Dnr:922/22-07

Institutionen för teknik

Examensarbete 15 högskolepoäng

Affärssystem för Gamersneed Sweden

Mattias Bertilsson & Johan Lindberg

Oktober 2007

Institutionen för Teknik
Högskolan Kristianstad
291 88 Kristianstad

School of Engineering
Kristianstad University
SE-291 88 Kristianstad
Sweden

Författare, program och år/Author, Program and Year:
Mattias Bertilsson, Datasystemutvecklingsprogrammet 2004
Johan Lindberg, Datasystemutvecklingsprogrammet 2004

Handledare/Instructor:
Stefan Olsson, Gamersneed Sweden

Daniel Hedman, HKr

Examen/Examination:
Detta examensarbete ingår i examenskraven för *Kandidatexamen i Datalogi.*

This graduation work is a part of the requirements for a *Bachelor of Science with a major in Computer Science* (as specified in the English translation).

Svensk titel:

Affärssystem för Gamersneed Sweden

English Title:

Business system for Gamersneed Sweden

Abstract:
This degree project deals with the creation of a business system for Gamersneed Sweden. The system is divided into two parts, one windows appliction and one websolution. (Swedish)

Språk/Language:
Svenska

Godkänd av/Approved by:

_____

Daniel Einarsson Datum/Date

Examinator/Examiner

Sammanfattning

Gamersneed Sweden är ett mindre företag som bedriver handel med datorhårdvara. Företaget inriktar sig mest på försäljning till privatpersoner.

För tillfället så hanteras allt inom företaget manuellt, så vi har blivit ombedda att skapa ett affärssystem som automatiserar och förenklar hanteringen.

Förutom affärssystemet fick vi även i uppdrag att göra en webshop som agerar mot det nya affärssystemet.

Innehållsförteckning sida

1. Inledning

1.1 Bakgrund

När det var dags att leta examensarbete så var det betydligt svårare att hitta ett som kändes relevant för själva utbildningen. Men efter ett antal olika förfrågningar fick vi ett erbjudande från Gamersneed Sweden som var både genomförbart och relevant.

1.2 Målsättning och Syfte

I detta skede så sköter GamersNeed all bokföring och orderhantering via papper. Vilket både är tidskrävande och blir lätt fel.

Alla ordrar läggs via telefon, vilket kräver mycket tid för företaget.

Syftet är att minska pappershantering och fel genom att datorisera merparten av arbetet med ett affärsystem. Samt bygga en E-handelslösning som både ökar företagets ”räckvidd” och underlättar orderläggningen.

Våran målsättning blev att utveckla ett lätthanterligt affärssystem som hanterar Kundregister, Fakturor, Inköp, Bokföring och E-handeln.

För våran egen del så är vårt mål att lära oss nya saker genom att göra ett projekt som verkligen kommer att användas. Lära oss att arbeta på ett uppdrag, kommunicera med kunden för att få reda på hur han vill det ska fungera. Det är stor skillnad mot att göra något åt sig själv då man exakt vet vad man vill ha.

1.3 Metodik och resurser

För att förstå hur vi skulle forma applikationen så fick vi utreda hur nuvarande hantering sköttes. Vilka krav som måste uppfyllas och hur användaren vill det ska se ut och fungera. Vilka resurser som finns att använda och hur vi bäst använder dessa.

För att nå detta så kontrollerade vi befintliga lösningar och skissade upp ett antal möjligheter som vi diskuterade med företaget.

När vi fått klart för oss i stora drag så började vi planera hur allt som skulle ingå skulle interagera med varandra. Vilka programspråk vi bör använda och hur vi ska dela upp arbetet mellan oss.

1.4 Företagspresentation

Gamersneed Sweden är ett litet företag som startades i början av 2006 och har bedrivit handel sedan dess.
Företaget drivs som en enskild firma och har för tillfället inga anställda.

Företagets inriktning är att sälja prestanda datorhårdvara till Datorspelare som kräver mer prestanda.

2. Utredning

2.1 Systemets uppgift

För att lättare förstå vad systemet skulle utföra tog vi reda på exakt vilka delar som skall ingå i systemet och utredde dessa separat.

Dom olika delarna systemet ska motsvara är

- Kundregister
- Varuregister
- Orderhantering(Fakturor)

Förutom detta så skulle även en E-handelsbutik skapas

2.1.1 Kundregister

Kundregistret bör innehålla en unik identifikation för varje kund.
Namn och minst en möjlighet att kunna få kontakt.(E-post, telefon nummer eller adress)

Det nya skall vara lättöverskådligt, vara lätt att hitta vilken kund som köpt vad och få fram alla ordrar som en specifik kund har lagt.
Sök funktion och editering är med ett måste.

2.1.2 Orderhantering

Orderhanteringen består mest av fakturor och orderinnehåll.

Befintlig orderhantering består av fakturor i pappersform och är arkiverade efter datum.

En order skall innehålla Namn på köparen, pris på ordern, samtliga varor som ingår.
Skall gå att editera, radera, skriva ut samt lägga ordrar manuellt.

2.1.3 Varuregister

Varuregistret ska innehålla namnet på varan och all relevant information.
Det kommer att användas både av affärssystemet och e-handelssystemet

Möjlighet att lägga till/ta bort måste finnas. Vid mån av tid så skall ett lagersystem vara knutet till varuregistret.

2.1.4 E-Handelssystem

E-handelssystemet består av två delar, En webbsida och en administreringsdel.
Administreringsdelen är integrerad tillsammans med affärssystemet.

Kunden skall lätt kunna navigera på sidan och söka efter önskad vara.
En kundvagn skall finnas där valda varor hamnar. Kunden skall själv kunna lägga en order och få en orderbekräftelse via E-post.

Efteråt skall kunden kunna se status på hans order.

När en order är lagd på E-handelssidan så ska automatiskt en kund skapas i affärssystemet samt en order läggas.

Applikationen bör upplysa att en obehandlad order finns.

Företaget skall själva lätt kunna administrera E-handeln.
Delar som måste kunna administreras är Kategorilista, varor som syns på hemsidan, Priser på varor, Erbjudande och nyheter.

2.2 Programmeringsmässiga krav

För att komma fram till vilka programmeringsspråk vi skulle använda så behövde vi ta reda på vilka miljöer dom olika systemen använder och vilka resurser som finns.

2.2.1 Affärssystemet

Affärssystemet kommer att köras på en dator som opererar i Windows miljö. Enda kravet är att det ska fungera utan några större ändringar i systemet.

Detta medför att vi kommer att kunna använda i princip vilket programspråk vi själva vill.

2.2.2 E-handelsbutiken

E-Handelsbutiken kommer att ligga på en webbserver som använder Linux som operativsystem. Andra resurser som finns tillgängliga är en My-SQL Server.

Webbservern stödjer PHP, Java. Dock inte asp.

2.3 Säkerhet och Pålitlighet

Då Applikationen och E-handelsbutiken kommer att användas av företaget i stor utsträckning så är stark säkerhet ett måste.

Ur säkerhetssynpunkt finns det två stora ”problem områden”. Det ena är ifall någon utomstående vill sabotera eller manipulera data.

Den andra är ifall Användaren hanterar resurserna på ett så felaktigt sätt att data blir korrupt eller t.o.m. raderad.

För att kunna av att hantera dessa områden måste resurserna vara slutna i sådan mån att man bara kommer åt dom man har befogenhet för. Och att man förutspår så många eventuella felaktiga sätt att hantera Systemet som möjligt och gör dessa omöjliga.

Resurser som för tillfället kan tänkas utsättas för angrepp är Webbservern och Databasservern. Båda två måste vara tillgängliga utifrån för att systemet ska kunna fungera som tänkt, Detta gör det möjligt för även obehöriga att kunna försöka få tillgång.

Själva pålitligheten i systemet måste vara hög. Med detta menas att Systemet skall kunna vara användbart i princip hela tiden. Oavsett omständigheter.

Omständigheter vi tagit med i beräkningen är ifall Anslutningen mellan kontorsdatorn och webbservern / databasservern försvinner. Även ifall detta händer måste man kunna lägga till ordrar manuellt. T.ex. ifall det kommer in en kund i butiken.

2.4 Krav Specifikation

Vi fick en del krav om vad som ska uppnås och vad som behöver vara möjligt i programmet. Det är efter dessa krav vi formade Systemet. Kraven vi fick var.

För Affärssystemet

- Varuregister
- Kundregister
- Orderregister
- Möjlighet att lägga ordrar manuellt
- Ordrar lagda från E-butiken ska automatiskt läggas in och en notifikation ska fås.
- Funktioner för att lägga ut varor och kategorier på E-butiken.
- Kunna skriva ut fakturor från en order.
- I mån av tid Lägga till ett inköpsregister

För E-Butiken

Via Webbsidan ska det vara möjligt att utföra nedanstående punkter.

- Skapa en kund samt kunna logga in som en kund
- Ha en kundvagn där man kan lägga obegränsat med kunder.
- Lista alla erbjudande samt paket som finns ute till försäljning på E-Butiken.
- Lägga en order.
- Få orderinformation från en lagd order.
- Möjlighet att spara undan en kundvagn för att ladda vid senare tillfälle.
- Sidan bör fungera i flera webbläsare. Speciellt Firefox, Opera och Internet Explorer 7.

Generella krav är att både webbsidan och applikationen skall vara lättförståelig, Lätt att navigera sig fram och vara säkert både mot andra och användaren.

Klient applikationen ska fungera under Windows 2000 utan några större ingrepp. Och webbsidan skall fungera på en Linux server med Apache(webbserver).

2.5 PHP

PHP (rekursiv akronym för "PHP: Hypertext Preprocessor") är ett vida spritt Open Source-scriptspråk med brett användningsområde. Det lämpar sig speciellt för webbutveckling och kan bäddas in i HTML.

Det som utmärker PHP från något som t.ex. JavaScript är att koden exekveras på servern istället för hos användaren. Om du skulle använda ett script liknande det ovan på din server, skulle klienten få resultatet av exekveringen men utan någon möjlighet att ta reda på vilken kod som ligger bakom den. Du kan till och med konfigurera din webbserver att tolka alla dina HTML-filer genom PHP-motorn, och då finns verkligen inget sätt för användaren att se vad som finns bakom.

Det bästa med PHP är att det är oerhört enkelt att lära sig, men ändå erbjuder avancerade funktioner för professionella programmerare. Bli inte avskräckt av den långa listan med funktioner, från det att man börjar lära sig PHP tar det inte längre än några timmar innan man kan skriva enkla script.

Även om PHP's utveckling fokuseras på serverside-scripting kan du göra mycket mer med PHP[1]

2.6 Apache HTTP Server Project

Apache är en fri webbserver utvecklad av The Apache Software Foundation. Den senaste versionen är 2.2.4 och finns tillgänglig till, bland andra, FreeBSD, GNU/Linux och Windows.

Apache introducerades 1995 och baserades på populära NCSA httpd 1.3 och är nu den mest använda webbservern i världen. Den används i närmare 50% av alla domännamns webbserversystem.

En av anledningarna till dess popularitet är att Apache är gratis att använda. Dessutom är källkoden öppen, vilket gör det möjligt att modifiera koden.

Apache är en tråddriven webbserver (eller processdriven, i Unix-liknande system används en processdriven variant), vilket betyder att Apache handhar en uppsättning mjukvarutrådar som är färdiga att hantera inkommande förfrågningar. När en förfrågning om en fil anländer till webbservern hanteras förfrågningen av en av de lediga trådarna. Apache har en begränsning av antalet trådar som samtidigt får köras i systemet. Om det maximala antalet aktiva trådar är uppnått, avvisas ytterligare förfrågningar från systemet. [2]

2.7 MySQL

MySQL [ma□eskju□el] är en databashanterare. Den använder sig av frågespråket SQL. MySQL är fri programvara, licensierad under GNU General Public License.

Programmet är skrivet av och underhålls av det svenska företaget MySQL AB i Uppsala. De säljer support och servicekontrakt såväl som kommersiella licensierade kopior av MySQL, och de har anställd personal över hela världen som kommunicerar över Internet. Programmets huvudsaklige utvecklare är finlandssvenske Michael Widenius och David Axmark.

MySQL är en av de mest populära databashanterarna inom Linux-världen, men finns även för ett flertal andra operativsystem, så som FreeBSD, HP-UX, Mac OS X, Netware, Solaris och Windows. [3]

2.8 .NET

Dotnet Framework är en standardiserad plattform för att köra Dotnetprogram. .NET är ur utvecklingssynpunkt språkoberoende. Allt som krävs är att programmeraren har en kompilator som kan översätta koden han/hon skriver till MSIL, Microsoft Intermediate Language. MSIL är bytekod (precis som "kompilerad" Java-kod) och JIT-kompileras när programmet körs av Dotnet Framework. Detta förutsätter alltså att användaren av Dotnetprogram har detta framework installerat på sin dator.

Common Language Infrastructure, CLI, är den standard (ECMA-335 och ISO/IEC 23271) som .NET Framework resulterat i. CLI beskriver bland annat hur program ska kunna köras i flera miljöer utan att behöva skrivas om. Standarden innehåller information om:

- Filformat
- Ett gemensamt typsystem (CTS)
- Ett utbyggbart metadatasystem
- Ett intermediärt språk (MSIL)
- Ett klassbibliotek

Standarden är framtagen gemensamt av Fujitsu, Hewlett-Packard, Intel, ISE, Microsoft samt Monash University.

Kärnan i Dotnet Framework är Common Language Runtime, CLR, som används för att köra programmen. CLR hanterar bland annat kodsäkerhet, objekts livscykelhantering samt avlusning och profilering. [4]

Common Type System, CTS, är den standardiserade del i Dotnet Framework som tillhandahåller olika typer som kan användas av alla språk (se nedan) i Dotnet för att göra program och komponenter språkoberoende.

CLS är den standard som alla Dotnetspråk skall uppfylla för att kunna köras i Dotnet Framework. Följande språk stöds av Microsoft för Dotnet:

- C#
- C++ (i viss mån uppgraderat och modifierat)
- Visual Basic.net
- J#
- Borland Developer Studio (Delphi 2005 och 2006)

2.9 C#

C# (*C-sharp*) är ett objektorienterat programspråk utvecklat av Microsoft som del av .NET-plattformen. C# är ett objektorienterat språk baserat till stor del på de populära språken C och C++, programspråket Java har även en hel del gemensamt med C#.

För utvecklingen av C# ansvarar Anders Hejlsberg.

Programkod skriven i C# omvandlas av en kompilator till så kallad MSIL-kod (Microsoft Intermediate Language), vilket är ett sorts objektkod vilken sedan körs i en virtuell maskin CLR (Common Language Runtime).

Fördelar

- Relativt lätt att lära sig – C#:s syntax liknar syntaxen i språk som C, C++ och Java. .NET innehåller ett stort kodbibliotek vilket förenklar vid utformning av komplexa system.
- Flera användningsområden – C# kan både användas som kompilerat språk på en lokal dator och som språk i ASP.NET. Detta gör det enkelt att länka samman program på en klientdator med serverdatorers program.
- Erbjuder enkel integration med andra Microsoft-baserade programvaror (OBS! Se även avsnittet Nackdelar för mer information om detta).
- C#-kompilatorn kan användas utan licenser och speciella utvecklingsverktyg.

Nackdelar

- Långsam programkörning – Tillämpningar skrivna i C# körs liksom javaprogram i en virtuell maskin vilket bl a innebär att programmet kompileras just innan det körs (se JIT-kompilering) vilket leder till en fördröjd uppstartsfas. Detta kan dock avhjälpas med verktyget ngen.exe som ingår i programsviten vilket förkompilerar MSIL-koden till maskinkod för aktuell processorarkitektur.
- Plattformsbegränsat – C# är starkt bundet till Microsoft Windows. [5]

3. Genomförande

3.1 Val av lösning

3.1.1 Affärssystemet

Efter att begrundat alla dom olika delarna som skall ingå och vilka valmöjligheter vi har så bestämde vi oss för att skriva hela Affärssystemet i C#.
Fördelarna med C# är att det går snabbt att skriva visuella applikationer och vi känner att det är det programspråk vi behärskar mest.

Som databas använder vi den befintliga My-SQL servern. Fördelarna med den är att den står på en väldigt snabb anslutning. Och backup tags dagligen.

Som utvecklingsmiljö använde vi Microsoft Visual Studio.

3.1.2 E-Butiken

Vi hade att välja på JSP Och PHP. Då vi inte hade nån större erfarenhet av PHP och det är något som kan vara bra o kunna valde vi det.

Hela hemsidan och all PHP kod är skriven i en vanlig text-editor(notepad++)

3.2 Databas Design

Själva kärnan i hela systemet är databasen. Det är här allting knyts samman och alla delar agerar med varandra. I Databasen finns all data lagrad för alla systemets delar. Ordrar, Kunder, Trädlistan på E-Butiken osv.

Vi lade ner mycket tid på Design av databasen, Den bestämmer i stora drag hur systemet kommer fungera, prestandan och tillförlitligheten. (Databasmodell bifogad Appendix A)

3.3 Grundläggande Klasser(Affärssystemet)

Klassdiagram bifogad(Appendix B). Som det syns så är Klassdiagrammet väldigt likt Databas modellen, Detta är för att applikationen är väldigt hårt knuten till databasen och fungerar ungefär som ett skal. Ett eget grafiskt gränssnitt mot databasen.

3.3.1 GNSQLConnection

Detta är själva anslutningen mellan applikationen och databasen. All trafik imellan sköts av denna klass. Här finns metoder för att Lägga till, Modifiera och Radera från alla tabeller man som användare har tillgång till. Är den största av alla klasserna, men för att lätt kunna felsöka och ha bra överblick så är den uppdelad i många mindre ”Partial Class” bitar. Detta är en funktion i .net Framework 2.0+ som gör att man kan dela upp en klass över många filer.

3.3.2 GNSQLAgent

Agenten används för att kontrollera händelser i databasen och rapporterar dessa till applikationen. Detta är ett måste ifall flera skall kunna jobba mot databasen samtidigt och för att indikera när något har hänt. Agenten jämför den aktuella listan applikationen jobbar med mot en

ny från databasen med en jämn intervall. Har det hänt något så antingen rapporteras bara skillnaden, eller så rapporteras skillnaden och listan uppdateras automatiskt beroende på hur man ställt in agenten.

3.3.3 Win. Klasserna

Alla Win. Klasserna är ett grafiskt gränssnitt. Det är vad själva användaren kommer att se när han kör programmet. Alla Win. Klasser är skapade efter ett visst mönster(Se mer under Gränssnitt).

3.3.4 DB Objects Klasserna

Dessa klasser representerar en rad i tabell från databasen. Ex Customer motsvarar en rad från tabellen customers. Klassen innehåller all data om kunden och har funktioner för att modifiera denna.

3.3.5 Collections Klasserna

Ett objekt från en av Collection klasserna motsvarar en hel tabell i databasen. Ex. GNCustomers innehåller Customer Objekt. Metoder finns för att lägga till, Modifiera och ta bort tillhörande objekt.

3.4 Användargränssnitt för Affärssystem

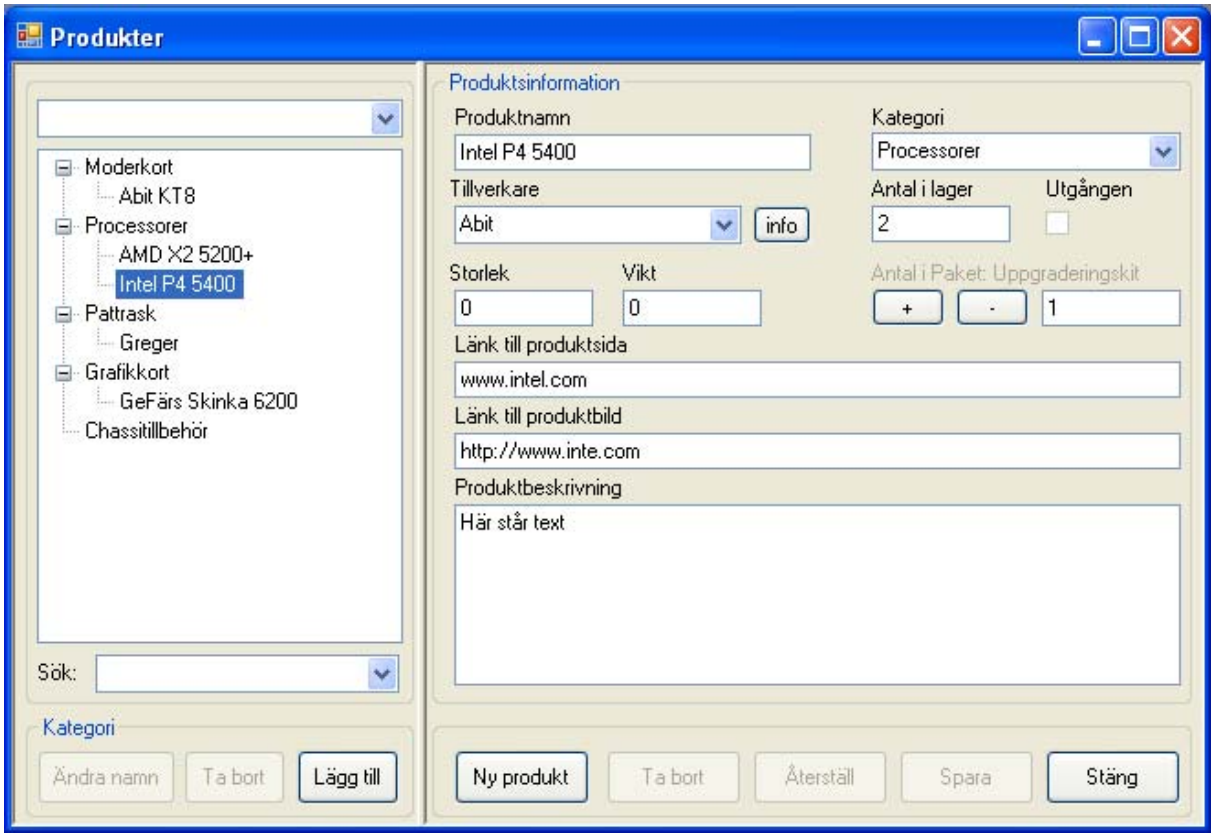
Då affärssystemset är en applikation i Microsoft Windows så är ett fungerande användargränssnitt en naturlig del i utvecklingen. Med tanke på den stora mängden poster som måste översees och hanteras så valde vi en design med ett delfönster för varje huvudregister där poster eventuellt kan förflyttas via s.k. *Drag & Drop*. Åtanken är att så många funktioner som möjligt skall vara uppenbara och lättanvända, samt att gränssnittet fungerar på ett sätt som man kan förvänta sig.

Som ett steg i att göra applikationen lättanvändlig så delar alla huvudfönster samma grundläggande utseende så att allting fungerar på ett logiskt och förutsägbart sätt. Som huvuddrag så är varje fönster uppdelat i två paneler med möjligheten att ändra storleksfördelningen mellan dessa igenom att dra mittenlinjen fram eller tillbaka.

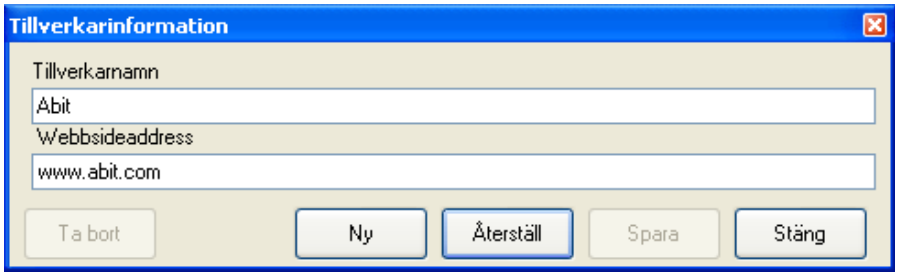
Navigation sker på vänster sida med en lista i trädstruktur (*TreeView)* så att poster kan kategoriseras på ett enkelt och smidigt sätt. Som del av navigationssidan så kan man även begränsa sitt urval med sökfältet, samt lägga till, ta bort eller ändra på kategorier.

Själva posterna visas på höger sida, i samband med knappar för att hantera dessa. Enligt konceptet att så många funktioner som möjligt skall ha fler än ett sätt att utföras så finns det även en knapp för att stänga fönstret förutom kortkommandon och X:et i övre högra hörnet. Knappar så som Spara aktiveras eller avaktiveras i samband med om de för tillfället kan användas – en post som inte är helt komplett (saknar namn, till exempel) kan ej sparas och därav ges ej möjligheten att göra så.

På skärmdumpen nedan ses även hur olika fönster kan samarbeta med hur ett valt paket på paketfönstret visas på produktfönstret och hur på så vis produkter kan läggas till i paket igenom samspel mellan båda fönstren. Som sagt så är det även påtänkt att framtida utveckling tillåter att produkter kan dras direkt från produktnavigationen till ett paket för att ytterligare förenkla användande av affärssystemet.



För mindre poster så som kategorier där det är onödigt med en hel postsida så används mindre fönster som popups (dvs. de måste stängas innan man kan fortsätta) för att ändra information. Liksom huvudfönstren så kan man även här lägga till och ta bort poster efter behov. Notera att man kan även skriva in ett namn på en oexisterande tillverkare eller kategori (till exempel) och applikationen kommer fråga om en ny post med inskrivet namn skall läggas till.



3.5 Användargränssnitt för E-Butik

Med tanke på att E-Butiken skall kunna användas av vem som helst oberoende av datorvana så krävs det en simpel och lättanvänd Layout.

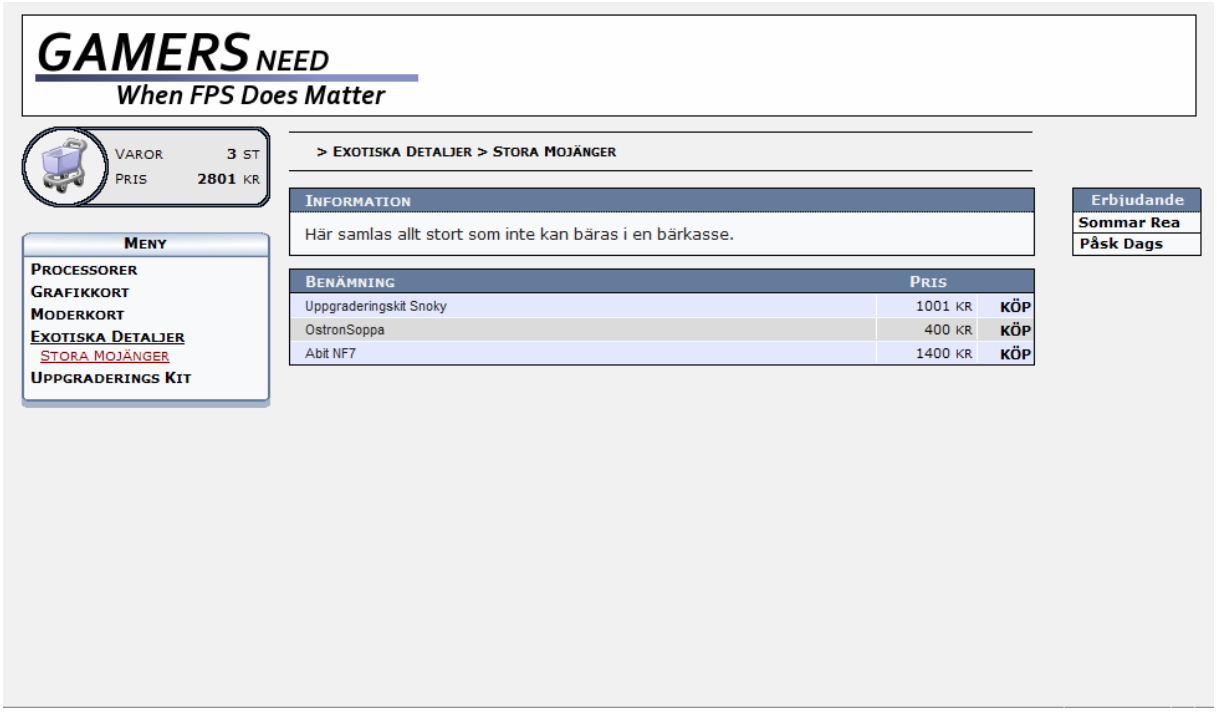
Med detta i tanke så gjordes det en ganska enkel utformning.(Se Bild)
Sidan är i stort sett indelad i Tre olika boxar.

En till vänster som innehåller kundvagn information och en trädstrukturerad meny för att lätt kunna navigera runt mellan kategorierna.

En stor box i mitten som innehåller all relevant data beroende vad man valt.
Här listas t.ex. kundvagns innehåll och varulistor.

På högersidan finns det en Nyhets lista, samt en erbjudande lista.
Allt detta genereras automatisk utifrån vad man ställt in i Affärssystemet.

Exempelvis, ändras en produkt i Affärssystemet så ändras den direkt i E-butiken med.



Bilden visar hur kategorin Stora Mojänger listas. Vill man lägga en vara i kundvagnen så klickar man bara på köp och varan läggs i kundvagnen.

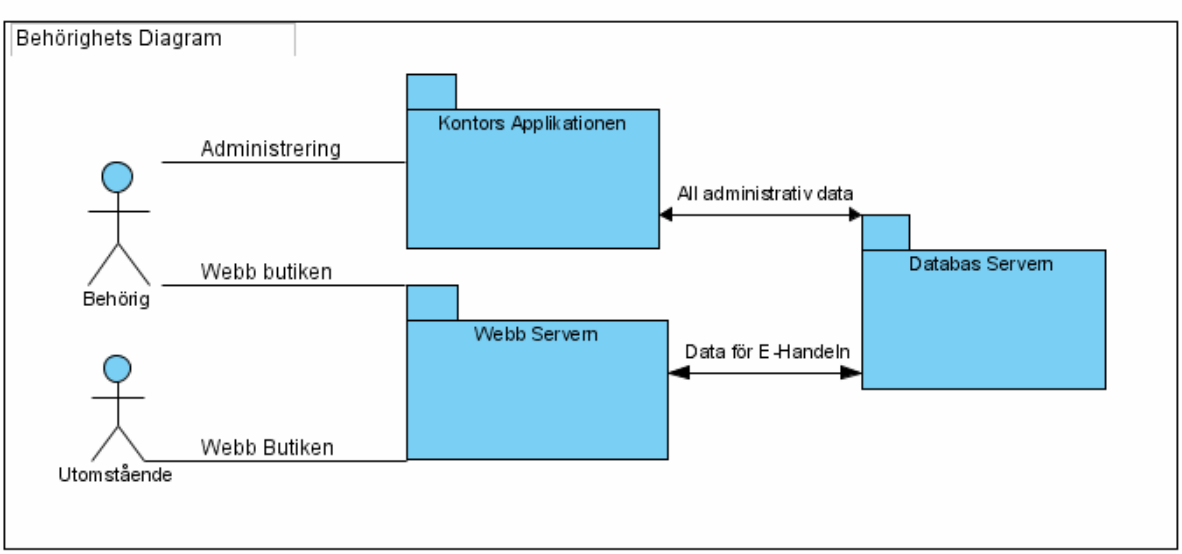
Vill man ha mer information om varan klickar man bara på Benämningen så får man fram mer detaljerad information.

Man ser även att kunden har 3st Varor i kundvagnen för en total summa av 2801kr.

3.6 Säkerhet

Som genomgånget tidigare i Utredningen så är säkerheten ett stort krav.
Inga obehöriga ska komma åt sådant som dom inte har rättighet till.

För att åstadkomma detta har vi avgränsat resurserna så mycket som möjligt.(Se bild)



Bilden visar hur vi begränsat åtkomsten så att dom enda som har utomstående kontakt med Databas servern är Kontors applikationen samt Webbservern.

Detta möjliggjorde vi genom att sätta upp en ”tillåtelse lista” på Databas Servern. Databas servern släpper inte igenom någon nätverkstrafik till någon som inte är på listan.

Nackdelar med detta är att ifall datorn som kontors applikationen körs på byter nätverksadress så måste listan uppdaterad.

Detta är för att ingen ska kunna köra någon sorts attack för att komma in i Databasen.

Ett annat måste var att kontrollera samtliga SQL Satser som Webbservern skickade till Databasservern. Detta är för att förhindra så kallade SQL-Inject attacker.

4. Resultat

Nu i slutskedet så har vi uppnått i princip alla krav som varit satta vid början av uppdraget. Vi har skapat ett komplett System som klarar hantera nästan hela försäljningdelen. Självklart så är aldrig ett System 100% färdigt utan nya funktioner och bättre lösningar är hela tiden ett måste. Ett system blir gammalt i takt med företaget expanderar. Men nuvarande så är det helt funktionellt och har hunnits testköras och buggfixas i en viss utsträckning.

5. Slutsatser

5.1 Användargränssnitt

Ett användargränssnitt är mycket mer än bara komponenterna man använder för att skapa det - användargränssnittet är det språk en applikation använder för att kommunicera med sina användare. Trots erfarenhet i programmering mot fönsterbaserade applikationer så kan vi erkänna att vi var inte riktigt förberedda på hur komplicerat ett gränssnitt som behöver sammarbeta över flera oberoende komponenter kan vara att skapa.

Vi satte tidigt ihop våra tankar om hur vi ville ha ett bra gränssnitt som enkelt kunde visa all information vi ville ha ut och samt sammarbeta innom sig självt för att information lätt skulle samköras på ett logiskt vis. Vikten av att spendera ännu mer tid och planering på detta stadiet visade sig tyvärr mycket senare när mindre detaljer vi hade förbisett och funktioner vi tyckte lät väldigt logiska och funktionella var allt annat än. Resultatet blev att flera funktioner och element i gränssnittet fick genomgå viss omdesign och påtejpade lösningar i och med att kommunikationen och funktionaliteten mellan SQL Databasservern och applikationen utökades.

Mycket av problemet med gränssnittet som vi fann det är att misstag tidigt i planeringen upptäcks inte förrän långt senare när gränssnittet och funktionerna får möjlighet att samspela, och minsta nödvändiga ändring i gränssnittet kan kräva att delar radikalt ändrar utseende.

En andra slutsats angående användargränssnittet som vi upptäckte var svårigheten som ligger i att se gränssnittet objektivt när man designar och utvecklar det. Vi som skaparna av applikationen tyckte naturligtvis att systemet det fungerade på var väldigt logiskt men i flera fall kom på oss frågandes om en speciell funktion var logisk, uppenbar och 'windows' nog - hur mycket instruerande behöver en möjlig ny användare för att använda programmet. Även detta faller på mer planering innan implementation och om man skall vara efterklok så hade det möjligtvis varit en god idé att skriva ut bilder på dom olika fönstren under planering och visa dessa för någon/några som inte är insatta i själva utvecklandeprocessen och fråga om de tycker utseendet och tillhavandet verkar uppenbart och logiskt nog.

Då gränssnittet i slutändan är klart och fungerande så är det uppenbart att mer av planeringsfokus kunde ha lagts på gränssnittet för att undvika flera av de problem vi stötte på med det långt senare.

5.2 E-Butiken

Att bygga ett fungerande E-handelssystem var betydligt större och mer krävande än vi väntat oss. Dom delar vi trodde skulle vara dom svåra visade sig vara lättare än många saker som man tror är grundläggande.

Det som var mest frustrerande var hur olika webbläsare allihopa bedömer html kod helt olika. Såg sidan ut som det var tänkt med en läsare kunde den vara helt förvrängd i en annan, även om man följer befintliga standarders till punk o pricka.

6. Förslag till fortsatt arbete

6.1 Webbapplikation

Webbapplikationens funktioner för nuvarande är rätt grundläggande. Så det finns många möjligheter att bygga vidare. Visuellt kunde man ha flera olika teman som kunderna skulle kunna välja mellan. Funktion att jämföra två varor sida vid sida hade också varit en bra sak.

En viktig sak att få med är fullt stöd för IE6 och äldre, Sidan fungerar visst för dessa men grafiska buggar finns på ett flertal ställe.

6.2 Administrationsverktyg

Det mest uppenbara fortsatta steget för själva C#-applikationen är mer eller mindre optimering. Göra all kontakt med SQL-servern asynkron och så vidare, helt enkelt göra så att gränssnittet och applikationen reagerar snabbare. I samband med databasen är det möjligt att det skulle vara intressant att föra en logg på ändringar i databasen så man har någon koll vad som har ändrats, när och av vem snarare än att endast ha ett datum för senast uppdatering.

7. Alternativa lösningar

7.1 Webbapplikation

Eftersom den existerande webb- och databasservern redan var fastsatt som en PC med Linux körandes MySQL så förfalls det sig naturligt att välja php som bas för webbapplikationen. Alternativet skulle till större delen vara att köra .NET fullt ut och köra ASP.Net istället, men då hade GamersNeed varit tvungna att införskaffa både en Windows Server och en Microsoft SQL Server licens, vilket hade tillfört en signifikant kostnad till det nya systemet utan att precis tjäna på det.

7.2 Administrationsapplikation

Då vi hade relativt fria händer med det administrativa verktyget bortom att det skulle köras på datorer med Microsoft Windows XP så bestämde vi oss för Visual Studio och C# som utvecklingsmiljö och programmeringsspråk. I och med Visual Studio så fick vi en kraftfull miljö att designa användargränssnittet med och C# är ett programmeringsspråk som vi båda behärskar. Hade kraven varit annorlunda så är det möjligt att applikationen hade kunnat bli skriven i Java eller tillochmed i php som en utökad del av webbgränssnittet. Då det inte fanns någon orsak att förvänta sig att administrationsverktyget skulle existera på någon annan platform än Windows i framtiden så bestämde vi oss för att satsa på den säkrare och enklare lösningen istället för att ge oss i kast med en utvecklingsmiljö som hade känts främmande.

7.3 Val av språk

Även om vi tyckte oss föredra engelska under programmering, särskilt med hur programmeringsspråken i sig själva är på engelska, så valde vi svenska som språk utåt mot användare. Tanken är eftersom både GamersNeeds personal och kunder är svenskar så tyckte vi det var säkrare att ta det i akt snarare än att anta att alla kunder fullt behärskar flytande engelska.

8. Referenser

[1] Webbsida http://se2.php.net/

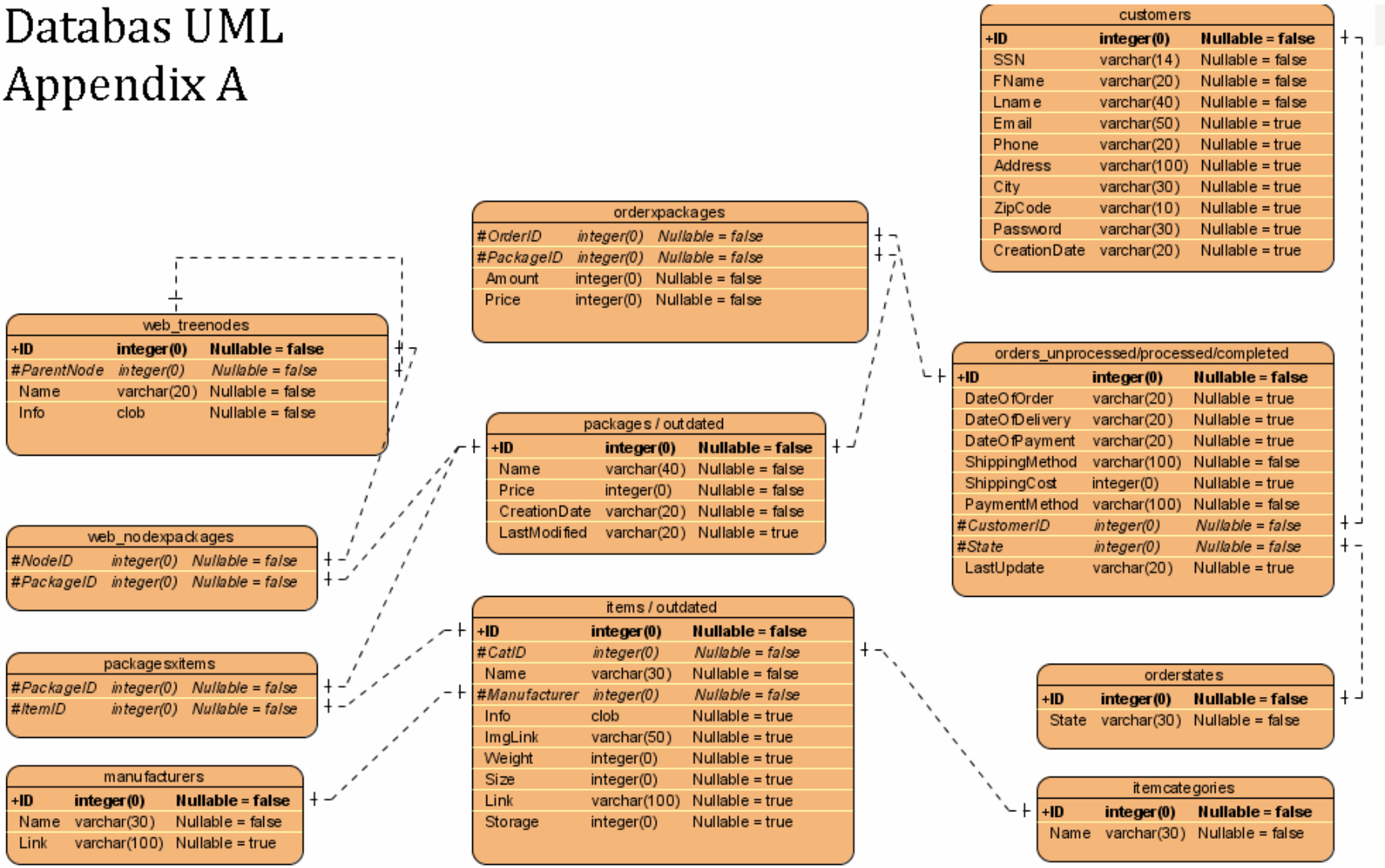
[2] Webbsida http://sv.wikipedia.org/wiki/Apache_HTTP_Server

[3] Webbsida http://sv.wikipedia.org/wiki/MySQL

[4] Webbsida http://sv.wikipedia.org/wiki/Dotnet

[5] Webbsida http://sv.wikipedia.org/wiki/C-sharp

Databas UML
Appendix A



Klass Diagram
Appendix B