Webbaserat inventeringssystem och produktkatalog

Ett produktutvecklingsarbete

Pernilla Haglund

Examensarbete
Informations- och medieteknik
2013

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	4204
Författare:	Pernilla Haglund
Arbetets namn:	Webbaserat inventeringssystem och produktkatalog Ett produktutvecklingsarbete
Handledare (Arcada):	Johnny Biström
Uppdragsgivare:	Oy Tecmarin Ship Supply Ab

Sammandrag:

Jag anställdes av Oy Tecmarin Ship Supply Ab, ett finländskt företag inom skeppshandel. Företaget var i behov av ett elektroniskt system för informationshantering. Systemet skulle användas för att hålla lagersaldot à jour, för att erbjuda företagets anställda enkel och effektiv tillgång till produktinformation, priser, kontaktuppgifter och försäljningsstatistik. Viktigt var att systemet blev säkert, enkelt att använda, flexibelt och kostnadseffektivt. Förutom informationshanteringssystemet behövdes även en skild webbapplikation för Tecmarins kunder, nämligen en produktkatalog där kunderna kunde bekanta sig med sortimentet. Produktkatalogen skulle vara enkel att söka i samt på tre olika språk - finska, svenska och engelska. Jag valde att bygga systemet på en relationsdatabas i MySQL och programmera applikationerna i PHP, HTML, JavaScript, Ajax och CSS. I den här rapporten beskriver jag hur systemet är uppbyggt samt diskuterar slutresultatet.

Nyckelord:	databas, relationsdatabas, MySQL, webbapplikation, CMS, datasystem, informationshantering, PHP, webbprogrammering, datasäkerhet
Sidantal:	65
Språk:	Svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information and media technology
Identification number:	4204
Author:	Pernilla Haglund
Title:	Web content management system and product catalogue A product development job
Supervisor (Arcada):	Johnny Biström
Commissioned by:	Oy Tecmarin Ship Supply Ab

Abstract:

I was employed by Oy Tecmarin Ship Supply Ab, a Finnish company in the international shipping industry. The company was in need of an information system to keep the stock balance up to date and to offer the employees efficient access to product information, prices, contact information and sales statistics. The security of the system was an important issue as well as that it was easy to use, flexible and cost-effective. Beside the information system, consisting of a database and a content management system, a separate web application was wanted to provide Tecmarin's customers a catalogue with all the products and prices. The product catalogue would give the customers opportunity to see the product range and make searches within it, in three different languages – Finnish, Swedish and English. I chose to build the system on a relational database in MySQL and do the applications in PHP, HTML, JavaScript, Ajax and CSS. In this report I describe the system's structure, technical solutions and user interface. Then I evaluate and discuss the results.

Keywords:	database, relational database, MySQL, web application, CMS, data system, information handling, PHP, web programming, data security
Number of pages:	65
Language:	Swedish
Date of acceptance:	

INNEHÅLL

Figurer

1 INLEDNING

1.1 Bakgrund

Jag anställdes av Oy Tecmarin Ship Supply Ab för att utveckla ett elektroniskt system för informationshantering inom bolaget samt en webbapplikation för att förbättra kundservicen. Tecmarin grundades 1986 och är ett av Finlands största företag inom skeppshandel. Koncernen består av moderbolaget Oy Tecmarin Ship Supply Ab och dotterbolaget Oy Tecmarin Trading Ab. Kontoret är beläget i Helsingfors men verksamheten är internationell. (Oy Tecmarin Ship Supply) Den årliga omsättningen ligger kring 1 700 000 euro. (Taloussanomat)

På Tecmarin planerades en grundlig lagerinventering vid tidpunkten då jag anställdes. Det företaget saknade var ett system som skulle möjliggöra att man efter inventeringen, på ett enkelt och effektivt sätt, kunde hålla produktsortimentet och lagersaldot uppdaterat. I samband med detta ville man även förverkliga elektronisk lagring och hantering av övrig information så som priser och kontaktuppgifter.

På Tecmarin satsar man på god kundservice så i och med att all information skulle överföras till elektronisk form önskades också en webbapplikation i form av en produktkatalog. En webbkatalog för översikt av produktsortimentet hade redan tidigare funnits i tankarna men dess förverkligande blev ännu mer betydande då en av bolagets viktiga och trogna kunder efterfrågat dylik service.

Då jag tackade ja till arbetserbjudandet hade jag ganska knappt om erfarenhet av databasteknik och webbprogrammering. Jag såg dock projektet som en ypperlig möjlighet att utvecklas inom branschen och eftersom färdigställandet inte var brådskande kunde jag räkna med att jag skulle hinna inhämta kunskapen jag saknade parallellt med utvecklandet av produkten.

Nu när systemet är klart och har tagits i bruk ser jag det som en självklarhet att skriva mitt examensarbete om hur jag egentligen gick till väga och vad resultatet blev.

1.2 Syfte och mål

I det här arbetet rapporterar jag om hur jag byggt upp datasystemet jag utvecklat för Oy Tecmarin Ship Supply Ab. Jag berättar om hur jag planerat systemet utgående från företagets behov, om vilka tekniska lösningar jag valt och varför, om datasäkerheten, applikationernas användargränssnitt samt analyserar slutresultatet. Jag diskuterar också vad jag lärde mig under projektets gång och hur det märktes i mitt arbete.

Syftet med detta är att dela med mig av mina erfarenheter och låta mina lösningar stå som exempel. Jag har som avsikt att på ett klart och tydligt sätt förklara vad jag gjort och hur, så att andra kan ta modell och lära sig av mitt arbete.

Målet är att först förklara systemets uppbyggnad för att sedan kunna analysera olika lösningar och resultatet i sin helhet. Genom att granska mitt resultat vill jag reda ut vad jag är nöjdast med och vad jag skulle göra annorlunda ifall jag byggde ett motsvarande system på nytt. Jag vill kunna ge mig själv och läsaren goda råd.

Det som varit den viktigaste biten, i synnerhet för mig personligen, är systemets säkerhet. Jag kommer därför att syna datasäkerheten lite extra noga och försöka komma fram till ifall systemets säkerhet kan anses vara tillräcklig.

Jag vill också ta reda på huruvida systemets användare varit nöjda. Fick Tecmarin det de ville ha? Har systemet fungerat problemfritt och var det enkelt att lära sig att använda? Hur har arbetsrutinerna ändrats och hur har applikationerna påverkat effektiviteten, försäljningen och företagets image? Genom att be företagets anställda om feedback skall jag hitta svar på frågorna.

1.3 Metoder

Metoderna jag använt är först och främst planering och utvecklande av en databas och två webbapplikationer, det vill säga praktiskt arbete. Efter det analyseras slutresultatet genom att både själv kritiskt granska det färdiga systemet och genom empirisk undersökning i form av intervjuer per e-post. Jag kommer att intervjua systemets användare samt ta kontakt med en expert inom informationsteknik.

1.4 Avgränsning

Jag utgår i mitt skrivande från att läsaren redan känner till området och tekniken. Jag ser också till att inte bli alltför detaljerad då jag beskriver arbetsprocessen och källkoden. Jag plockar fram exempel i stället för att beskriva precis allt. Avsikten är att skapandet av ett likadant system skulle kunna upprepas av någon annan inom branschen på basis av min rapport och att analysen av slutresultatet belyser mina tekniska lösningars för- och nackdelar.

2 SYSTEMETS RIKTLINJER

Jag fick mycket fria händer för arbetets förverkligande. Huvudsaken var att det färdiga systemet mötte kraven på funktionalitet och effektivitet. Viktigt var självklart också att säkerheten var tillräckligt hög och att produkten inte blev dyrare eller mer komplicerad än nödvändigt. Det systemet behövdes till var främst att hålla lagersaldot uppdaterat, ge tillgång till produkternas prisinformation samt erbjuda företagets kunder en katalog över sortimentet.

I och med att det här var frågan om att lagra stora mängder data som kostnadseffektivt och flexibelt skulle kunna administreras, var det självklart för mig att grunden för systemet skulle vara en relationsdatabas i MySQL. MySQL är den populäraste RDBMS:en (Relational Database Management System) som är gratis att använda (Oracle Corporation) och jag hade arbetat med systemet tidigare. Dessutom var det MySQL som fanns att tillgå hos webbhotellet Netsor (Netsor Oy) som Tecmarins webbsidor redan fanns på. Eftersom databasen skulle ha två skilda användningsområden tyckte jag också att det bästa var att producera två olika applikationer. En applikation för företagets anställda (databasens hanteringsverktyg) och en annan applikation för företagets kunder (produktkatalog).

Databasens hanteringsverktyg, eller CMS (Content Management System), skulle komma att vara i dagligt bruk och därför var det av stor vikt att användningen av det, det vill säga att göra uppdateringar och sökningar i det, gick snabbt och lätt. Man ville också dela upp användarrollerna i två grupper – i administratörer och vanliga användare – för att öka säkerheten. De vanliga användarna skulle få se all data och göra sökningar i det

medan administratörerna även kunde lägga till, ta bort och ändra data. Språket i användargränssnittet skulle vara finska. I CMS:et skulle man ha åtkomst till databasens information om produkter, kunder, leverantörer och priser samt olika uträkningar på lagervärdet.

Produktkatalogen skulle vara en tydlig översikt på varorna med specifik produktinformation så som bild och försäljningspris. Här skulle användargränssnittets språk finnas på både finska, svenska och engelska, men det bestämdes först i ett senare skede.

Lösningen för hur dessa två applikationer skulle skapas såg jag genast i självständig webbprogrammering. Jag begrundade knappt andra alternativ eftersom jag ansåg att det ändå rörde sig om så enkla och få funktioner att det mest ändamålsenliga var att programmera allt själv i stället för att till exempel börja anpassa ett färdigt CMS. Ett skräddarsytt hanteringsverktyg skulle även trygga ett, ur användarnas synvinkel, enkelt och logiskt användargränssnitt som inte innehöll fler funktioner än nödvändigt. Programmeringsspråken jag använde var i huvudsak PHP, SQL, HTML och CSS. För vissa funktioner använde jag även JavaScript och Ajax. Jag använde mig helt enkelt av de kunskaper jag inhämtat under studierna på Arcada och byggde sedan vidare på dem. Detta innebar även att programmeringsparadigmen var procedurell programmering.

Tecmarin hade sedan tidigare en webbplats gjord med Joomla vilket innebar att en domän redan fanns registrerad, MySQL och PHP5 var installerade och utrymme fanns att tillgå. Det var därför bekvämt att kunna använda sig av samma domän och databas genom att placera applikationernas kataloger under samma domän och lägga till nya tabeller i samma databas. Jag satte även upp en egen lokal utvecklingsmiljö på min dator; XAMPP 1.7.3 för Windows. Programmeringen gjorde jag i Notepad++, CSS-filerna i Microsoft Sharepoint Designer och bildediteringen i Adobe Photoshop.

3 DATABASEN

3.1 Planering av databasen

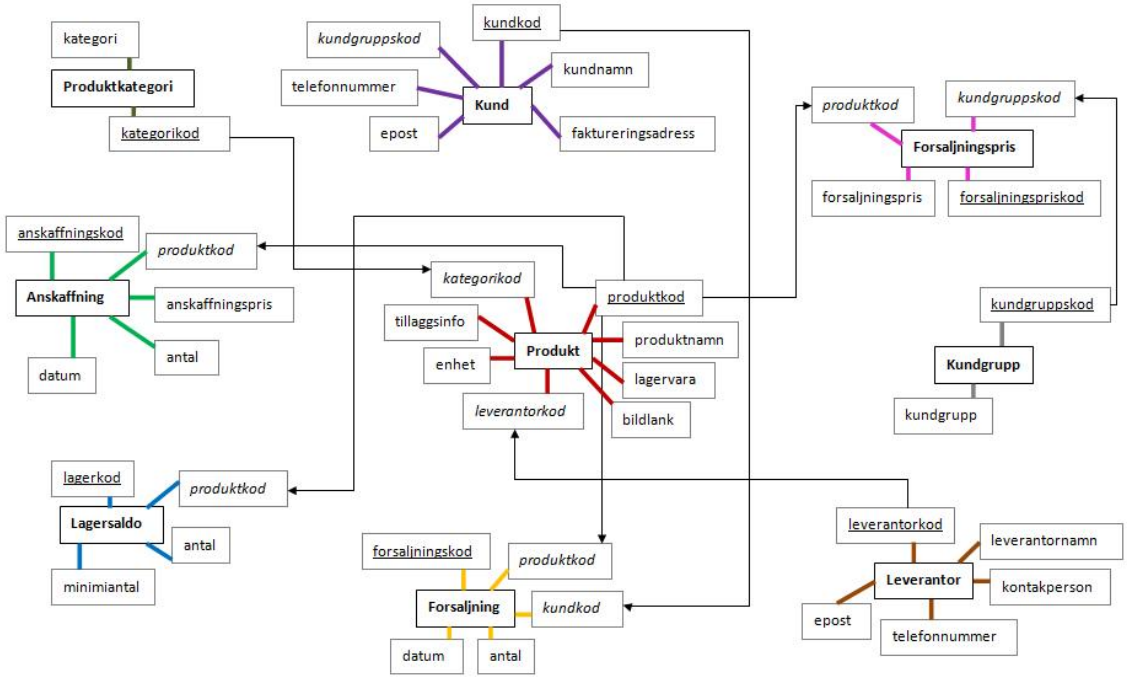
Då jag hade datasystemets användningsändamål och krav klara för mig kunde jag sätta igång med att planera databasens struktur. Jag hade nu några viktiga saker att ta i beaktande.

Lagersaldot för olika produkter skulle som sagt uppdateras dagligen i och med anskaffningar och försäljningar. Inte minst för att CMS:ets administratörer skulle vara fler än en kom jag fram till att varje anskaffning och försäljning bör loggas för att undvika felaktiga siffror. Ett dåligt alternativ skulle ha varit en kolumn med lagersaldot för varje produkt som skulle uppdateras direkt utan att spara information om själva ändringen. Det skulle till exempel kunna leda till att Admin1 räknar om lagersaldot efter en försäljning och gör ändringen i databasen för produkten i fråga. Därefter gör Admin2 samma ändring, ovetande om att den redan gjorts, och lagersaldot blir felaktigt. Genom att i stället konstruera två skilda tabeller för att lagra data om alla anskaffningar och försäljningar kunde detta undvikas. Dessutom skulle det ge företaget andra fördelar så som möjlighet att föra försäljningsstatistik av olika slag fastän det insåg jag först senare.

En annan central sak var försäljningspriserna, de kunde nämligen variera kunder emellan. Tillsammans med min uppdragsgivare Charlotta Åberg på Tecmarin delade vi därför in kunderna i grupper där alla inom varje grupp hade samma försäljningspris för samma produkt. Antalet kundgrupper blev 11.

I nedanstående avsnitt berättar jag hur jag designat databasen. Jag har för tydlighetens skull delat upp tabellerna i vad jag kallar huvudtabeller och övriga tabeller. Huvudtabellerna är de som står i relation till varandra och innehåller all information om produkter, kunder och leverantörer. De övriga tabellerna är de som står för inloggningsdata, bildlänkar för användargränssnitten samt för raderade poster.

3.1.1 Diagram över huvudtabellernas attribut och relationer



Figur 1. Huvutabellernas schema. För större format, se bilaga 1.

Ovanstående diagram är en översikt på tabellerna, attributen (kolumnerna) och nycklarna som skapar relationerna tabellerna emellan. Jag presenterar tabellerna i den ordning de bör skapas i den fysiska designen.

Att få till ovanstående diagram gick inte hur lätt som helst. Det var en hel del funderande och ändrande och funderande igen innan jag slutligen var nöjd. Resultatet blev nio tabeller; *Leverantor*, *Produktkategori*, *Produkt*, *Kundgrupp*, *Kund*, *Forsaljningspris*, *Anskaffning*, *Forsaljning* och *Lagersaldo*. Jag lät bli att använda å, ä, ö, mellanslag och specialtecken i tabell- och kolumnnamnen för att undvika problem. Lägg också märke till att tabellnamnen i designen har stor begynnelsebokstav medan kolumnnamnen har liten så att man skall veta vilketdera som avses i fall namnen annars är identiska.

I den ursprungliga versionen av diagrammet är namnen på finska men jag valde att i efterhand göra en svenskspråkig version för att underlätta presenterandet av den här i dokumentationen. Likaså är den fysiska designen gjord med finska benämningar. För att ytterligare förtydliga min text låter jag tabell- och kolumnnamnen stå i kursiv stil.

Tabellen *Leverantor* innehåller primärnyckeln *leverantorkod* och de andra attributen *leverantornamn* som är namnet på bolaget i fråga samt *kontaktperson*, *telefonnummer* och *epost*. Attributet *kontaktperson* innehåller både för- och efternamn på den person Tecmarin oftast har att göra med då de beställer in nya varor.

Enligt diagrammet består *Produktkategori* endast av två attribut – primärnyckeln *kategorikod* och *kategori* för dess benämning på finska. Eftersom jag inte ännu i planerandets skede visste att produktkatalogen även skulle finnas på svenska och engelska var det först senare som jag lade till de två attributen *kategori_sv* och *kategori_en* för översättningar av kategorinamnen.

Produkt-tabellen är den mest centrala. Dess primärnyckel *produktkod* pekar på fyra olika tabeller, nämligen på *Försäljning*, *Anskaffning*, *Försäljningspris* och *Lagersaldo*. För alla poster (rader) i dessa tabeller måste man, givetvis, veta vilken produkt som avses. *Produkt* består också av två främmande nycklar, *leverantornamn* och *kategorikod* eftersom varje produkt måste ha en leverantör samt höra till en produktkategori. Från början var tabellens övriga attribut *produktnamn*, *lagervara*, *bildlank*, *enhet* och *tillaggsinfo*. Men av samma orsak som framkommer i beskrivningen av *produktkategori*, gjordes en ändring i efterhand då attributen *produktnamn_sv* och *produktnamn_en* lades till.

Eftersom vissa produkter klassas som lagervara och andra ej, behövs ett attribut för att definiera det. I *bildlank* sparas en URL (Uniform Resource Locator) till en bild på produkten om så finns och *enhet* ger information om i vilken enhet produkten säljs (i liter, kilogram, rulle, set osv.). *Tillaggsinfo* är för små anteckningar som kan behöva sparas om produkten av en eller annan orsak.

För klarhetens skull påpekar jag ännu att en post i tabellen är en slags vara med unika egenskaper. Det kan alltså finnas en godtyckligt stor uppsättning av samma vara där varje enhet definieras av samma produktkod. Dock om det finns två varor som annars är helt likadana men som köpts in från olika leverantörer blir posterna två.

Kundgrupp behövdes som sagt för att kunna ha olika försäljningspriser för olika kunder och består av primärnyckeln *kundgruppskod* och attributet *kundgrupp* för benämning.

Tabell *Kund*s attribut är primärnyckeln *kundkod*, främmande nyckel *kundgruppskod* samt *kundnamn*, *faktureringsadress*, *epost* och *telefonnummer*. *Kundgruppskod* vittnar

om tillhörande kundgrupp och *kundnamn* kan vara antingen namnet på ett bolag eller en privatpersons för- och efternamn.

I *Försäljningspris* finner vi primärnyckeln *försäljningspriskod*, främmande nycklarna *produktkod* och *kundgruppskod* samt *försäljningspris*. Detta innebär nu att det finns lika många poster per produkt som det finns kundgrupper. Det finns med andra ord 11 skilda rader med försäljningspris per produktkod.

Anskaffning är en tabell med *anskaffningskod* (primärnyckel), *produktkod* (främmande nyckel), *anskaffningspris*, *antal* och *datum*. I denna tabell är varje post en skild anskaffning av vara. Köper Tecmarin till exempel in 30 stycken bollventiler i mässing med diametern 32mm den 10de januari 2013 så loggas en unik anskaffningskod, ventilens produktkod, anskaffningspriset per enhet, antalet 30 med dateringen 2013-01-10. Detta kräver dock att produkten i fråga redan existerar i tabellen *Produkt*.

Liksom varje inköp av vara loggas i *Anskaffning*, loggas varje försäljning i tabell *Försäljning*. Här har vi primärnycken *försäljningskod*, främmande nycklarna *produktkod* och *kundkod* samt *antal* och *datum.* På så vis hålls reda på hur många enheter av vilken produkt som har sålts till vem och när.

Den sista huvudtabellen är *Lagersaldo*. Den finns till för att ange hur många stycken av en viss produkt som finns på lager för tillfället. Loggningarna av anskaffning och försäljning görs ju av användaren i databasens innehållshanteringsverktyg (som jag berättar mera om i nästa kapitel) och varje gång en loggning sker uppdateras lagersaldot automatiskt i denna tabell. Attributet som uppdateras här heter *antal*. De övriga attributen är den främmande nyckeln *produktkod*, primärnyckeln *lagerkod* och slutligen *minimiantal*. Minimiantal finns ifall man vill ge en vara en nedre gräns på hur många enheter som alltid bör finnas på lager.

3.1.2 Huvudtabellernas datalexikon

För huvudtabellernas datalexikon, se bilaga 2.

I datalexikonet definieras attributens datatyper och begränsningar. Så som också var fallet i diagrammet (bilaga 1), saknar datalexikonet attributen som jag lade till senare för att möjliggöra översättning av produktnamn och kategorinamn till svenska och engels-

ka. Även här har jag översatt tabellerna och attributen till svenska eftersom de ursprungligen var på finska.

I datalexikonets första kolumn har vi tabellnamnen, i andra attributen och i den tredje en kort beskrivning. Från och med den fjärde kolumnen finns olika definitioner och begränsningar för attributets data, så som definition av datatyp och – längd, eventuella förinställda värden osv.

För attribut vars fält kan innehålla olika sorters tecken (namn, e-post, telefonnummer etc.) har naturligtvis datatypen VARCHAR valts. Datalängderna för dessa varierar från 40 för *telefonnummer* till 200 för *tillaggsinfo*.

För de attribut vars fält endast kommer innehålla siffror har jag satt antingen SMALLINT, INT eller BIGINT, beroende på vad jag ansåg lämpligast. Till exempel för primärnycklarna i *Anskaffning* och *Forsaljning* valde jag BIGINT eftersom posterna där kan bli hur många som helst. För *kundgruppskod* satte jag SMALLINT eftersom kundgrupperna hålls på ett mycket lägre antal.

De attribut som kan stå för decimaltal tilldelades datatypen DOUBLE. *antal*-attributen begränsades till att vara högst tio siffror långa medan pris begränsades till nio. Jag medger att det är lite i överkant men hellre lite för långa än för korta tyckte jag. Båda datatyperna kan ha högst tre decimaler. Datumen sparas i formatet ÅÅÅÅ-MM-DD med hjälp av datatypen DATE och om inte annat anges, sätts dagens datum med funktionen DEFAULT CURDATE(). DEFAULT används också för *kundgruppskod* i *Kund* samt för *lagervara* i *Produkt*. Det innebär att en ny kund läggs i kundgruppen ”Övriga” om inte annan grupp tillskrivs samt att det förinställda värdet för lagervara är ”nej”.

Alla främmande nycklar (i datalexikonet förkortat FK) tilldelades egenskaperna ON DELETE CASCADE och ON UPDATE CASCADE eftersom jag på ett enkelt sätt ville förhindra att ofullständig eller föråldrad data skulle hänga kvar.

De attribut, förutom primärnycklarna, som jag ville att skulle vara unika för att undvika dubbel data, gav jag restriktionen UNIQUE och antal varor i *Anskaffning* och *Forsaljning* kontrolleras vara större än noll med hjälp av CHECK(antal>0). Vissa attribut behöver inte nödvändigtvis innehålla något data, dessa är bland andra *bildlank* och *tele-*

fonnummer. De får därför vara NULL medan de viktigare attributen är NOT NULL. Alla primärnycklar är unika sifferkoder och sköts av funktionen AUTO_INCREMENT.

3.1.3 Databasens övriga tabeller

De övriga tabellerna i databasen är tolv. Sju av dem är till för data som kan raderas genom hanteringsverktyget. Med andra ord har alla huvudtabeller förutom *Produktkategori* och *Kundgrupp* en så gott som motsvarande tabell med prefixet Del_. Om man via hanteringsverktyget t.ex. raderar en post i tabellen *Kund,* flyttas posten först över till *Del_kund* innan den tas bort från *Kund*. Inget viktigt data kan m.a.o. raderas via hanteringsverktyget utan det flyttas i stället över till en annan tabell. Den största skillnaden mellan *Tabell* och *Del_tabell* är att primärnyckeln i *Del_tabell* är en egen sifferkod som också fungerar med AUTO_INCREMENT. På så sätt sorteras posterna i den ordning de lagts till, vilket gör posten (eller posterna) som ur CMS:et raderats av misstag lättare att återställa.

Förutom Del-tabellerna finns det fyra tabeller med inloggningsdata och en tabell för bildlänkar för bilder som inte hänger ihop med en viss produkt. Två inloggningstabeller är för hanteringsverktyget och de två andra är motsvarande tabeller för produktkatalogen. En inloggningstabell innehåller uppgifter som namn, användarnamn, krypterat lösenord och salt medan den andra lagrar information om inloggningar (eller inloggningsförsök) som användarnamn, IP-adress och tidpunkt. Tabellen med bildlänkar har tre attribut; *bildkod, URL* och *beskrivning*.

3.1.4 Databasens realisering

Då databasens relationer och restriktioner var klara för mig var följande steg att formulera DDL (Data Definition Language) -satserna i SQL. Jag ger DDL-satsen för *Produkt* som exempel:

```
CREATE TABLE produkt
(produktkod INT NOT NULL AUTO_INCREMENT,
produktnamn VARCHAR(90) NOT NULL,
enhet VARCHAR(50),
kategorikod SMALLINT NOT NULL,
leverantorkod INT NOT NULL,
tillaggsinfo VARCHAR(200),
bildlank VARCHAR(80),
lagervara TINYINT(1) NOT NULL DEFAULT 0,
PRIMARY KEY (produktkod),
UNIQUE (produktnamn),
FOREIGN KEY (kategorikod) REFERENCES produktkategori(kategorikod)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (leverantorkod) REFERENCES leverantor(leverantorkod)
ON UPDATE CASCADE ON DELETE CASCADE) ENGINE=INNODB;
```

Figur 2. $DDL-satsen för tabell Produkt.

Inget konstigt här, attributen skapas och datatyper, nycklar och andra egenskaper definieras. Motsvarande Del-tabell skapades däremot utan att definiera främmande nycklar. Eftersom produktkategorier och kundgrupper inte kan administreras via CMS:et existerar heller inte *Del_produktkategori* och *Del_kundgrupp*. Därmed existerar heller inga relationer mellan Del-tabellerna och DDL-satsen för *Del_produkt* ser ut så här:

```
CREATE TABLE del_produkt
(del_produktkod INT NOT NULL AUTO_INCREMENT,
produktkod INT,
produktnamn VARCHAR(90),
enhet VARCHAR(50),
kategorikod SMALLINT,
leverantorkod INT,
tillaggsinfo VARCHAR(200),
bildlank VARCHAR(80),
lagervara TINYINT(1),
PRIMARY KEY (del_produktkod)) ENGINE=INNODB;
```

Figur 3. DDL-satsen för tabell Del_produkt.

Här ser man också att jag endast försett primärnyckeln med restriktioner, jag tyckte det var bäst så för att vara säker på att överflyttningen av en post lyckas, före den raderas ur huvudtabellen.

Då alla tabellers DDL var skrivna, exekverade jag dem i phpMyAdmin i min egen utvecklingsmiljö. Efter det körde jag in testdata i tabellerna för att kunna kontrollera med satser i DML (Data Manipulation Language) att allt verkligen fungerade som det skulle. Jag testade med lite mer invecklade men relevanta frågesatser, bland annat denna:

```
SELECT produkt.produktnamn, produktkategori.kategori,
produkt.enhet, forsaljningspris.forsaljningspris
FROM forsaljningspris
INNER JOIN produktkategori
ON produkt.kategorikod=produktkategori.kategorikod
INNER JOIN produkt
ON forsaljningspris.produktkod=produkt.produktkod
INNER JOIN kundgrupp
ON forsaljningspris.kundgruppskod=kundgrupp.kundgruppskod
WHERE kundgrupp='Viking';
```

Figur 4. DML-sats för att kontrollera att ihopkopplandet av olika tabeller fungerar.

Eftersom satserna utan krångel gav resultaten jag ville ha, kunde jag nu räkna med att databasen var färdig för det riktiga datat. Jag hade bett att få det i Excel-tabeller, organiserat och uppdelat på ett sätt som efterliknade databasens tabeller. Jag hade också påpekat att vissa benämningar, som produktnamnen, måste vara unika och inte flera tecken än vad jag hade bestämt. Data skulle köras in i alla huvudtabeller förutom i *Anskaffning* och *Forsaljning*, de skulle börja fyllas i manuellt först senare via hanteringsverktyget. Innehållet i tabellen *Lagersaldo* skulle också köras in först då hanteringsverktyget var klart att tas i bruk, annars skulle informationen hinna föråldras.

4 INNEHÅLLSHANTERINGSSYSTEMET

I detta kapitel presenterar jag först CMS:ets användargränssnitt så att man får en uppfattning om programvarans struktur och funktionalitet. Därefter går jag in på applikationens tekniska förverkligande genom att bland annat beskriva katalogstrukturen samt visa och kommentera utdrag ur källkoden.

4.1 Innehållshanteringssystemets användargränssnitt

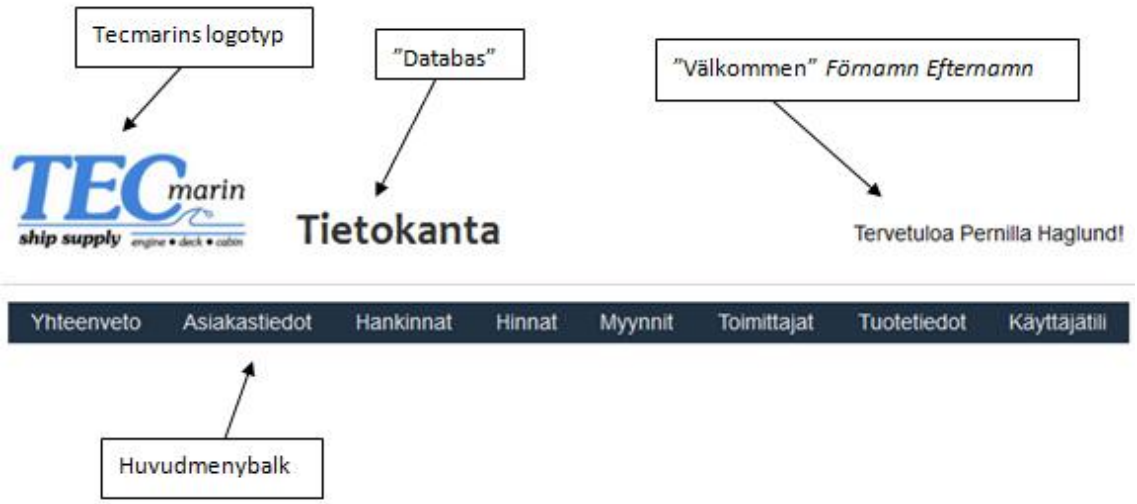
Användargränssnittet skulle vara så enkelt och logiskt som möjligt så att alla, oberoende datorerfarenhet, snabbt kunde lära sig utnyttja programmet. I det här delkapitlet tar jag upp alla viktiga vyer skilt för sig. Det som gör detta delkapitel lite problematiskt är att språket i användargränssnittet är finska. I figurerna har jag därför vid behov översatt och förklarat texter och element med hjälp av orange text eller textrutor och pilar.

4.1.1 Inloggningssida

Figur 5. CMS:ets inloggningssida

I bilden ovan ser vi sidan för inloggning. Sidan är ett simpelt formulär med fält för användarnamn, lösenord och CAPTCHA (Completely Automated Public Turing-test to tell Computers and Humans Apart), samt en Logga in-knapp.

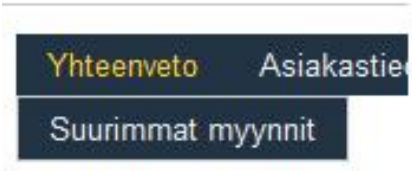
4.1.2 Header



Figur 6. CMS:ets header

Ovan ser vi CMS:ets header där vi bland annat har stommen för programmets design, nämligen huvudmenyn. Eftersom kundgrupperna och produktkategorierna inte skulle förändras på regelbunden basis lämnade jag bort administrationen av dem från CMS:et. Tabell *Lagersaldo* skulle heller inte uppdateras direkt av en administratör, utan automatiskt varje gång en ny post sattes in i *Anskaffning* eller *Försäljning*. En sida med statistik hade också önskats och så ville jag möjliggöra byte av lösenord för systemets användare, så en skild sida för den funktionen krävdes också. Huvudmenyn fick därför bestå av alternativen (från vänster till höger) Sammandrag, Kunddata, Anskaffningar, Priser, Försäljningar, Leverantörer, Produktdata och Användarkonto. Headern är alltid synlig och då man klickar på ett alternativ i menyn syns respektive innehåll undertill.

4.1.3 Sammandrag och statistik



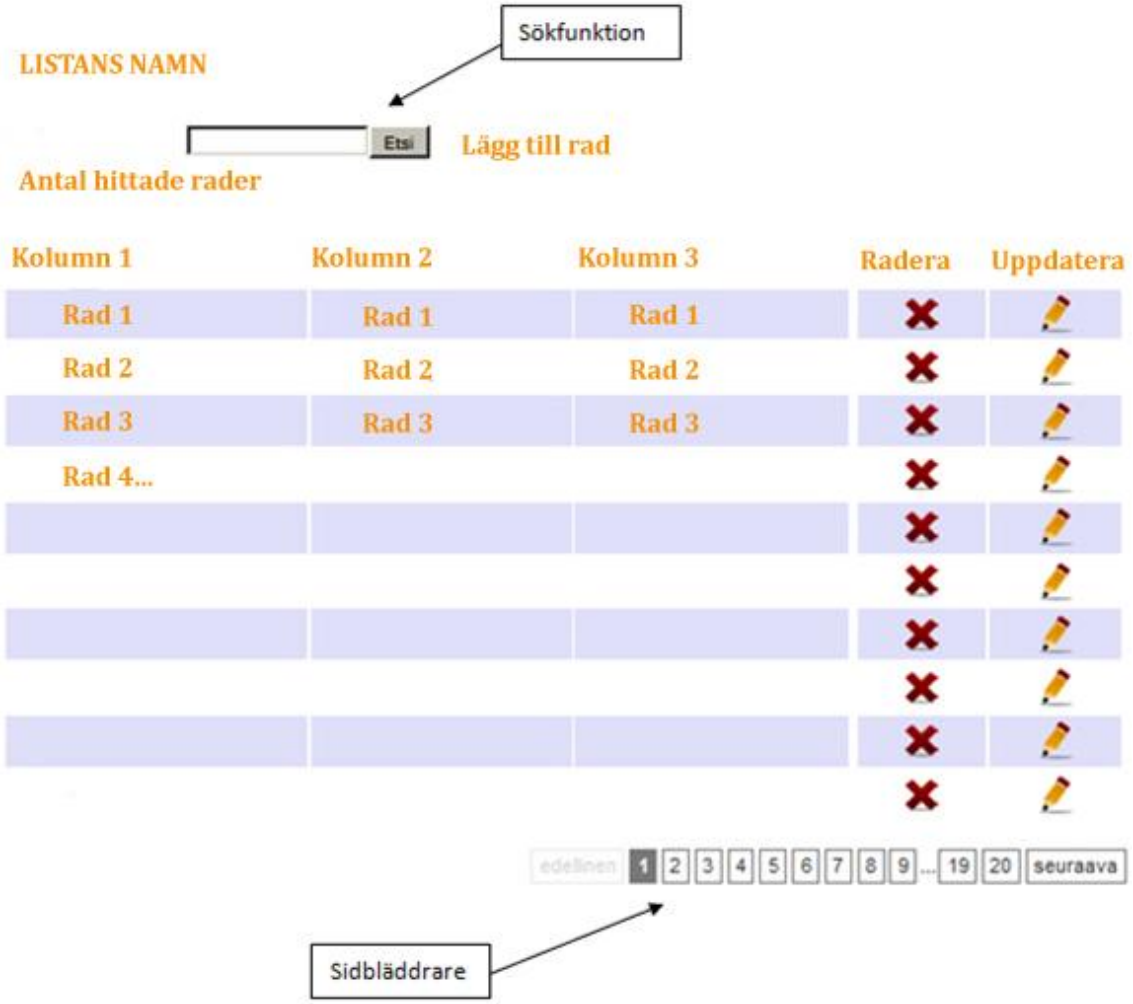
Figur 7. Huvudmenyns första alternativ ”Sammandrag” med tilläggsalternativet ”Största försäljningar”

Det första alternativet i huvudmenyn är ”Sammandrag”. Klickar man på det får man fram uträkningar på det aktuella lagervärdet - både för alla produkter över huvudtaget och skilt för produkterna inom varje produktkategori. Lagervärdet räknas genom att summera produkterna av antalet varuenheter och deras inköpspris (*SUM(antal enheter på lager * inköpspris)*). I samma vy finns inköpssumma och försäljningssumma per månad för det senaste halvåret.

För man kursorn över Sammandrag får man också fram ett annat alternativ, ”Största försäljningar”. Klickar man på det i stället får man fram en lista på de tio produkter som värdemässigt gjort största försäljning. Denna uträkning görs först för alla försäljningar och sedan skilt för försäljningar inom varje kundgrupp. Värdet räknas genom att multiplicera försäljningspriset och det sålda antalet.

4.1.4 Listning av data

I det här avsnittet berättar jag om hur data presenteras i CMS:et. Jag använder inte termerna tabell, attribut och post här eftersom datalistorna t.ex. kan bestå av hopkopplad data från olika databastabeller. En datalista ser alltså inte lika ut som en tabell. I stället använder jag termerna (data)lista, kolumn och rad.

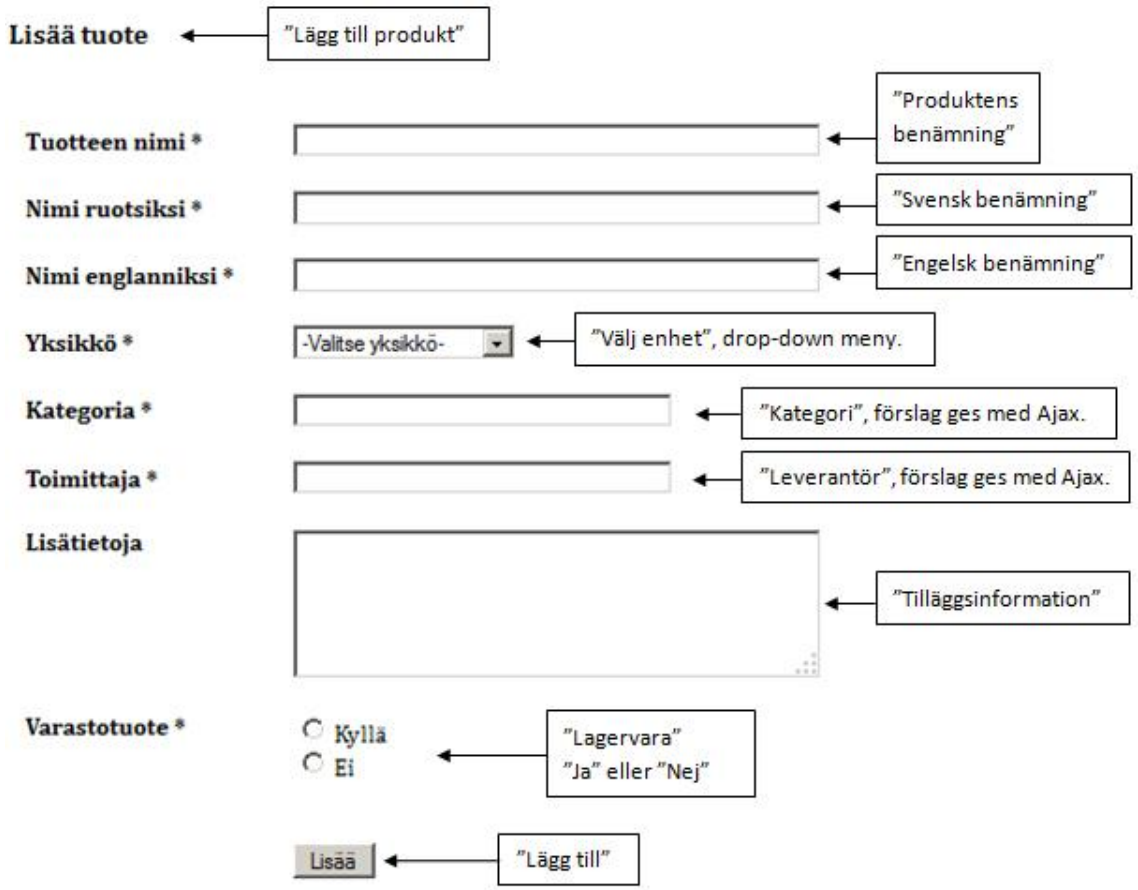


Figur 8. Modell för hur huvudmenyns andra till sjunde alternativ (Kunddata, Anskaffningar, Priser, Försäljningar, Leverantörer och Produktdata) listas i CMS:et. Texten i orange förklarar vyns innehåll på svenska.

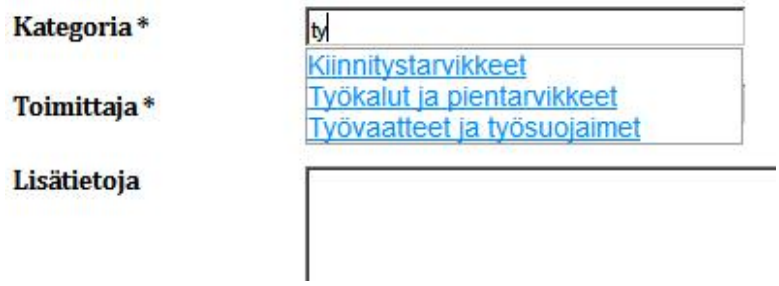
I figur 8 ser vi hur datalistorna ser ut för en administratör, för en vanlig användare saknas funktionerna för att lägga till, uppdatera och radera. Överst står listans benämning och under den finns en sökfunktion och en länk för insättning av ny rad. Datat listas sedan och 10 rader visas åt gången, förutom då man listar priser. Då man listar priser syns 11 rader per sida eftersom kundgrupperna är 11. På så sätt ser man en produkt per sida med försäljningspriser för alla kundgrupper. Vid listning av priser kan man också välja att endast se prisen för valbar kundgrupp. Då väljer man kundgrupp ur en drop-down meny vänster om sökfunktionen och 10 rader visas per sida.

En drop-down meny finns också då man klickar fram produktlistan. Ur den kan man välja att endast lista de produkter som tillhör en viss kategori. I produktlistan ser man också det aktuella lagersaldot för varje produkt.

4.1.5 Lägga till, uppdatera och radera



Figur 9. Ovanstående formulär visar oss hur det ser ut då man skall lägga in en ny produkt i databasen.



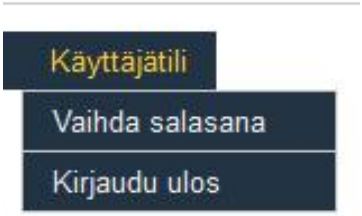
Figur 10. Ajax-meny för val av produktkategori

För att visa hur det ser ut då man lägger till, uppdaterar eller raderar en rad data har jag använt produktlistan som exempel. I figur 9 ser vi hur man lägger till en ny produkt. Sådana fält som endast kan ha specifika värden väljs ur en lista. I det här fallet är det produktens enhet, kategori och leverantör. Eftersom valbara enheter är ganska få, väljs de ur en drop-down meny. För värden som har många alternativ, här kategori och leverantör, genereras förslag ur databasen med hjälp av en Ajax-funktion. Förslagen dyker upp under fältet då användaren fyllt i så många tecken att föreslagna värden är högst tio stycken. Ett exempel på en Ajax-meny ser vi i figur 10. Då man uppdaterar en post ser formuläret likadant ut som för insättning av ny, men då är naturligtvis fälten redan ifyllda.

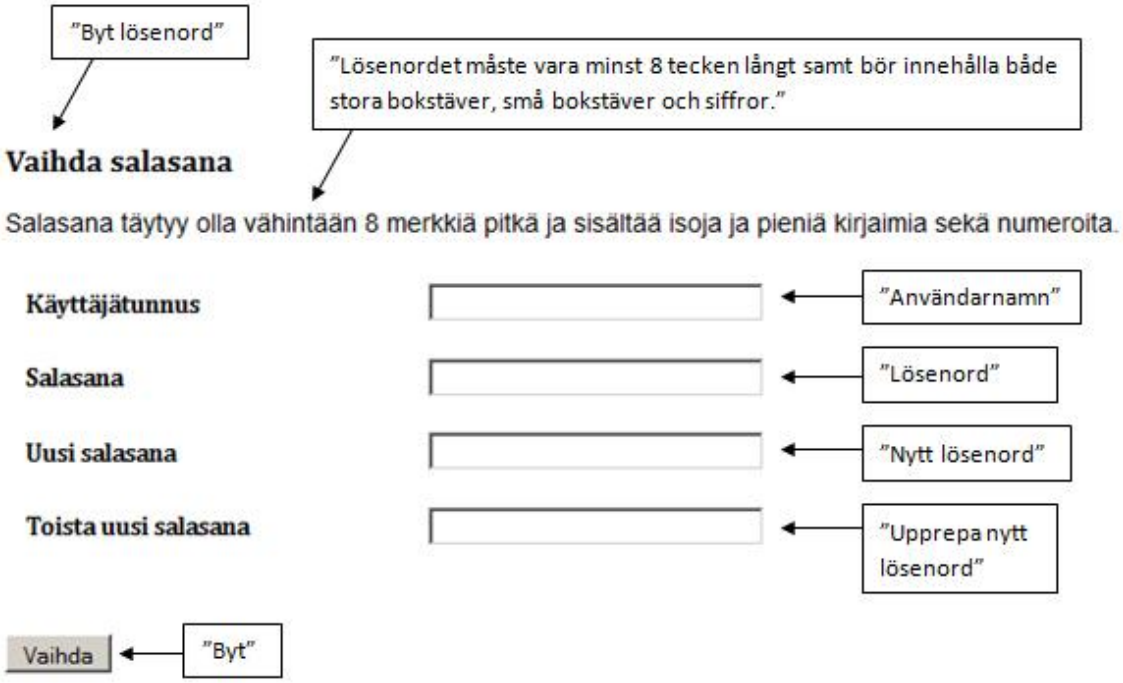
Figur 11. Alert-ruta vid radering

Klickar man på det röda krysset bredvid en rad dyker en alert-ruta upp som ber om användarens bekräftelse. Då kan användaren antingen godkänna raderingen eller ångra sig. Rutan varnar även om att all tillhörande data (i det här fallet även inköp och försäljningar av produkten t.ex.) försvinner ifall raden tas bort.

4.1.6 Användarkonto



Figur 12. Huvudmenyns sista alternativ ”Användarkonto” har två underordnade alternativ; ”Byt lösenord” och ”Logga ut”.



Figur 13. Formuläret för lösenordsbyte.

Klickar man på ”Byt lösenord” får man fram formuläret med fält för användarnamn, lösenord och nytt lösenord. Ovanför formuläret ges krav på lösenordets egenskaper. Klickar man på ”Logga ut” förstörs alla sessioner och man förflyttas till inloggningssidan.

4.1.7 Övrigt

Efter varje åtgärd (insättning, uppdatering, radering, lösenordsbyte) meddelar programmet huruvida handlingen lyckats eller inte. Orsaken till misslyckad handling ges också om den är känd.

Uppe i högra hörnet vid alla datalistor finns även en utskriftsikon. Klickar man på den öppnas en ny flik i webbläsaren med samma datalista i utskriftsvänlig version. Den utskriftsvänliga versionen är svartvit och saknar header samt kolumner för uppdatering och radering.

4.2 Innehållshanteringssystemets tekniska uppbyggnad

Det här delkapitlet behandlar tekniska lösningar så som sessionshantering, autentisering och filtrering av data.

4.2.1 Katalogstruktur

Hela CMS:et ligger i en huvudkatalog som är döpt med små bokstäver, stora bokstäver och siffror. I huvudkatalogen finns php-filen för inloggningsformuläret som är döpt efter samma principer, en index.html-fil samt underkataloger. Alla underkataloger innehåller också en index.html-fil vilken i sin tur endast innehåller en p-tag.

Figur 14. CMS:ets katalogstruktur

Som redan nämnts är användarrollerna två. Vanliga användare, som endast kan lista data och göra sökningar, har en egen underkatalog med sådana filer i som skiljer sig från administratörernas. Annars har vi underkatalogerna indelade per funktion. En katalog innehåller t.ex. alla skript för radering medan en annan katalog innehåller alla utskriftsrelaterade filer.

4.2.2 Inloggningsfunktionen

I bilaga 3 har jag illustrerat hur inloggningsformuläret och inloggningsskriptet fungerar tillsammans. Det är två skilda php-filer, formuläret är inramat i blått och skriptet i gult. De gröna pilarna representerar godkänt resultat som tillåter skriptet att fortsätta. De röda pilarna betyder att kontrollen gett felaktigt resultat och skickar användaren tillbaka till formuläret. Koden är modifierade utdrag ur de verkliga filerna men motsvarar funktionaliteten och strukturen. Om inloggningen lyckats genereras en ny token och därefter skickas användaren vidare till huvudsidan.

4.2.3 Huvudsidan

Eftersom huvudsidan skiljer sig så mycket från de andra sidorna tar jag upp dess struktur här i ett eget avsnitt. Huvudsidan är nämligen alltid synlig, det är den som ger oss Tecmarins logotyp och huvudmenybalken. Alla filer som nås direkt ur huvudmenyn inkluderas här i en iframe och dessa filer kopplar sedan vidare (genom att inkludera, länka eller skicka vidare data) till andra filer.

Högst upp i huvudsidans källkod finns en funktion som håller reda på hur lång tid som gått sedan senaste aktivitet noterats i programmet. Om användaren varit inaktiv i mer än 30 minuter loggas denne automatiskt ut. Utloggningen förstör därmed alla sessioner och förflyttar användaren till inloggningsformuläret. Denna funktion efterföljs av en annan, som i sin tur förnyar sessionsidentifikationen ifall sidan laddas om efter att det gått mer än 60 sekunder sedan senast. Sidan laddas t.ex. om varje gång något klickas på i huvudmenyn. Funktionerna ser vi nedan i figur 15.

```php
// Om senaste aktivitet mer än 30 minuter sedan...
if (isset($_SESSION['LAST_ACTIVITY'])
&& (time() - $_SESSION['LAST_ACTIVITY'] > 1800)) {
/* ...förstör alla sessioner och skicka användaren till
inloggningssidan. */
$_SESSION = array();
session_destroy();
session_unset();
header ("Location: ../ NagOtfilNman123.php");
}
// Annars uppdatera tidpunkt
$_SESSION['LAST_ACTIVITY'] = time();


/* Skapa en session med aktuell tidpunkt om en sådan
inte ännu finns... */
if (!isset($_SESSION['CREATED'])) {
$_SESSION['CREATED'] = time();
}
/* ...annars, om sessionen startade mer än 60 sekunder sedan,
generera om sessionens id och gör den gamla ogiltig med
parametern "true". */
else if (time() - $_SESSION['CREATED'] > 60) {
 session_regenerate_id(true);
```

Figur 15. Högst upp i huvudsidans fil används tidmätning av sessioner för att öka säkerheten.

Efter detta kontrolleras den token som genererades i inloggningsskriptet. Skulle den mot förmodan inte stämma, förpassas användaren till inloggningssidan och resten av skriptet dör. Om token är okej, granskas själva inloggningssessionen. Då kollas att sessionen är skapad samt att dess variabel är den rätta. Det finns nämligen två olika huvudsidor och två olika sessionsvariabler. Den ena fungerar om man är inloggad som administrator och den andra om man är inloggad med vanliga användarrättigheter.

Då skriptet säkrat att man är inloggad, och med rätt rättigheter, skapas kontakten till databasen. Sedan plockas alla behövliga bildlänkar fram, så som URL:en till logotypen och utskriftsikonen. Efter dessa följer småningom koden för huvudmenyn samt inkludering av innehållsfiler. Då någon länk i huvudmenyn klickas på, sätts en variabel för *$_SESSION['aktivitet']*. Genom att sedan kontrollera sessionens aktuella variabel vet skriptet vilken fil som skall inkluderas. Här i figur 16 följer ett modifierat utdrag ur koden för att beskriva det bättre:

```php
//PRISINFORMATION
if ($_SESSION['aktivitet']=="priser") {
include ("lista_priser.php");
}
//KUNDINFORMATION
else if ($_SESSION['aktivitet']=="kunder") {
include ("lista_kunder.php");
}
//OCH SÅ VIDARE
else if ($_SESSION['aktivitet']=="osv") {
include ("lista_osv.php");
}
```

Figur 16. Genom att kontrollera variabeln för $_SESSION[’aktivitet’] inkluderas rätt innehåll under headern på huvudsidan.

Direkt efter att man loggat in i programmet är ”sammandrag” det förinställda värdet för *$_SESSION['aktivitet’]*. Det kan tilläggas att CMS:et över huvudtaget fungerar till stor del med olika sessionsvariabler. I slutet av huvudsidans skript stängs kontakten till databasen.

4.2.4 Allmän kodstruktur

Gemensamt för alla php-skript är att de allra först startar sessionen med funktionen *session_start()*. Efter det följer öppnandet av html-taggen, header sektionen och öppnandet av body-taggen. I header-taggen länkas css-filen in. Innehållet i body-taggen börjar med inloggningskontroll och därefter skapas kontakten till databasen. Eftersom de flesta skript även behandlar någon indata behövs följande funktion för filtrering som förebyggande av SQL-injektioner:

```php
//Funktionen makeSafe() filtrerar all indata.
    function makeSafe($indata) {
    $indata = mysql_real_escape_string(trim($indata));
    return $indata;
    }
```

Figur 17. Denna funktion behövs i de flesta av CMS:ets skript för att filtrera data.

Högt uppe i skripten deklareras även alla variabler för bl.a. bilder och sidbläddrare. Efter det kommer koden för det egentliga innehållet så som formulär, SQL-satser och tabeller. Alla skript avslutas med att stänga av kontakten till databasen med funktionen *mysql_close($variabel)* och såklart sluta taggarna för body och html.

4.2.5 Att visa listor och göra sökningar

Då systemets användare skall lista data kan det finnas flera DML-satser skriptet skall välja mellan. Om man exempelvis listar kundinformation är satserna endast två – en om alla kunder listas och en annan om en sökning bland kundinformationen gjorts. Då en sökning görs skickas sökordet i formuläret till samma skript det ligger i. Ordet plockas upp och filtreras och en sessionsvariabel sätts. Med en if-else-sats kan skriptet sedan välja rätt frågesats.

```
//Fånga och filtrera sökvariabeln från sökformuläret
$letaKund=makeSafe($_POST["leta_kund"]);

//Sätt session om sökord angetts
if ($letaKund != "") $_SESSION['kundletande']=$letaKund;

//Om sökning gjorts
if(isset($_SESSION['kundletande'])) {
    $letaEfter = $_SESSION['kundletande'];
//Frågesats 1 (innehåller uttrycket LIKE '%" . $letaEfter . "%')
}

//Ej sökning, visa alla kunder
else {
//Frågesats 2
}
```

Figur 18. Ovanstående kod demonstrerar hur sökordet fångas upp och hur rätt frågesats väljs för listning av kunddata.

Vissa datalistor har dock fler möjligheter. Då man listar priser syns som sagt alla försäljningspriser för alla produkter. Med 11 stycken kundgrupper blir det 11 rader per vy så att en produkts alla försäljningspriser syns samtidigt. För att underlätta sökandet och bläddrandet kan man även välja att endast visa priserna för en viss kundgrupp. Detta gör man genom att välja en grupp ur drop-down menyn som finns placerad uppe till vänster

på sidan. Då laddas vyn om automatiskt med JavaScript funktionen *on-Change="this.form.submit()",* en session för vald kundgrupp sätts och 10 produkter med försäljningspriset för vald grupp visas sedan per vy. Det hela fungerar alltså enligt samma principer som sökfunktionen. Detta betyder att datalistan för priser har fyra olika alternativa frågesatser. En som används om ingen session är satt, en om bara sessionen för kundgrupp är satt, en om bara session för sökord är satt och en om båda sessionerna är satta vilket utför en sökning inom en kundgrupp. Rätt frågesats väljs med if-else if-sats.

4.2.6 Kontroll av data vid insättning, uppdatering och radering

Förutom att filtrera indata med funktionen *makeSafe()* (som beskrivs i avsnitt 4.2.4) utförs även andra viktiga funktioner vid hantering av det. Då en administrator skall lägga till eller uppdatera information kontrollareas t.ex. att inga obligatoriska (”not null”) värden står tomma. Med hjälp av funktionen (*preg_match('/[\'&#"+]/', $indata))* stoppas indata som innehåller specialtecken och funktionen *(filter_var($e-postadress, FILTER_VALIDATE_EMAIL)=== false)* granskar att e-postadressen står i rätt format. Värden som skall vara unika, som produktnamn och kundnamn, kontrolleras även att inte redan finnas i databasen. Sådana värden som tvärtom redan måste finnas i databasen kontrolleras också. Man kan alltså inte lägga till en produkt om dess benämning redan använts inom samma attribut. Man kan heller inte lägga till en produkt om man angett att dess leverantör är någon som inte finns i tabellen *Leverantor*. Därmed måste man välja ett alternativ ur Ajax-listan. Även andra kontroller på indata görs men jag tar inte upp varenda en.

Om skriptet upptäcker ett fel skrivs ett felmeddelande ut på skärmen, annars utförs handlingen och meddelande om att insättningen eller uppdateringen lyckats ges. Radering av data fungerar, som visserligen tidigare nämnts, genom att först be användaren bekräfta raderingen i en JavaScript alert-box. Därefter flyttar skriptet över datat till *Del_tabeller* för att slutligen radera datat från huvudtabellerna, vilket gör att datat inte längre är synligt eller åtkomligt via CMS:et men kan återkallas i phpMyAdmin.

5 PRODUKTKATALOGEN

I det här kapitlet presenterar jag först hur produktkatalogen ser ut och vilka funktioner som finns. Efter det berättar jag om vissa tekniska lösningar. Eftersom flera saker fungerar på samma sätt som i CMS:et tar jag bara upp den programmering som är annorlunda här. Båda applikationerna loggar t.ex. ut användaren efter en viss tids inaktivitet och filtrerar indata på samma sätt. Eftersom jag redan berättat om dessa funktioner behandlar jag inte dem på nytt. Jag berättar i stället om hur inloggningsskripten i applikationerna skiljer sig åt och om hur jag löste språkvalsfunktionen för produktkatalogen.

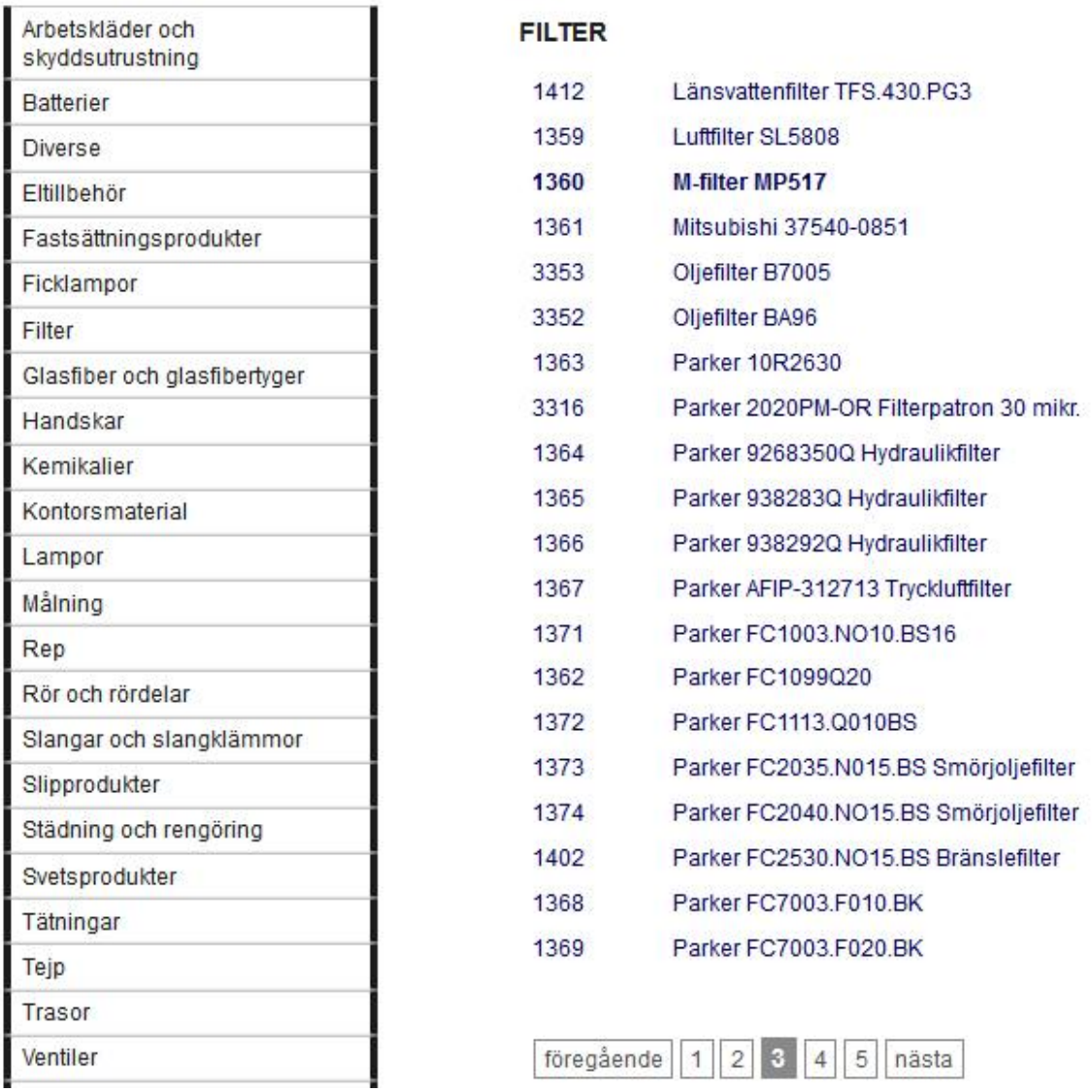
5.1 Produktkatalogens användargränssnitt och funktioner

Figur 19. Produktkatalogens inloggning. Klickar man på ”Logga in” öppnas formuläret.

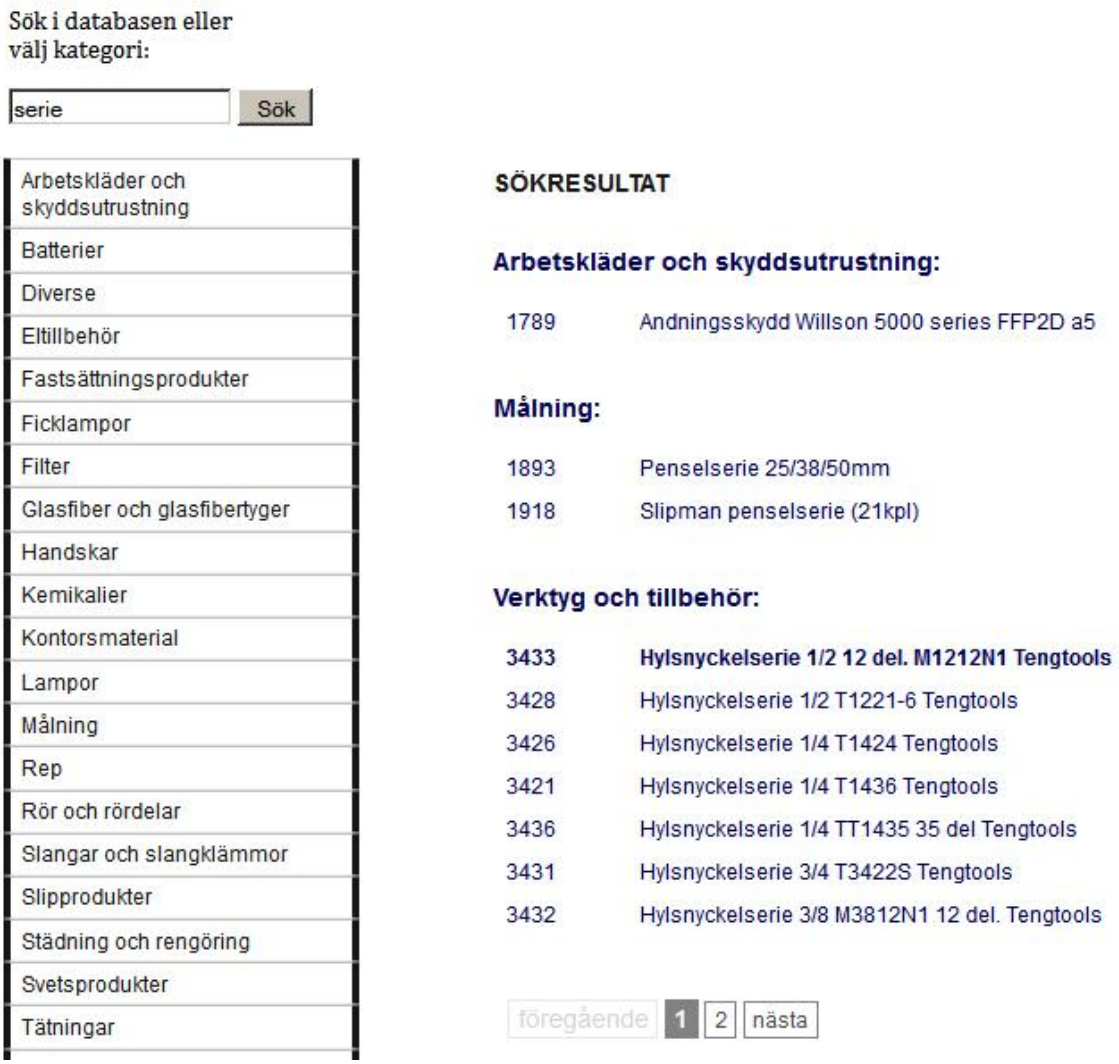
För att användaren skall nå inloggningsformuläret till produktkatalogen måste personen först klicka på en länk. Jag valde den lösningen i stället för CAPTCHA för att göra funktionen mer användarvänlig.

Figur 20. Produktkatalogens huvudsida. Då man för kursorn över en kategori i menyn markeras den i blått.

I ovanstående figur ser vi en bit av huvudsidan i produktkatalogen. Här kan användaren byta mellan språken finska, svenska och engelska, lista produkter per kategori eller genom att göra en sökning samt logga ut ur programmet. Både kategorierna i menyn och listorna med produkter skrivs ut i alfabetisk ordning. Då man för kurson över en länk markeras den, i menyn ser vi hur alternativet blir blått och i följande figur hur produktens kod och benämning får en fetare font.



Figur 21. Klickar man på en kategori i menyn listas produkterna till höger. För man kursorn över en produkt transformeras texten till fet stil.



Figur 22. Gör man en sökning listas träffarna inom sina produktkategorier. Här var sökordet ”serie”.

I figur 22 ser vi hur sökträffarna läggs fram. De delas upp i respektive produktkategorier för att ge en klarare översikt. Då man sedan klickar på en produkt får man fram mer specifik information vilket framställs i figur 23.

PRODUKTINFORMATION

Produktkod:	3433
Namn:	Hylsnyckelserie 1/2 12 del. M1212N1 Tengtools
Enhet:	set
På lager:	ja
Pris:	X.00 €

Fråga tilläggsinformation om produkten

Figur 23. Klickar man på en produkt får man fram mer specifik information.

FRÅGA OM PRODUKTEN

Produktkod och -namn:	3433 Hylsnyckelserie 1/2 12 del. M1212N1 Tengtools
***Kund:**	
***Förnamn:**	
***Efternamn:**	
Telefonnummer:	
E-post:	
Meddelande:	

Skicka

Figur 24. I figur 23 ser vi länken ”Fråga tilläggsinformation om produkten”. Den öppnar formuläret ovan.

Formuläret i figur 24 ger kunden möjlighet att enkelt ta kontakt med Tecmarin angående en viss produkt. Förutom den information som användaren matar in i formuläret skickas även namnet på den kundgrupp användaren är inloggad som. Formuläret skickas till Tecmarins gemensamma e-postadress.

5.2 Produktkatalogens tekniska lösningar

Efter att jag programmerat hela innehållshanteringssystemet hade jag redan fått mycket bättre kunskap i PHP. Därför var det ingen svår match att sedan bygga produktkatalogen, det var ju mest att plocka fram lite data ur databasen och koda ett formulär. Det gick nästan mer tid åt att arbeta fram en snygg layout och skriva CSS-filen. Jag anser att jag inte har något vidare bra grafiskt öga så satsade därför på ett enkelt och tydligt utseende.

Det var också bekvämt att kunna återanvända CMS:ets inloggningssystem. Några ändringar gjorde jag ändå i det, som att lämna bort CAPTCHAn. I stället skall användaren klicka på en länk för att skapa en session. Sessionen tillåter sedan inloggningsformuläret att köras. Då formuläret sedan skickas till inloggningsskriptet innehåller det, liksom i CMS:et, en token. Inloggningsskriptet i produktkatalogen loggar också användarnamn, IP och tidpunkt men gör det oberoende om angivet användarnamn existerar eller ej (till skillnad från CMS:et, se bilaga 3 för jämförelse).

I produktkatalogen har alla 11 identiteter samma användarroll. Men för varje identitet, dvs. kundgrupp, sätts en egen sessionsvariabel så att systemet vet vilken kundgrupp som är inloggad och plockar fram försäljningspriserna därefter.

Det som ändå är helt nytt i produktkatalogen är möjligheten till språkval. Det sköttes genom att lägga till översättningar i databasen för produkt- och kategoribenämningarna samt skriva ett skilt PHP-skript för andra översättningar som sedan inkluderades i alla övriga skript. I följande figur visas en förkortad version av filen.

```php
<?php
session_start();
if (empty($_SESSION['language'])) {
$_SESSION['language'] = "fin";
}

//FINNISH
if ($_SESSION['language'] == "fin") {
$theh1text = "Tuoteluettelo";
$shortinfotxt = "Pidätämme oikeuden kirjoitusvirheisiin ja hinnanmuutoksiin.";
$category = "kategori";
$product_name = "produktnamn";
$search = "Etsi";
$welcome = "Tervetuloa";
}

//SWEDISH
else if ($_SESSION['language'] == "swe") {
$theh1text = "Produktkatalog";
$shortinfotxt = "Vi förbehåller oss rätten till tryckfel och prisförändringar.";
$category = "kategori_sv";
$product_name = "produktnamn_sv";
$search = "Sök";
$welcome = "Välkommen";
}

//ENGLISH
else if ($_SESSION['language'] == "eng") {
$theh1text = "Product catalog";
$shortinfotxt = "We reserve the right to printing errors and price changes.";
$category = "kategori_en";
$product_name = "produktnamn_en";
$search = "Search";
$welcome = "Welcome";
}
?>
```

Figur 25. En förkortad och modifierad version av översättningsskriptet.

```php
//Div for info text
echo "<div style=\"position: absolute; z-index: 40;
left: 1%; top: 940px;\" id=\"infotxt\">
<comment>" . $shortinfotxt . "</comment>
</div>";
```

Figur 26. Ett utdrag ur huvudsidans kod för att demonstrera hur översättningsvariablerna i figur 25 används.

Variabeln för *$_SESSION['language']* sätts då man klickar på en av språkflaggorna som finns uppe på huvudsidan (se figur 20). Genom att inkludera översättningsfilen (figur 25) i de andra källkodsfilerna kan dess variabler användas så som demonstreras i figur 26. Även i SQL-satserna används översättningsvariablerna för att plocka attribut på rätt språk.

Det finns även andra saker som skiljer koden i de två applikationerna åt men det är mest frågan om detaljer, så som att den automatiska utloggningen sker efter lite kortare tid i produktkatalogen, att funktionen *session_regenerate_id(true)* även används på andra sidor än på applikationens huvudsida och så vidare. Men det är onödigt att gå in på sådana detaljer. Påpekas kan ändå att jag insåg att kommentering av källkoden hade varit bristfällig i CMS:et så i produktkatalogens kod såg jag till att kommentera noggrant på engelska. I CMS:et hade jag mitt eget språk, en mix av svenska, finska och engelska, som troligtvis skulle vara svårläst för någon annan.

6 ANALYS OCH UTVÄRDERING

I det här kapitlet skall göra en utvärdering av datasystemet. Jag tar hjälp av artiklar om datasäkerhet, en experts tankar om systemets datasäkerhet och användarnas åsikter. På nätet har jag forskat om datasäkerhet i PHP-baserade webbapplikationer genom att läsa en säkerhetsguide för PHP, artiklar på teknikinriktade bloggar samt specifikt om vissa funktioner. Jag nämner även mina egna lösningar för ökad säkerhet. Dessutom skrev jag ett brev åt en dataexpert i vilket jag summerade upp de säkerhetsåtgärder jag vidtagit för databasen och webbapplikationerna. Jag bad om hans åsikter och tankar om huruvida systemet är säkert och vad jag kunde göra för att höja säkerheten ytterligare. För utvärdering av själva användarupplevelsen och projektets slutresultat i sin helhet bad jag systemets användare svara på sex stycken frågor. Eftersom produktkatalogen inte ännu tagits i bruk av så många kunder intervjuade jag endast Tecmarins anställda.

6.1 Säkerhet

Datasäkerheten är en viktig fråga i systemet. Datasäkerhet är dock ett väldigt djupt ämne och jag kan inte gå in på detaljnivå eftersom det skulle bli för omständigt. Vi skall

också komma ihåg att det här datasystemet inte innehåller lika känslig data som i en nätbank till exempel. Eftersom databasen och applikationerna ligger på en hosting server som utför säkerhetskopiering en gång per dygn, är eventuell förlust eller manipulering av data inte så mycket att bekymra sig om i och med att det skall gå att återställa. Den största säkerhetsrisken här är att någon obehörig får tillgång till skillnaderna i försäljningspriserna eftersom det är affärshemligheter. Därför är den viktigaste biten inloggningsfunktionen. Jag delar in det här delkapitlet i tre avsnitt; Skydd mot typiska attacker, Övriga säkerhetsfrågor och Bedömning av dataexpert.

6.1.1 Skydd mot typiska attacker

För att kunna utvärdera systemets säkerhet läste jag bl.a. säkerhetsguiden för PHP på phpsec.org. I guiden tas upp hur man skyddar sitt system mot de vanligaste typerna av attacker så som XSS (Cross-Site Scripting) och CSRF (Cross-Site Request Forgery). Den allra viktigaste åtgärden över huvudtaget, men också mer specifikt för att skydda systemet mot XSS-attacker och SQL-injektioner, är att filtrera *all* extern data (PHP Security Consortium). Detta gör jag konsekvent med funktionen makeSafe som presenteras i figur 17. Modell för funktionen är tagen ur artikeln på addedbytes.com (Child 2004). Förutom makeSafe valideras data även på andra sätt, beroende på datats karaktär (se kapitel 4.2.6). I inloggningsfunktionen räknas t.ex. även antalet tecken. De huvudsakliga åtgärderna för att skydda applikationen mot CSRF är att använda POST i ställer för GET som metod i formulär, samt att kontrollera att formuläret som använts är det rätta genom att bifoga en dold token. I systemet används metoden POST i inloggningsformulären och även i de flesta andra. I inloggningsformulären genereras även en token i enlighet med rekommendationerna i säkerhetsguiden.

Ur säkerhetsguidens kapitel om sessionshantering framgår att de vanligaste attackerna mot sessioner är session fixation och session hijacking (PHP Security Consortium). För att förhindra dessa har jag tagit till funktionen *session_regenerate_id(true)* som körs ofta i båda applikationerna. Parametern *true* raderar den gamla sessionsdatan (The PHP Group).

Det sägs också att register_globals bör vara ur bruk vilket är standard fr.o.m. PHP 4.2.0 (PHP Security Consortium). Jag kontrollerade ändå detta med resultatet:

register_argc_argv	Off	Off
register_globals	Off	Off
register_long_arrays	Off	Off

Figur 27. Med funktionen phpinfo() kontrollerade jag att register_globals säkert är ur bruk.

Då jag designade inloggningen, som i stort sett ser likadan ut i båda applikationerna, tog jag modell av två artiklar i vilka man ger exempel på säkra inloggningsfunktioner i PHP. Ur artikeln på tinsology.net lärde jag mig att använda säker hashfunktion och salt för lösenord samt fick tips om sessionshantering. Här följer ett sammandrag på artikelns konkreta och relevanta råd av vilka alla är implementerade i systemet:

- Använd POST, inte GET, som metod i inloggningsformuläret.
- Använd hashade lösenord och salt.
- Kontrollera att användaren är inloggad m.h.a. sessionsvariabler.
- Filtrera all indata för att skydda systemet från SQL-injektioner.
- Förstör alla sessioner vid utloggning.
- Vid byte av lösenord, be användaren först ange det gamla.

(Tinsology.net)

Jag jämförde informationen jag fått ur artikeln på tinsology.net med en liknande artikel på devshed.com. I början av den finns en redogörelse på vilka viktiga egenskaper deras exempel på inloggningssystem innehåller. De säkerhetshöjande egenskaperna som är relevanta för mitt system är följande:

- Ett hashat lösenord i sha256 samt salt.
- Förhindra brute force genom att först använda CAPTCHA och sedan förbjuda tillträde efter att inloggningsförsöken per IP-adress är fler än X antal.
- Skydd mot session hijacking och fixation.
- Automatiserad förstörelse (utloggning) av sessioner efter viss tids inaktivitet hos användaren.
- Autentisering vid utloggning eller radering av data för att förhindra CSRF.
- Inget val för ”kom ihåg mig” -funktion.

- Filtrering av indata för att förhindra SQL-injektioner och de vanligaste formerna av XSS-attacker.
- Förhindra direkt åtkomst till autentiseringsskript.
- Kryptera inloggningssessionerna genom att använda HTTPS (Hypertext Transfer Protocol Secure).

(Developer Shed Network)

Av dessa egenskaper finns det några som mitt system saknar. En av dem är att systemet inte räknar inloggningsförsök och därmed kan blocka en IP-adress efter ett visst antal misslyckade försök. Jag använder heller inte HTTPS-protokoll i systemet. Att kunna blocka misstänkta IP-adresser och/eller bara tillåta vissa IP-adresser skulle dock relativt enkelt kunna tilläggas för att höja säkerheten lite. Att övergå till HTTPS skulle utan tvekan höja säkerheten ganska mycket men min första tanke var att det ändå är lite onödigt i det här sammanhanget. Vid närmare eftertanke kom jag fram till att jag nog skall forska i saken, det skulle kanske löna sig ändå. Vad jag lade till utöver ovanstående egenskaper är att logga varje inloggning eller inloggningsförsök med använt användarnamn, IP-adress och tidpunkt.

6.1.2 Övriga säkerhetsfrågor

Förutom risken att systemet utsätts för en extern attack finns det även andra, kanske mer troliga saker som kan ställa till med skada. I och med att jag nu under skrivprocessen synat applikationernas kod på nytt har jag också hittat en del bristfälligheter som kan ställa till med bekymmer. Ett av dessa är ifall en användare byter lösenord i CMS:et. I instruktionerna står att det nya lösenordet bör vara minst åtta tecken långt och innehålla både stora och små bokstäver samt siffror. Det enda skriptet dock egentligen kollar är att fältet för det nya lösenordet inte är tomt (och såklart att det gamla lösenordet stämmer). Men detta betyder att en användare kan ändra sitt lösenord till vilket enkelt och förutsägbart ord som helst. En annan säkerhetsrisk angående lösenord som jag varit medveten om redan länge men inte tagit itu med, är att jag inte kontrollerar att användarna verkligen byter lösenord med jämna mellanrum, vilket jag har rekommenderat. Att byta lösenord varje månad t.ex. skulle höja säkerheten en aning.

En annan, kanske ännu värre brist i koden finns i funktionerna för radering av data. Tanken är ju att allt som användaren raderar bara skall flytta till en annan tabell. Huvudtabellernas främmande nycklar är ju försedda med ON DELETE CASCADE men raderar användaren t.ex. en leverantör i CMS:et så flyttar skriptet bara just det data (leverantörens namn och kontaktuppgifter) till en del_-tabell. Det betyder att allt annat relaterat data, så som produkter, anskaffningar och försäljningar verkligen försvinner. Skulle användaren göra en sådan här radering och sedan ångra sig skulle man måsta kontakta Netsor för att få den senaste säkerhetskopian av databasen för att återställa misstaget. Så hade jag ju inte tänkt att det skulle fungera, detta måste åtgärdas så snart som möjligt.

Annars anser jag att jag lyckats ganska bra eftersom insättning och uppdatering av data valideras noggrant och inloggningssessionen kontrolleras i varje skript som kan visa eller manipulera information.

6.1.3 Bedömning av dataexpert

För att få en objektiv bedömning på systemets säkerhet skickade jag ett brev per e-post åt en helt utomstående person inom IT-branschen. Jag summerade upp datasystemets egenskaper för datasäkerhet och bad dataexperten Arto Peterzens ge sitt utlåtande på basis av det. Peterzens är utbildad diplomingenjör och arbetar som datachef på Algol Oy Ab. Brevet är skrivet på engelska och finns med som bilaga 4. Svaret jag fick är bilaga 5 och är skrivet på finska.

I Arto Peterzens svar fick jag beröm för en klar och tydlig säkerhetsbeskrivning. Han sade att det bästa beviset på att databasen är fungerande och välplanerad är att den faktiskt varit i daglig användning i 10 månader utan att ha några problem. Lite roligt var att han påpekade att ON UPDATE CASCADE av främmande nycklar nog tyder på god praxis med de facto är onödigt här eftersom alla primärnycklar skapas av AUTO_INCREMENT och uppdateras ej. Det hade jag inte tänkt på.

Peterzens menade också att inloggningsproceduren var okej, han hade också försökt komma in i produktkatalogen utan resultat. Hans förbättringsförslag var att byta till HTTPS, vilket jag skall överväga. Peterzens undrade också ifall lösenorden är krypterade i databasen, vilket de är. Annars fick jag bra feedback för användarrollerna, radering av data och att spara information om inloggningsförsök.

Slutligen funderade Peterzens på ifall det kan vara en säkerhetsrisk att servern som systemet ligger på är delad. Ofta är det så på webbhotell men det är klart att en skild server skulle vara säkrare. Enligt säkerhetsguiden är den största risken med delade servrar att sessionerna lagras på samma ställe, standard är att alla lagras i /tmp. För att gå runt det här säkerhetshålet skulle man t.ex. kunna lagra alla sina sessioner i en databastabell i stället. (PHP Security Consortium)

Säkerhetsbeskrivningen var m.a.o. välformulerad och feedbacken tydde på att systemet inte innehåller några allvarligare riskfaktorer. Samtidigt skall vi komma ihåg att Peterzens inte är expert på datasäkerhet inom PHP och MySQL utan känner till området på en mer allmän nivå. Före han gav sitt svar hade han dock läst sig in på området.

6.2 Funktionalitet och användarvänlighet

Databasen har inte användarna någon direkt erfarenhet av så det är en bit jag får utvärdera på egen hand m.h.a. litteratur. Applikationerna har jag däremot bett om feedback för av användarna själva.

6.2.1 Databasen

Jag är riktigt nöjd med databasen, den har möjliggjort alla funktioner som behövts i applikationerna. Databasen är nästan helt normaliserad, men en brist förekommer i 1NF (första normalformen). För att en databas skall vara normaliserad enligt 1NF skall den inte innehålla några attribut som är likadana, alla fält i varje post skall endast ha ett värde och alla poster inom varje attribut skall vara av samma datatyp. I vissa av tabellerna i databasen lagras faktiskt flera värden inom samma attribut. I tabell *Leverantor* kan *kontaktperson* bestå av både för- och efternamn. Dessa kunde spjälkas upp till *kontaktperson_fornamn* och *kontaktperson_efternamn.* Samma sak gäller i tabell *Kund* t.ex., där *faktureringsadress* kunde delas upp i gatuadress, postnummer, stad och land. Ingendera tabellen är dock för tillfället i så kraftig användning eftersom företaget bl.a. sköter faktureringen via ett annat system. Att spara kontaktuppgifterna om kunder och leverantörer (och inte endast deras namn och relation till priser och produkter) var mera tänkt som en enkel tilläggsfunktion, de är inte en central del i systemet. Skulle det dock stå inför vidareutveckling, kan det hända att dessa attribut skulle behöva spjälkas upp i fler.

Då skulle även tabellerna bli fler om man skulle vilja spara kontaktuppgifterna för fler personer inom samma bolag eller kanske ha flera olika telefonnummer per kontaktperson.

För att en databas skall uppfylla 2NF skall alla fält, som inte är nycklar, vara direkt relaterade till nyckeln och för 3NF krävs det att det inte finns några transitiva beroenden. (Björk 2003 kapitel 6) Detta har jag sett till genom att ha en skild tabell för varje typ av information och kopplar bara ihop dem m.h.a. nycklar som alla sköts av AUTO_INCREMENT för att minimera lagringsutrymme. Det finns endast en primärnyckel per tabell och inget data lagras på flera olika ställen så redundansen är obefintlig. Man kan alltså säga att databasen skulle vara helt normaliserad ifall man ännu delade upp den typ av attribut som nämndes i föregående stycke. Å andra sidan kunde man tycka att databasen övernormaliserats då, eftersom det skulle vara onödigt i de nuvarande applikationerna och mödan att skriva de extra JOIN-satserna skulle vara större än vad normaliseringen skulle spara.

Fördelarna med en normaliserad databas är att innehållet blir enkelt och effektivt att komma åt och upprätthålla, lagringutrymmet som krävs blir möjligast litet, man förebygger att ofullständig eller felaktig data lagras och ställer till med problem och så är flexibiliteten stor för framtida ändringar och vidareutveckling. (W3Schools)

6.2.2 Applikationerna

Då jag designade de två applikationerna gjorde jag mitt bästa för att de skulle bli riktigt enkla och användarvänliga. Eftersom ingendera består av så väldigt många funktionaliteter var det ganska enkelt att få det logiskt uppställt.

På Tecmarin är det sex stycken anställda som sysslar med försäljning eller fakturering och har användarrättigheter till CMS:et. Av dessa svarade fem på mina frågor om användarupplevelsen och slutresultatet i sin helhet, den sjätte har inte utnyttjat systemet i sitt arbete och svarade därför inte. I bilaga 6 har vi frågorna och svaren. Svaren var över huvudtaget väldigt positiva och alla anser att CMS:et är enkelt, eller ganska enkelt, att använda.

Av svaren att döma finns det ännu rum för förbättring i sökfunktionerna och inloggningsfunktionen. Sökningarna skulle lätt kunna vidareutvecklas så att man ur en drop-down meny kunde välja vilken typ av data det är man söker. Skulle man t.ex. välja ”Produktkod” i drop-down menyn och sedan skriva in siffrorna man söker på i formulärets fält, skulle sökningen effektiveras. Inloggningen å sin sida skulle kunna göras mer användarvänlig genom att kräva CAPTCHA först efter andra eller tredje inloggningsförsöket. Det skulle inte sänka säkerheten, bara förbättra användarupplevelsen.

Annars anser alla att datasystemet på ett eller annat sätt effektiverat arbetsrutinerna och ger företaget en mer professionell image. Jag är också själv nöjd i stort sett, allt har ju fungerat hittills. Men skulle jag ge mig in på ett liknande arbetsuppdrag igen skulle jag från början vara noga med att kommentera koden ordentligt och på engelska. Skulle det vara frågan om ett mer omfattande system tror jag också att objektorienterad programmering är att föredra. Det är nog det här systemets största nackdel att det kan vara svårt för någon annan än mig att göra ändringar i det. Det är inte heller speciellt lämpligt att integrera hos andra företag eftersom det är helt skräddarsytt för Tecmarin. Att arbeta tillsammans med en grafiker skulle också vara trevligt i framtida projekt eftersom jag inte är så insatt i eller intresserad av grafisk design. I vissa fall kan ju utseendet på användargränssnittet vara väldigt viktigt.

6.3 Resultat

Efter att ha granskat datasäkerheten med hjälp av säkerhetsguiden för PHP, begrundat andra säkerhetsfrågor, själv hittat ett par brister i min källkod och fått ett utlåtande av en dataexpert tycker jag att jag behandlat datasäkerheten tillräckligt. Det finns förbättringar att göra men i stort sätt tror jag säkerhetsnivån är hög nog för ändamålet. Eftersom databasen är så gott som normaliserad och kritiken jag fick av systemanvändarna var så positiv, anser jag att jag lyckats med projektet. Jag har lärt mig väldigt mycket och intresset för området har vuxit. Jag är väldigt glad att Tecmarins anställda anser att det här datasystemet har försnabbat arbetsrutinerna och gett mervärde åt företaget.

KÄLLOR

Arto Peterzens
CIO
Algol Oy Ab
Tel. +358 9 509 9227
E-post: arto.peterzens@algol.fi

Björk, Jonas. 2003, *Databashantering*, [www].
Tillgänglig: http://www.rejas.se/fritis/databashantering/book1.html Hämtad: 16.4.2013

Charlotta Åberg-Ohlström
Fakturering, marknadsföring
Oy Tecmarin Ship Supply Ab
Tel. +358 20 155 8256
E-post: charlotta.aberg@tecmarin.fi

Child, Dave. 2004, *Writing Secure PHP, Part 1*, Added Bytes Ltd, [www].
Tillgänglig: http://www.addedbytes.com/articles/writing-secure-php/writing-secure-php-1/ Hämtad: 16.4.2013

Developer Shed Network. 2011, *Simple and Secure PHP Login Script*, [www].
Tillgänglig: http://www.devshed.com/c/a/PHP/Creating-a-Secure-PHP-Login-Script-59941/ Hämtad: 16.4.2013

Fredrik Ohlström
Försäljning
Oy Tecmarin Ship Supply Ab
Tel. +358 20 155 8254
E-post: fredrik.ohlstrom@tecmarin.fi

Kristina Mähönen
Försäljning, logistik
Oy Tecmarin Ship Supply Ab
Tel. +358 20 155 8255
E-post: kristina.mahonen@tecmarin.fi

Lars-Erik Åberg
Försäljning
Tecmarin Ship Supply Ab
Tel. +358 20 155 8252
E-post: lars-erik.aberg@tecmarin.fi

Netsor Oy. *Webhotellit ja verkkotunnukset*, [www].
Tillgänglig: http://netsor.fi/webhotellit-ja-verkkotunnukset.html Hämtad: 16.4.2013

Nina Ahlmark
Fakturering, reskontra
Oy Tecmarin Ship Supply Ab
Tel. +358 20 155 8257
E-post: nina.ahlmark@tecmarin.fi

Oracle Corporation. *About MySQL*, [www].
Tillgänglig: http://www.mysql.com/about/ Hämtad: 16.4.2013

Oy Tecmarin Ship Supply Ab. *Om företaget*, [www].
Tillgänglig:
http://www.tecmarin.fi/index.php?option=com_content&view=article&id=56&Itemid=60&lang=sv Hämtad: 16.4.2013

PHP Security Consortium. *PHP Security Guide,* [www].
Tillgänglig: http://phpsec.org/projects/guide/ Hämtad: 16.4.2013

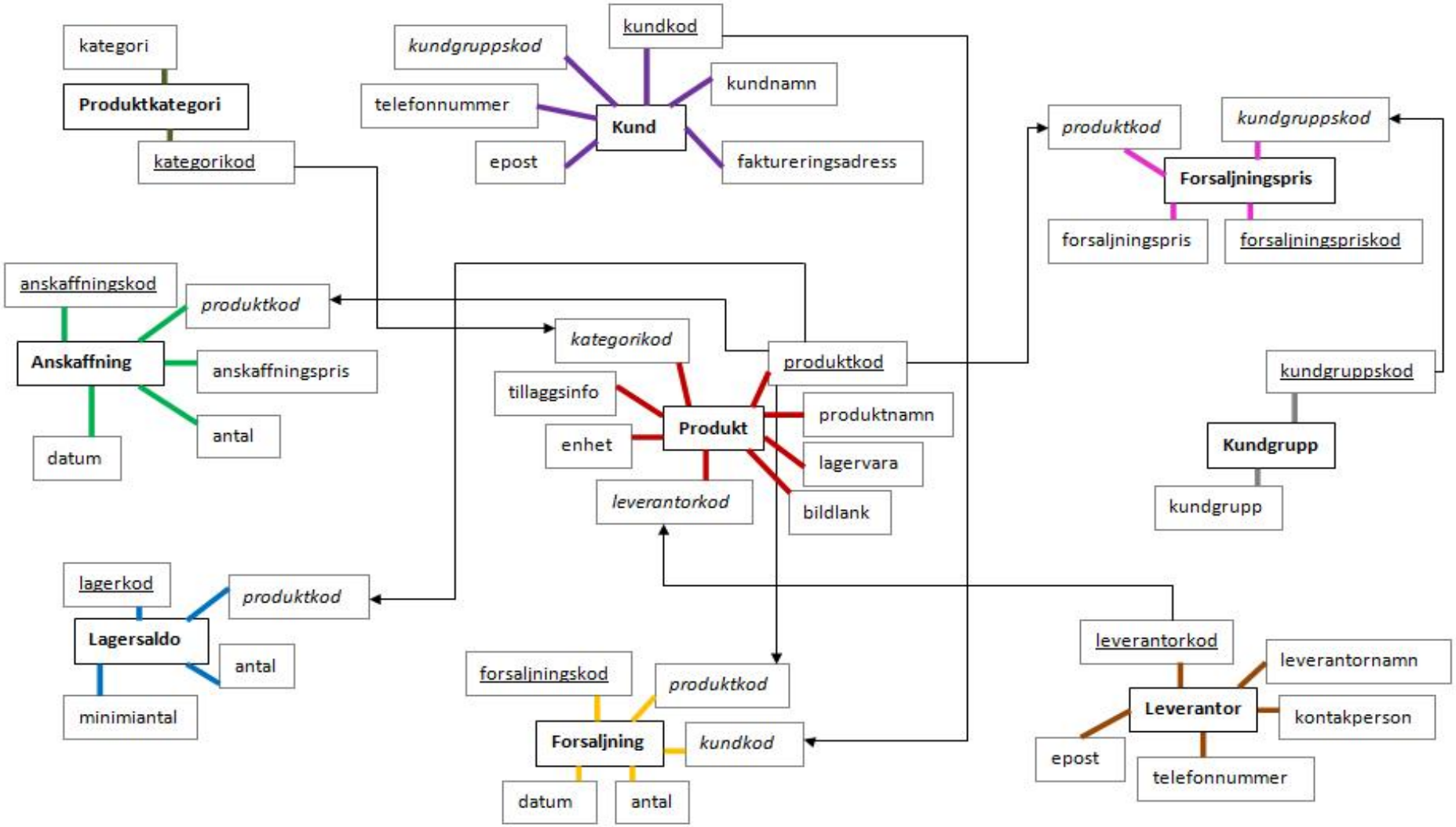
Taloussanomat. *Tecmarin Ship Supply Ab Oy, Taloustiedot,* [www].
Tillgänglig: http://yritys.taloussanomat.fi/y/tecmarin-ship-supply-ab-oy/helsinki/0625818-4/ Hämtad: 16.4.2013

The PHP Group. *PHP Manual - Session functions - session_regenerate_id,* [www].
Tillgänglig: http://php.net/manual/en/function.session-regenerate-id.php Hämtad: 16.4.2013

Tinsology.net. 2009, *Creating a Secure Login System the Right Way,* [www].
Tillgänglig: http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way Hämtad: 16.4.2013

W3Schools. 2012, *MySQL Tutorial – Database Normalization,* [www].
Tillgänglig: http://www.w3schools.in/mysql/database-normalization/ Hämtad: 16.4.2013

BILAGA 1: DIAGRAM ÖVER HUVUDTABELLERNAS ATTRIBUT OCH RELATIONER



BILAGA 2/1(3): HUVUDTABELLERNAS DATALEXIKON

Tabell	Attribut	Beskrivning	Datatyp och -längd	Nyckel och egenskaper	Andra begränsningar	Nödvändighet
Leverantor	leverantorkod	unikt, identifierande nummer för varje leverantör	INT	PK	AUTO_INCREMENT	not null
	leverantornamn	unikt namn på leverantören	VARCHAR(60)			not null
	kontaktperson	för- och efternamn på kontaktpersonen hos leverantören	VARCHAR(40)			null
	telefonnummer	telefonnumret till kontaktpersonen hos leverantören	VARCHAR(40)			null
	epost	leverantörens e-post adress	VARCHAR(40)			null
Produktkategori	kategorikod	unikt, identifierande nummer för varje produktkategori	SMALLINT	PK	AUTO_INCREMENT	not null
	kategori	produktkategorins namn	VARCHAR(40)		UNIQUE	not null
Produkt	produktkod	unikt, identifierande nummer för varje produkt (produktnamn, inte enskild vara)	INT	PK	AUTO_INCREMENT	not null
	produktnamn	benämningen på produkten	VARCHAR(90)		UNIQUE	not null
	kategorikod			FK ON DELETE CASCADE ON UPDATE CASCADE		not null
	leverantornamn			FK ON DELETE CASCADE ON UPDATE CASCADE		not null
	enhet	i vilken enhet produkten säljs/köpes; kg, styck, l, set	VARCHAR(50)			not null
	tillaggsinfo	tilläggsinformation om produkten	VARCHAR(200)			null
	bildlank	länk till eventuell bild på produkten ../katalog/bild.jpg	VARCHAR(80)			null
	lagervara	antingen ja eller nej (1 eller 0)	TINYINT(1)		DEFAULT 0	not null

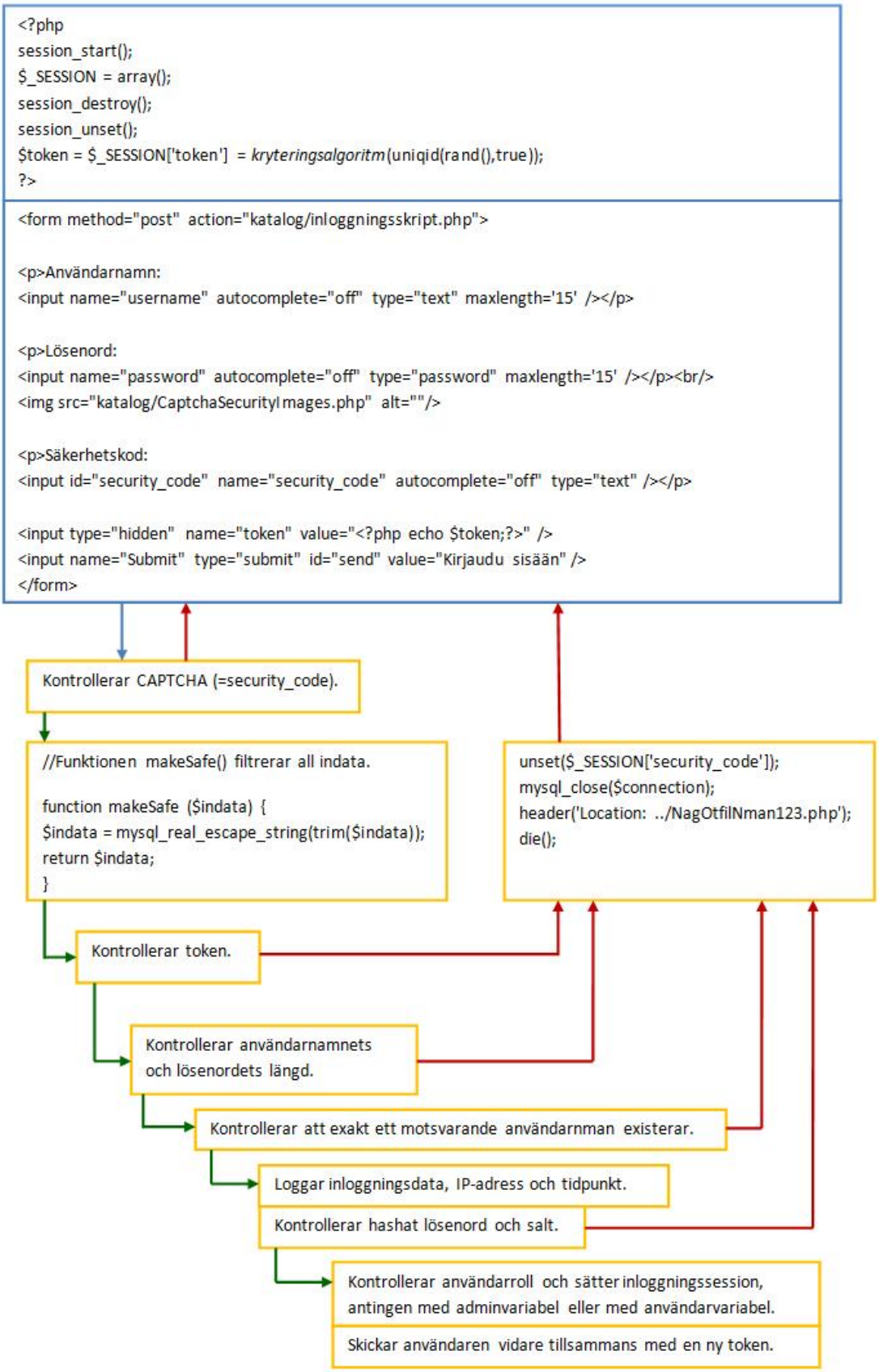
BILAGA 2/2(3): HUVUDTABELLERNAS DATALEXIKON

Kundgrupp	kundgruppskod	unikt, identifierande nummer för varje kundgrupp	SMALLINT	PK	AUTO_INCREMENT	not null
	kundgrupp	kundgruppens namn	VARCHAR(40)		UNIQUE	not null
Kund	kundkod	unikt, identifierande nummer för varje kund	SMALLINT	PK	AUTO_INCREMENT	not null
	kundnamn	kundens/kundbolagets namn	VARCHAR(60)		UNIQUE	not null
	faktureringsadress	kundens adress	VARCHAR(100)			null
	epost	kundens e-post adress	VARCHAR(40)			null
	telefonnummer	kundens telefonnummer	VARCHAR(40)			null
	kundgruppskod			FK ON DELETE CASCADE ON UPDATE CASCADE	DEFAULT 1	not null
Forsaljningspris	forsaljningspriskod	unikt, identifierande nummer för varje försäljningspris beroende på kundgrupp och produkt	BIGINT	PK	AUTO_INCREMENT	not null
	kundgruppskod			FK ON DELETE CASCADE ON UPDATE CASCADE		not null
	produktkod			FK ON DELETE CASCADE ON UPDATE CASCADE		not null
	forsaljningspris	försäljningspriset för varje produkt och kundgrupp	DOUBLE(9,3)			null

BILAGA 2/3(3): HUVUDTABELLERNAS DATALEXIKON

Anskaffning	anskaffningskod	unikt, identifierande nummer för varje inköp av produkt(er)	BIGINT	PK	AUTO_INCREMENT	not null
	produktkod			FK ON DELETE CASCADE ON UPDATE CASCADE		not null
	anskaffningspris	inköpspriset per produktenhet, dvs per l, kg, styck, set. Samma enhet som produkten säljs i.	DOUBLE(9,3)			not null
	antal	antal produktenheter, dvs. antal set, liter, kg... som införskaffades	DOUBLE(10,3)		CHECK (antal>0)	not null
	datum	datumet då inköpet gjordes (YYYY-MM-DD)	DATE		DEFAULT CURDATE()	not null
Forsaljning	forsaljningskod	unikt, identifierande nummer för varje försäljning av produkt(er). Varje produkt och kund utgör en enskild försäljning.	BIGINT	PK	AUTO_INCREMENT	not null
	produktkod			FK ON DELETE CASCADE ON UPDATE CASCADE		not null
	kundkod			FK ON DELETE CASCADE ON UPDATE CASCADE		not null
	antal	antal produktenheter, dvs. antal set, liter, kg... som såldes	DOUBLE(10,3)		CHECK (antal>0)	not null
	datum	datumet då försäljningen gjordes (YYYY-MM-DD)	DATE		DEFAULT CURDATE()	not null
Lagersaldo	lagerkod	unikt, identifierande nummer för varje post i tabellen (för varje produkt), kunde fungera som hyllnummer.	INT	PK	AUTO_INCREMENT	not null
	produktkod			FK ON DELETE CASCADE ON UPDATE CASCADE	UNIQUE	not null
	antal	antal/mängd som finns kvar av ifrågavarande produkt	DOUBLE(10,3)			not null
	minimiantal	minimigräns för hur många / hur stor mängd som rekommenderas finnas i lagret av ifrågavarande produkt	SMALLINT			null

BILAGA 3: INLOGGNINGSSYSTEMET



BILAGA 4/1(4): BREV TILL DATAEXPERT

25.03.2013
Mr Arto Peterzens
CIO
Algol Oy

Dear Mr Peterzens,
My name is Pernilla Haglund and I'm working on my thesis for the study programme Information and Media Technology at Arcada. The thesis is a report on a database system I've built for a company. In my report I describe the work I've done and then I will evaluate the result. In the evaluation the data security is an important issue. I would really appreciate it if you would be that kind to give me your opinion on the system's security, based on my following description. The deadline for my thesis is in about two weeks (10.4.2013) which means I would need your answer as soon as possible, but the very latest by 7.4.2013. I'm aware of that it's quite soon and I apologize for that. I write this letter in English but if you prefer, you can answer me in Finnish as well.

The database system consists of a MySQL database and two web applications I've built in PHP 5, HTML, JavaScript, Ajax and CSS. The programming paradigm is procedural. One of the applications is a CMS for the database users working at the company. The other one is a web catalog for the company's customers in which they can check out the product range. The whole system is located on a hosting server (Apache) provided by the Finnish company Netsor. Backup copies are taken every night.

The database
The database is carefully planned, divided into many tables with functional relationships. All primary keys are numeric codes generated by the auto_increment function. The attributes are defined with suitable data types and lengths and have restrictions like null/not null, default, unique and check - depending on what's relevant and needed. All foreign keys have the qualities of "on update cascade" and "on delete cascade" to ensure the database is staying clean. (If some tuple is being deleted, the refering tuple(s) in an

other table won't be necessary anymore.) The database and the CMS have now been in daily use for almost 10 months and have been working properly.

The CMS

The log in page to the CMS is a php-file located on the domain tecmarin.fi in a folder. Both the folder and the file have random names consisting of big letters, small letters and numbers. I've instructed the employees to not give away the link to any unauthorized person. The CMS software files are divided up in several folders, all containing an index.html file with only an irrelevant <p> tag in it.

The log in page is a simple form with one field for username, one for password, one for a randomly generated captcha and a submit button. The username and password fields have a maxlength of 15 characters, autocomplete is set to "off" and the type of the password field is naturally "password". A random token encrypted in XXXX is also generated every time the page is loaded. A session for the token is set at the same time. When the form is submitted, the data is sent to another php-script. The first thing that is checked is if the captcha is right. If not, the user is sent back to the log in page, on which all previous sessions are destroyed and a new captcha and token is generated. Every time the script finds something that's not right, the user is sent back to the log in page by this code:

```
header('Location: ../file_name_of_log_in_page.php');
die();
```

The next thing is to check whether the token submitted from the log in form is the same as in the token session. If not, the user is sent back to the log in page, otherwise the script proceeds. The next thing is that the form field data gets filtered through this function:

```
function makeSafe($var) {
$var = mysql_real_escape_string(trim($var));
return $var;
}
```

After that, the data is checked to not be more than 15 characters long. Next thing is to control that there's exactly one tuple in the database that corresponds to the given username. If everything is fine so far, the username, IP-address and a timestamp is inserted to one of the tables in the database.

Finally the hashed password and salt are checked to be the right ones. If they are, a log-in session as well as a new encrypted token and token session is generated and the user is sent into the main page in the CMS. On that page, the new token is controlled to be right and the login session is checked to exist, with the right variable. All usernames have letters and numbers in them and all passwords should have big letters, small letters and numbers and be at least 8 characters long. I have asked the users to change their passwords once a month, but I'm not sure if they actually do.

There are two different user roles, administrator and regular user. The regular users cannot insert, update or remove any data from the system, just make searches and view the database content. The administrators can insert, update and remove data. But before a remove is done, the CMS asks in a pop up (confirm) window if the user is sure about it and informs about the consequences (e.g. "If you remove this product, all data that's related to it will be removed too"). If the administrator really wants to remove the data and clicks "OK" the data is first copied to another similar table with the prefix "del_", a so called backup table. After that the data is deleted from the main table. In other words, no data can be entirely removed if I don't delete it myself from the del_tables in phpMyAdmin.

Else: all input data is filtered by the makeSafe function (to prevent SQL-injections), all files check that the log in session is set and defined by the right variable, the mysql connection is closed in the end of every file and the user is automatically logged out if no activity has been noticed in 30 minutes. When the user is logged out, all sessions are destroyed.

The web catalog

The security measures in the application are quite the same as in the CMS. Some differences can be found, though. There's a link to the web catalog from the company's website so everyone can find the application. The files in this application are quite few so I have them all in the same folder.

Here, there's no captcha on the log in page to prevent machine input. Instead, the user first has to click on a link that says "Log in". When that is done, a session is created and the form is displayed with field for username, password and a submit button. Just as in the log in system of the CMS, a token is generated when the form is loaded. When the data is submitted, the script checks all the same things. If the token is empty or wrong or if the input data is longer than what's defined in the form's maxlength tag, the mysql-connection is closed and the script dies. If just the username or password is wrong, the session is destroyed and the user is sent back to the log in page where he or she has to click the Log in-link again to get to the form.

Even if the username doesn't exist, the given username, IP-address and a timestamp is saved into the database, which wasn't the case in the CMS.
Otherwise, here in the web catalog, the session id is regenerated on most of the pages. The user is automatically logged out after 20 minutes without activity.

I think this description sums up the data security measures I've taken pretty well. Does the system seem secure to you? Is there anything you can think of that should be added or changed for increased data security? If anything is unclear, please ask me about it by sending me an e-mail (pernillamariahaglund@gmail.com) or call me on my mobile (+358503697447). Thank you.

Best regards,
Pernilla Haglund

BILAGA 5/1(3): SVAR AV DATAEXPERT

4.4.2013

Hei,

Kiitokset selkeästä ja hyvin jäsennellystä Tecmarinin www-sovellusten tietoturvakuvauksesta. Luin dokumentin läpi ja kirjasin muutamia huomioita alle. Toivottavasti näistä on jotain hyötyä lopputyösi kannalta. Kaiken kaikkiaan hienoa työtä, kun oletettavasti koko tutkielma noudattaa tätä samaa linjaa!

Yleistä:

Dokumentti on erittäin selkeä ja systemaattinen esitys sovelluksen keskeisistä toteutustekniikoista (kirjautuminen, tietokanta, yhteyden/session hallinta). Toteutus on tehty selvästikin erittäin huolellisesti, hyvää ohjelmointitapaa sekä käytäntöjä noudattaen ja ottamalla huomioon useita erilaisia käyttötapauksia tietoturvan ja tiedon säilyvyyden näkökulmista. On tunnustettava, että omasta ohjelmointityöstä on jo "jonkin verran" aikaa ja muutamat tässä esitetyt asiat olivat tällä yksityiskohtaisuuden tasolla vieraita. Toivon, että en tästä syystä käsittele asioita liian yleisellä tasolla.

Kommentteja:

- palvelimen/palvelimien ulkoistus: hyvä ja järkevä ratkaisu, jolla laitteiston käyttöjärjestelmän, varusohjelmistojen ja tietoliikenteen tietoturva on ammattilaisen vastuulla, joka huolehtii myös tarvittavista päivityksistä mahdollisiin ja tunnettuihin haavoittuvuuksiin liittyen. Myös varmuuskopiointi on järjestetty.

- tietokanta: paras evidenssi kannan ja sen rakenteen toimivuudesta on tietysti toteamus, että CMS-sovellusta on käytetty päivittäin 10 kuukauden ajan. Kenttien tietotyyppien, pituuksien ja rajoituksien määrittely liittyy ehkä enemmän juuri huolelliseen

ohjelmointityyliin ja -tapaan kuin varsinaiseen järjestelmän tietoturvaan. Myös "on update cascade" ja "on delete cascade" kuvastaa hyvää tapaa, vaikka kaiketikaan tuo "on update cascade" ei olekaan tarpeen, jos ja kun primäärejä avaimia ei päivitetä (numeric codes generated by the auto_increment function)

- CMS/Web Catalog:

BILAGA 5/2(3): SVAR AV DATAEXPERT

* log in -form ja proceduuri ok

* käytössä http -> olisiko mahdollista käyttää https-protokollaa vai lähettääkö "pass-word" salasanan verkon yli salattuna? Riski salasanan paljastumiselle on teoreettinen mutta mahdollista.

* riittävän pitkä ja vahva salasana (8-15 merkkiä), myös numeroita ja pieniä sekä suuria kirjaimia

* käytännössä salasanan vaihto kerran kuussa on loppukäyttäjän näkökulmasta varsin vaativaa, kerran esim. kolmessa kuukaudessa riittää hyvin

* session eheyden varmistaminen on hyvä asia

* käyttäjätunnuksen, IP-osoitteen ja aikaleiman tallennus istuntojen monitoroinnin ja jäljitettävyyden näkökulmasta (myös mahdolliset virhetilanteet) on hyvä ratkaisu

* oletettavasti salasanat eivät ole myöskään kannassa selväkielisinä?

* eri roolit hyvä ratkaisu, tiedon poistaminen varmistettu hienosti (vielä mahdollisuus palauttaa "tuhottu" data on tyylikäs ratkaisu)

* näet todennäköisesti minun pari kirjautumisyritystäni sivulta http://www.tecmarin.fi/wwwLuettelo/login.php

KYS: Aiheutuuko mitään mahdollisia ongelmia siitä, että samalla palvelimella ajetaan sekä www.netsor.fi että www.tecmarin.fi?

C:\Users\AP>nslookup www.tecmarin.fi
Server: kotiboksi.Elisa
Address: 192.168.100.1

BILAGA 5/3(3): SVAR AV DATAEXPERT

Non-authoritative answer:

Name: www.tecmarin.fi

Address: 212.213.88.40

C:\Users\AP>nslookup www.netsor.fi

Server: kotiboksi.Elisa

Address: 192.168.100.1

Non-authoritative answer:

Name: www.netsor.fi

Address: 212.213.88.40

C:\Users\AP>

Terveisin,

Arto

Arto Peterzens

CIO | Algol Oy

+358 40 7723 015 | +358 9 509 9227 | arto.peterzens@algol.fi

Karapellontie 6 | P.O. Box 13 | FI-02611 ESPOO | FINLAND

BILAGA 6/1(4): ANVÄNDARINTERVJUER

FRÅGOR TILL DE ANSTÄLLDA PÅ TECMARIN

1. Tycker du att databasens hanteringsverktyg är logiskt och enkelt att använda? Om inte, vad kunde förbättras?
2. Hur har produktkatalogen påverkat Tecmarins kundrelationer/image?
3. Hur har datasystemet i sin helhet påverkat arbetsrutinerna på Tecmarin?
4. Vad har blivit bättre, eller eventuellt sämre, efter ibruktagandet av datasystemet?
5. Har datasystemet gett mervärde åt Tecmarin?
6. Andra synpunkter?

SVAR

Nina Ahlmark, adminrättigheter

1. Ganska lätta att använda. Vissa problem att ibland hitta pris. När man slår in produkt-koden och skall söka pris så söker den igenom alla ”siffror” som finns i databasen.

2. Positivt, jag har försökt att komma ihåg att berätta åt kunderna att vi håller på med en databas och att det i framtiden gynnar kunderna i och med att de kan själv söka pris och produkter. Ger genast en proffsigare bild av företaget när det finns konkret något att ”ta i”.

3. För tillfället så är det Lotta och jag som mest jobbar med databasen. Jag tror att det tar lite längre för de andra att komma in i rutinerna som har med databasen och göra (har direkt och göra med det att alla är inte så inkomna att använda data överhuvudtaget).

4. Jag personligen ser bara positiva saker med databasen. Det har gjort det lättare att prissätta lagervaror och försnabbar faktureringen.

5. Ja, utan den så har vi egentligen ingenting. Det att vi har ett lager som ingen hittar något i har inget värde för varken oss eller våra kunder. Nu har vi också ett verktyg som berättar oss ifall något tar slut eller ifall vi behöver besluta att göra av med något som länge legat i hyllan. Arbetet med lagret kommer alltid att fortsätta, men nu har vi ett verktyg som underlättar det.

6. Det känns ibland lite jobbigt när man inte använder databasen på en stund så stänger den sig. Att logga in sig tar ju alltid en stund. Jag vet att det är en säkerhetsfaktor men kanske man kan göra något åt det.

Charlotta Åberg-Ohlström, adminrättigheter

1. I och med att hanteringsverktyget utvecklats och förbättrats som en process anser jag att det som det är idag, är logiskt och motsvarar Tecmarins behov. Ifall man skulle vela vidareutveckla verktyget kunde man t.ex. Göra det möjligt att söka på flera sökord på en gång. Det är främst sök-och sorteringsfunktionen som man ifall det i framtiden finns behov, skulle kunna vidareutveckla.

2. Produktkatalogen har ett gett "ansiktsupplyft" för Tecmarins image. I dagens e-värld är det värdefullt att kunna erbjuda kunden möjligheten till en e-produktlista om de så önskar. Tecmarin har också vid flera tillfällen fått positiv feedback t.ex. om hemsidan. Att utveckla ett fungerande datasystem (databas, produktkatalog, hemsida) är att svara på kundens behov, vilket alltid gynnar kundrelationer och förbättrar en företagsimage.

3. Datasystement har påverkat Tecmarins arbetsrutiner mycket. I och med att en stor del av försäljningen baserar sig på offertförfrågningar är det viktigt att priserna är snabbt och lätt tillgängliga. Som försäljare använder man databasen dagligen vilket bidragit till att offertprocessen blivit snabbare och enklare. Det är enkelt att få all basinformation om en produkt (när den är köpt, till vilket pris, antal i lager, vilket pris den sålts för till respektive kundgrupp). I och med datasystemet har man också gjort en klar strategisk segmentering av kunder för att t.ex. få statistik på en kundgrupps köpbeteende.

4. Det har definitivt blivit bättre att hålla prissättningen jämn då man har detta verktyg. Det är betydligt lättare att följa med lagervaror och se vilka produkter som rör sig / icke rör sig. Dessutom underlättar verktyget den årliga inventeringen.

De enda svårigheterna som man stött på har att göra med personalens vanor, att få dem att ändra på sina rutiner och börja använda nya verktyg i sitt arbete. På grund av själva datasystemet har ingenting blivit sämre, bara bättre.

5. Datasystemet har gett ett betydligt mervärde åt Tecmarin. Förr var prissättningen beroende av en person, men idag är det lätt för alla försäljare att prissätta enligt samma mönster och följa med varandras prissättning. Datasystemen sparar arbetstid, vilket innebär att en försäljare kan använda sin arbetstid mer effektivt. Dessutom är underlättar dessa datasystem inskolning av möjliga nya försäljare i framtiden.

6. Vi har varit nöjda med databasen och med hur den fungerar. Det är trevligt att se att något som verkat så gott som omöjligt i början (göra en totalt ny omstrukturering av Tecmarins produktsortiment) lyckats så bra!

Kristina Mähönen, adminrättigheter

1. Det är mycket enkelt att använda

2. Vet inte riktigt. Vi får ju nuförtiden så mycke offert förfrågningar. Image har blivit mera ”proffs”

3. Det går bra att hitta pris

4. Det finns ett ställe var man hittar pris (keskitetysti)

5. Det ser ut mycket mera ”Proffs”

6. Samma som punkt 5

Lars-Erik Åberg, vanliga användarrättigheter

1. Ja, den är logisk och lätt att använda. Man hittar produktgrupperna lätt och får den data man behöver.

2. Den har säkert gett en bättre bild av Tecmarins produkturval åt sådana personer som inte känner så bra till det från tidigare

3. Det underlättar offertgivnings processen. Jag använder det dagligen.

4. Det går snabbare att få fram priser och uppgifter om lagersaldo.

5. Ja det sparar tid.

6. Hela datasystemet, hemsidan med produktkatalogen ger en mera professionell bild av företaget utåt.

Fredrik Ohlström, vanliga användarrättigheter

1. Jag tycker nog att den är logisk. I framtiden kunde kanske vi är göra priskategorierna mer enhetliga så att prislistor blir mer överskådliga.

2. Vi har inte ännu lanserat den till alla kunder. Men för dem som använder den, tror jag nog att det är bra för dem att kunna söka i våra produkter i en lista.

3. Nog har det ju underlättat arbetet. Jag tittar alltid lagerpriser därifrån först när jag offererar. Det kunde man inte göra förr. Det har försnabbat arbetsrutinerna.

4. Det har blivit lättare att hitta pris och hålla en jämn prisnivå. Dåligt är kanske att man inte behöver gå så mycket till lagret mera så man vet inte vad/var där sakerna finns ;-)

5. Joo, nog har det underlättat. Vi får snabbare iväg offerter åt kunder och kan bättre hålla reda på priser och statistik.

6. Om man vill utveckla sku man kunna t.ex. Göra att lisätiedot skulle synas på alla flikar (även vid myyntihinnat)