4

Walker, Shaun, Scarbeau, Brian, Hardy, Darrell, Schultes, Stan and Morgan, Ryan 2009. Professional DotNetNuke®5 Open Source Web Application Framework for ASP.NET. Wiley Publishing, Inc. ISBN: 978-0-470-43870-1

Washington, Michael and Lackey, Ian 2010. Building Websites with DotNetNuke 5. Packt Publishing. ISBN 978-1-847199-92-8

Wellman, Dan 2009. jQuery UI 1.7 The User Interface Library for jQuery. Packt Publishing. ISBN 978-1-847199-72-0

BILAGA

```
declare @rc int

declare @TraceID int

declare @maxfilesize bigint

declare @traceoptions int

declare @endtime datetime

set @traceoptions = 2

set @maxfilesize = 999

set @endtime = '2010-09-15 20:48:55.000'

exec @rc = sp_trace_create @TraceID output, 0, N'C:\Trace\trace14', @maxfilesize, @endtime

if (@rc != 0) goto error

declare @on bit

set @on = 1

exec sp_trace_setevent @TraceID, 10, 14, @on

exec sp_trace_setevent @TraceID, 10, 15, @on

exec sp_trace_setevent @TraceID, 10, 16, @on

exec sp_trace_setevent @TraceID, 10, 1, @on

exec sp_trace_setevent @TraceID, 10, 18, @on

exec sp_trace_setfilter @TraceID, 10, 0, 7, N'SQL Profiler'

exec sp_trace_setstatus @TraceID, 1

select TraceID=@TraceID

goto finish

error:

select ErrorCode=@rc

finish:

go
```