Fakultet
Datavetenskap

Martin Janota & Sebastian Thernström

Webbaserat Resurshanteringssystem

CTLAMS - Compare Testlabs Asset Management System

Webbased Asset Managament System

CTLAMS - Compare Testlab Asset Management System

Examensarbete 15 hp
IT - designprogrammet - Programvarudesign

Datum: 2011-06-09
Handledare: Thjis Holleboom
Examinator: Donald F. Ross
C2011:09

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Martin Janota & Sebastian Thernström

Webbaserat Resurshanteringssystem

CTLAMS - Compare Testlabs Asset Management System

Webbased Asset Managament System

CTLAMS - Compare Testlab Asset Management System

Examensarbete 15 hp
IT - designprogrammet - Programvarudesign

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Martin Janota

Sebastian Thernström

Godkänd, 2011-06-09

Handledare: Thijs Holleboom

Examinator: Donald F Ross

Innehåll

Figurlista

Kodexempel

Abstract

Compare Testlabs are in grave need of a newer and more advanced asset management system than they currently have installed. After contacting them, they explained their need, demands and their wishes. The solution became realized as a PostgreSQL database with an online interface which is only accessible on their internal network. Since they, Compare Testlabs also want to have a mobile way of identifying their assets we had to consider a solution to fit their needs. The results of the six-month period of development is a easy to use and easy to administrate system with a centralized database running on a virtual server and a homemade solution to their need of an Android application.

Sammanfattning

Compare Testlab är i behov av ett nyare och mer avancerade resurshanteringssystem än det de för närvarande har installerade. Efter att vi hade kontaktat dem, förklarade de sina behov, krav och önskemål. Lösningen blev realiserad som en PostgreSQL-databas med ett användarvänligt gränssnitt som endast är tillgänglig på deras interna nätverk. Eftersom de, Compare Testlab, även vill ha ett mobilt sätt att identifiera sina tillgångar var vi tvungna att frambringa en passande lösning . Resultatet av den sex månader långa utvecklingen är ett lättanvänt och lättadministrerat system med en central databas som körs på en virtuell server och en specialanpassad lösning till den önskade applikationen för Androidtelefoner.

1 Introduktion

Uppdraget var att ta fram ett resurshanteringssystem för Compare Testlab eftersom deras nuvarande system varken har den skalbarheten eller funktionaliteten som krävs av ett system för just denna arbetsuppgift. För tillfället använder sig Compare Testlab av ett excelark för att hantera alla sina resurser och letar efter ett mer avancerat och optimalt sätt att hantera sina resurser på. En mer detaljerad beskrivning av deras befintliga system hittas i kapitel 1.3.1.

1.1 Bakgrund

Compare Testlabs har införskaffat kringutrustning efter behov under sin verksamhetstid. Det handlar om sådant som switchar, tangentbord och stationära datorer etc. Mängden tillgångar har till slut nått en sådan nivå att det har blivit svårt att hålla reda på allting, och det finns behov av ett system för att underlätta hantering av all utrustning.

Det ett sådant system skulle behöva göra är att lagra information om alla föremål på ett sätt så att det går att skilja på dem, vilket pågående projekt de används till för tillfället samt vilken lagerplats de ska ligga på i de tre till fyra olika förvaringsrum som finns tillgängliga när de inte används. Ett annat önskemål som presenterades var att kunna märka de olika föremålen med en streckkod som kopplas till systemet, så att med hjälp av en streckkods scanner skulle kunna får ut information om föremålet på ett smidigt sätt.

Ytterligare uppgifter systemet skulle kunna hjälpa till med kom upp under arbetes gång, så som uträkning av sammanlagda inköpspriset på en mängd föremål, eller deras aktuella värde efter avskrivningar.

När uppgiften pressenterades fanns det inget befintligt system, mer än ett excelark som anställda på Compare Testlabs fyllde manuellt in alla tillgångar och sparade den på ett gemensamt utrymme.

1.2 Compare Testlab

Compare Testlab är en gemensam satsning av Compare, som är en plattform för Karlstadsregionens IT-företag med ett 80-tal företag som representerar ca. 3500 anställda, Karlstads Universitet och Sätterstrand Business park inom ICT branschen. Det Compare Testlab vill erbjuda sina kunder är en öppen och leverantörs oberoende testmiljö för affärs- och kompetensutveckling. Rent principiellt erbjuder Compare Testlab sina kunder en möjlighet att testa, validera samt kvalitetssäkra sina ICT-system. Genom att hålla tät kontakt med Karlstads Universitet kan Compare Testlab erbjuda och

trygga en kompetensförsörjning och kompetensutveckling tack vare den kontinuerliga forskningen och utbildningen med inriktning på test.

1.3 Projektdiskussion

1.3.1 Befintligt system

För närvarande så katalogiserar anställda på Compare Testlabs alla enheter med hjälp av ett ständigt växande exceldokument . Eftersom det ständigt tillkommer nya enheter till företaget så har filen i princip växt bortom kontroll och nu krävs det ett smidigare och mer skalbart system för att hantera den mängden enheter som är befintliga. Dessutom skulle ett centraliserat register underlätta uppdatering och kontroll, till skillnad från exceldokumentet som bara finns på en enskild dator. Ett exceldokument är inte skapat för att vara lätt att söka i eller att sammanställa data till läsbar information vilket i sin tur födde behovet av ett nytt system. Målet är ett system som är både skalbart, enkelt att använda och presenterar data begripligt och i ett sammanhang.

1.3.2 Prototyp

För att sammanfatta uppgiften så skall det skapas en databas i PostgreSQL med ett webbgränssnitt för att katalogisera och lätt kunna lokalisera alla tillgångar samt ett enkelt sätt att identifiera de genom. Identifieringen skall ske genom en streckkod som tilldelas beroende på vilket ID enheten har i databasen. Som ett alternativt tillägg i systemet så har Compare Testlabs begärt en Androidapplikation för att snabbt och smidigt kunna identifiera ett föremål med en streckkod utan att behöva sätta sig ner vid en dator.

1.3.3 Kravspecifikation

Efter att uppdragsgivaren och vi har sammanställt en kravspecifikation som kan ses i bilaga 1 kan det sammanfattas att det systemet som skall skapas måste kommas åt med hjälp av ett webbgränssnitt med login funktion, användaren skall presenteras med ett val mellan om den vill logga in som Administratör eller som gäst. Administratör behörigheter låter en lägga till, ta bort, redigera och ändra i databasen där gästen kan bara söka i databasen. När administratören vill lägga till en tillgång måste den välja om tillgången är en investering eller bara förbrukningsvara (administratören måste skilja mellan en dator och en TP-kabel och liknande). Vid registrering av förbrukningsvaror måste användaren fylla in vad det är för föremål och hur många de är dock skall så skall användaren utelämnar avskrivningsregler, pris, serienummer samt ägande projekt. När användaren däremot skall registrera en investering så som en ny dator eller en switch tvingas den till att fylla in serienummer, inköpsdatum, inköpsansvarig, ägande projekt, typ samt namn. Det mesta fylls in manuellt förutom ägande projekt, inköpsdatum och typ, de valen presenteras med hjälp av drop-down menyer med de

olika valen, detta för att undvika flera olika stavningar på samma projekt och typer samt olika format på datum. Alla tidigare nämnda alternativ är obligatoriska att fylla in.

Vid gästinloggninig som inte kräver något lösenord, tillåts användaren söka på allting i hela databasen men inte redigera någon information. Sökningen avgränsas med en typ av filter (figuren nedan är ett enkelt exempel på hur sökningen kan se ut)

Figur 1 – Sökfälts exempel.

Sökningen kommer att försöka matcha de sökfält som fylls in, med andra ord om användaren inte fyller in någonting i fälten ”Namn”, ”Projekt” men fyller in bara streckkoden så kommer sökningsalgoritmen att endast söka efter alla tillgångar som har matchande streckkod. Denna söknings funktion kommer utökas allt eftersom det tillkommer fler relevanta fält kunden önskar att ha. Resultaten kommer upp i en egen ruta som en simpel lista av element.

1.4 Struktur av uppsatsen

Det första kapitlet beskriver uppdragsgivaren samt grunderna till varför behovet av systemet uppkom samt själva projektets bakgrund. Kommande kapitlet, kapitel 2, ägnas åt en detaljerad beskrivning av implementationen, de alternativa implementationerna samt det slutgiltiga resultatet. I samma kapitel beskrivs även grunderna till varför vissa av valen har gjorts och varför de andra inte kom till. Efterkommande kapitlet, kapitel 3, innehåller en beskrivning ar problemen som uppkom under projektets gång och slutsatser som baseras på dem. Kapitel 4 innehåller alla slutsatser samt en slutlig överblick över systemet och dess uppkomst, så som de olika utvecklingsstadierna som systemet befann sig i och vad de ledde till.

Alla källor som användes under utvecklingen samt framtagandet av uppsatsen är listade i kapitel 6 samt den bifogade originella kravspecifikationen hittas i kapitel 7.

1.5 Sammanfattning

För att summera detta kapitel, Compare Testlab är en gemensam satsning av Karlstads IT-företag för att kunna utveckla en leverantörsoberoende testplattform för sina kunder. Detta innebär att CTL (Compare Testlabs) leder en stor mängd av olika projekt och varje projekt i sin tur kan bestå av subprojekt, som CTL har valt att kalla dem. Alla de projekten och subprojekten kräver en sorts hårdvara och denna hårdvaran skall registreras och hållas koll på. Detta har hittills gjorts med hjälp av ett excelark vilket gör det till ett medium som inte är optimerat för skalbarhet eller underhåll, så vår uppgift är att skapa ett ersättningssystem som är både skalbart, enkelt att underhåll och att använda och erbjuder en mer avancerad funktionalitet som samtidigt är enkel att hantera och underhålla. Enda kraven som lagts på oss av företaget är att systemet skall använda en PostgreSQL databas istället för en MySQL databas, att det skall ha ett webbaserat gränssnitt och som ett litet extraknep vill CTL implementera en skrivare som skriver ut streckkoder som i sin tur skall läsas in av en Android telefon som i sin tur ska kontakta databasen och kontrollera vart föremålet ska befinna sig, vilket projekt det tillhör osv. Alla de olika strukturer och alternativa lösningar för hur systemet skall struktureras upp så att det är både enkelt, snyggt och smidigt att använda. På de mer tekniska delarna kommer beskrivas lite senare i uppsatsen.

2 Prototyp & implementation

2.1 Introduktion

Språken som valdes för att implementera hela systemet i är HTML för hemsidans uppbyggnad, PHP för all logik som användaren inte ser så som alla SQL-frågor och skapelsen av tabellerna, för att underlätta användningen och göra tabellerna dynamiska så skapas de allt eftersom resultatet av en SQL-fråga mot databasen returnerades. För att göra sidorna lite mer dynamiska användes JavaScript, på sidan som presenterar alla tabeller till användaren hämtas alla tabeller på en gång och sedan beroende på vilken flik användaren trycker på visas de olika tabellerna enskilt med hjälp av JavaScript.

Eftersom vi inte hade någon större erfarenhet av de språken så spenderades en stor del av startsträckan på att läsa in grunderna. Dock var inte PostgreSQL en större utmaning eftersom principen och ideologin bakom den var samma som bakom MySQL enda skillnaden var att PostgreSQL är en större variant av MySQL, mer om detta i kapitel 2.2.2 och 2.3.2.
PHP/HTML/JavaScript var inte de första valen som gicks igenom, ett antal andra språk var tvungna att avvisa med tanke på att plattformen som Compare Testlabs valde var Unix baserad och de andra alternativen var bundna till Microsoft plattformar. Vilka val valdes emellan gås igenom enare i kapitlet.

2.2 Alternativa implementeringar

Ett av de första valen var vilket språk som skulle välja, eftersom det fanns många kandidater och många av dem var inte alls självklara vinnare eller förlorare. Som till exempel, i kapitlet 2.2.4 där programmeringsspråket Python läggs under förstorningsglaset. Där har varken PHP eller Python en klar fördel över varandra vilket inte underlättade valet, detta gås igenom mer grundligt i angivna kapitlet.

2.2.1 ASP.NET

ASP.NET är ett webb programmeringsspråk utvecklat av Microsoft och baseras på .NET Framework, samma ramverk som Microsoft Windows arbetar på.

Eftersom vår plattform körde CentOS vilken är en Red Hat Linux variant valdes det att inte använda ett Microsoft specifikt språk, det var inte omöjligt att få de två att fungera tillsammans men det

skulle kräva en stor mäng arbete med en variation av tillägg till Apache och liknande. Detta ledde oss till att använda Apache, vilken är stort överlägsen jämt emot alla andra http hosts med tanke på att den är en av de första och anses ha bidragit mest till WWWs tillväxt (World Wide Web). Eftersom det var enklare att implementera PHP stöd på en Apache server än vad det var att implementera ASP.NET så valdes det senare alternativet bort. Ett annat skäl som bidrog till det är det faktumet att PHP anses samarbeta bäst med databaser, vilket är anledningen till att den berömda blogg-servern WorldPress.com, som hanterar flera tusen blogginlägg om dagen, använder sig av MySQL och PHP för att hantera alla de.

2.2.2 MySQL

Valet av databastypen gjordes åt oss av våra uppdragsgivare. Eftersom de helst ville använda sig av open source produkter och eftersom Sun har blivit uppköpta av Oracle. Uppköpningen innebar att vissa delar av MySQL blivit betaltjänster, och därför ville våra uppdragsgivare använda sig av PostgreSQL. PostgreSQL är en open source variant av MySQL med lite små ändringar men detta ägnas kapitel 2.3.2 åt.

2.2.3 Perl

Perl är ett generellt högnivåspråk som utvecklades ursprungligen, av Larry Wall, som ett skriptspråk för att underlätta bearbetning av rapporter. Språket lånar karaktäristiken från C, sh(Shell Scripting), AWK och sed.

Perl valdes bort mestadels eftersom språket var menat som ett skriptspråk för hantering och manipulering av stora textfiler. Språket glänser som mest när det kommer till CGI scripts (CGI eller Common Gateway Interface är en standard för hur webbservrar kan delegera generationen av webbsidor till fristående program) och hantering av riktigt stora mängder data så som inlägg i en wiki eller liknande (några exempel på hemsidor som använder perl är www.amazon.com, www.craigslist.com, www.bbc.co.uk). Eftersom enorma mängder data inte hanteras i detta system så kändes inte Perl som ett rätt val för denna implementation.

2.2.4 Python

Python är ett högnivå språk som designades för en klar syntax och hög läsbarhet. Språket stödjer flera programmeringsparadigm så som OOP (objekt orienterad programmering), imperativ programmering samt funktionell programmering. Python används oftast som ett skriptspråk men dess användningsområde är inte begränsat till endast det.

Den stora anledningen till att Python valdes bort var att det inte var lika enkelt att få den att samarbeta med HTML som PHP. Samt PHP inbyggda fördelen med databashanteringen och den

utförliga dokumentationen kring den. Det som spelade en väldigt stor roll var även faktumet att vi inte hade arbetat med Python innan men däremot hade vi båda liten, hur som helst ytlig, erfarenhet av PHP tack vare tidigare programmeringsprojekt.

2.3 Utvecklings- & designbeskrivning

Det som fanns på ruta ett var en virtuell installation på en av de lokala servrarna på Compare Testlab (se delkapitel 2.3.5 för en beskrivning av en virtuell server) av operativsystemet CentOS (se delkapitel 2.3.7 för beskrivning av CentOS), med tillhörande tillägg samt en PostgreSQL installation, som var den databas som användes (se delkapitel 2.3.2 för beskrivning av PostgreSQL).

För att arbeta med servern användes en SSH-tunnel, under utvecklingsprocessen (se delkapitel 2.3.6 för beskrivning av SSH-tunnel), mellan den och utvecklingsdatorerna. Det innebär att en viss port öppnas på servern och en SSH server applikation installeras och körs. Det möjliggjorde åtkomst och ändring bland alla filer på servern, samt körning shell-kommandon på den direkt från utvecklingsdatorerna. Något grafiskt gränssnitt användes inte då serverdatorn var svåråtkomligt placerad i ett serverrum och åtagandet att installera en lösning som kunde användas från utvecklingsdatorerna ansågs innebära mer extra arbete än nytta i slutändan, då de ändringar som behövde göras på servern gick utmärkt att göra med shell-kommandon.

På utvecklingsdatorerna användes operativsystemet GNU/Linux i Ubuntu-distributionen, en gratisdistribution som används i väldigt stor utsträckning över hela världen. Anledningen att just Ubuntu användes var först och främst att det är en linuxdistribution som vi har tidigare erfarenhet av och att den är väldigt väl underhållen av utvecklaren och det finns mycket information och support kring det på internet. Det som vid vissa tillfällen ställde till det lite för oss var att Ubuntu i sin tur är baserat på linuxdistributionen Debian, en av de största och äldsta linuxdistributionerna, medan CentOS (operativsystemet på servern, se delkapitel 2.3.7) är baserat på Red Hat. Detta medförde att vissa applikationer som fungerade på utvecklingsdatorerna krävde en hel del anpassning och modifikation för att fungera på serverdatorn. Ett exempel på en sådan applikation är ”qrencode” (se delkapitel 2.3.8) som fick installeras som Debian-version på utvecklingsdatorerna och Red Hat-version på serverdatorn. Funktionaliteten var dock densamma och det gick att lösa utan bestående problem.

Efter att ha etablerat grunden till projektet i form av server och utvecklingsdatorer samt fri åtkomst däremellan inleddes själva utvecklingsarbetet. I och med att någon direkt kravspecifikation inte tillhandahölls av uppdragsgivaren var det första steget att designa den översiktliga strukturen av

systemet samt databasen och hur den information som behövde lagras där skulle representeras. Under det första utvecklingsmötet bestämdes, i samverkan med uppdragsgivaren, att det grafiska gränssnitt som skulle användas till systemet skulle vara webbaserat, och således vara lättillgängligt över nätverket genom en vanlig webbläsare. En webbaserad lösning är även plattformsoberoende då alla webbläsare presenterar en webbsida på mer eller mindre samma sätt, vilket är en fördel på Compare Testlab där de anställda använder sig av både Windows- och Linuxdatorer.

Vidare bestämdes också under det första utvecklingsmötet hur den första prototypen av databasen skulle designas. Denna process innefattade brainstorming från uppdragsgivarens sida där de försökte tänka ut all möjlig information som skulle kunna tänkas lagras i databasen, varpå vi fick strukturera upp den i tabeller på ett så lämpligt sätt som möjligt. Designen för den första prototypen av databasen fokuserade på ett generellt upplägg av tabeller och dess attribut som på ett bra sätt underlättade vid alla de framtida förändringar och utökanden av databasen som självklart förväntades göras. Det innebar i stort sett att ITEM-tabellen skapades, samt vissa komplementtabeller så som TYPE och PROJECT (se delkapitel 2.3.2 för utförlig beskrivning av databasen och dess tabeller). Det utgjorde en väldigt simpel första version av databasen som kunde användas för att testköra de första grundläggande funktionerna på webbsidan, vilken stod näst på tur att påbörjas.

För att ens kunna hosta PHP-sidor på en server krävs att en http-server applikation körs på den, en så kallad webbserver. Detta för att servern skall veta hur den skall behandla de http-begäran som kommer från klientmaskinerna när de kontaktar servern via http-protokollet, förslagsvis genom en webbläsare. I vårt fall blev valet av mjukvara den beprövade Apache-servern (se delkapitel 2.3.9). Eftersom Apache är väldigt väl använt och etablerat, med omfattande dokumentation och support, gick installationen och uppstarten utan några större problem, och den virtuella servern blev en fungerande webbserver vars PHP-filer kunde nås genom webbläsaren på de datorer som befann sig på samma lokala nätverk. De få ändringarna som behövde göras på standardinstallationen av Apacheservern var småsaker som att ställa in så att den tar emot http-begäran från alla ip-adresser på det lokala nätverket, samt vart i filsystemet på servern standardmappen för hämtning av PHP-filerna var placerad. För mer detaljerad beskrivning av inställningarna, se delkapitel 2.3.9.

Nästa steg var att koppla ihop databasprototypen och webbservern så att data från databasen kunde presenteras i ett webbgränssnitt. Detta var i slutändan vad systemet var avsett för, och ansågs vara en för viktig bit för att lämnas till senare i utvecklingen. I den installationen av PHP som fanns på servern följde ett tilläggspaket med som innehöll en mängd PHP-funktioner som används för att direkt kommunicera med en PostgreSQL-databas, det som behövde göras var att etablera en

koppling mellan Apacheprocessen och PostgreSQL-processen (då de båda kördes lokalt på servern) och använda dessa PHP-funktioner på rätt sätt. Att få Apache, som stod för presentationen av allt PHP-material, att kunna komma åt data från PostgreSQL-databasen krävde några inte helt uppenbara inställningsändringar på servern. Kortfattat innefattade problemet att den användaren som Apacheprocessen körs som på servern är den användaren som behöver komma åt databasen när man hämtar data genom en PHP-funktion, vilket kräver att denna användare har läs- och skrivrättigheter i databasen. För mer utförlig beskrivning av problemet och den lösning vi fann, se delkapitel 3.2.2.

När en koppling till databasen från webbservern var etablerad så att data kunde hämtas och presenteras på webbsidan kunde utvecklingen av webbgränssnittet påbörjas. Till en början konstruerades en simpel PHP-sida som presenterade innehållet i den största tabellen i databasen, ITEM, i en vanlig HTML-tabell. Det var så vi valde att börja, för att ha ett första steg i arbetet. För att sedan se till den överliggande designen av webbplatsen och dess olika sidor. Hur det skulle gå till för att skilja på användarklasser med olika behörighetsnivå så som administratörer och vanliga användare och hur alla åtgärder som behövde kunna utföras i databasen var åtkomliga från webbplatsen. Utvecklingen av systemet fortsatte i samma manér, där arbetet gick mellan design och implementation i mindre iterationer. Detta kom sig av kravspecifikationens karaktär, där vi som utvecklare fick välja själva hur upplägget av sidan skulle se ut, varpå vi förde en dialog med uppdragsgivaren där de fick komma med synpunkter och önskemål allt eftersom. För vidare detaljerad beskrivning av webbplatsen, dess sidor samt exempel på den kod som användes, se kapitel 2.3.1.

Figur 2 visar hur kommunikationsvägarna går vid användning av systemet med en klientdator och smartphone. De röda linjerna representerar datoranvändning via webbplatsen som finns på servern, medan den de blåa linjerna visar hur flödet går när systemet används via en smartphone.

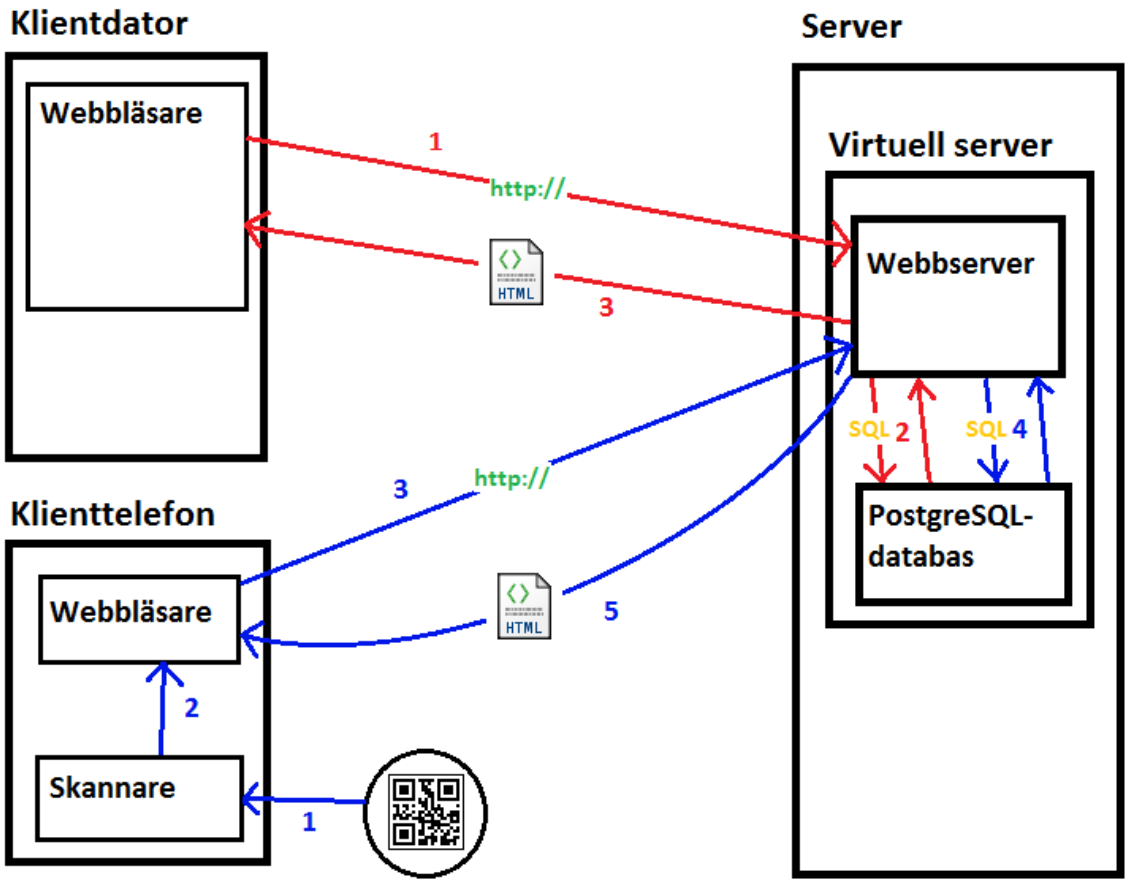
Röd linje, användning via webbläsare i dator:

1. En http-begäran skickas till servern när användaren försöker komma åt serverns adress genom en webbläsare.
2. Webbservern hämtar data från databasen genom en SQL-fråga.
3. En fullständig HTML-sida returneras till klientdatorn, varpå processen kan upprepas.

Blå linje, användning via smartphone:

1. QR-koden på ett föremål skannas in och avkodas genom telefonens kamera med hjälp av en streckkodsskanningsapplikation.

2. Den skannade webbadressen öppnas i telefonens webbläsare.
3. En http-begäran skickas till servern från telefonens webbläsare.
4. Webbservern hämtar data från databasen genom en SQL-fråga.
5. En fullständig HTML-sida returneras till telefonen, och information om föremålet som skannades in visas dess webbläsare.



Figur 2 - Anropsflödesschema över systemet.

Ett av de ursprungliga målen med projektet var att koppla de olika föremålen i databasen till en unik streckkod som kunde fästas på dem för att snabbt kunna hitta information om dem med hjälp av en smartphone. Detta integrerades i systemet i form av en print-knapp på huvudsidan där alla föremål i databasen visas som skriver ut en QR-kod(Quick Response kod, se kapitel 2.3.3 för detaljerad beskrivning) på en streckkodsskrivare som är ansluten till nätverket. För att skapa streckkoden och koda rätt data i den används biblioteket qrencode (se delkapitel 2.3.8 för mer information om qrencode). För att den väsentliga informationen i systemet sedan skulle vara lätt åtkomlig från en smartphone valde vi att konstruera en webbsida visar upp information om ett visst föremål i databasen. Data som kodas i varje föremåls streckkod är en unik adress till just det föremålets variant av denna webbsida. När man använder en enkel streckkodsläsningsapplikation, som finns

gratis till både iPhone- och Androidtelefoner, kan man enkelt navigera till http-adressen där informationen om föremålet presenteras genom telefonens webbläsare (för mer utförlig information om lösningen vi valde för smartphones, se delkapitel 2.3.3).

2.3.1 PHP & HTML

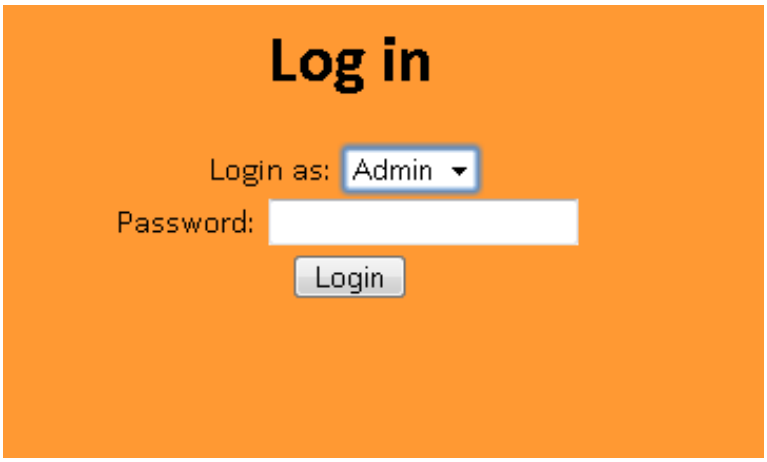
Webbplatsen består av sex webbsidor; loginsidan, huvudsidan, tabellhanteringssidan, objekthanteringssidan, sidan för att lägga till nya objekt och hjälpsidan. Dessutom finns en sida som bara länkas till i QR-koderna, som inte används på webbplatsen, och en PHP-fil som inkluderas av ett flertal av sidorna som innehåller lite funktionalitet. Hela webbplatsen är skriven på engelska på uppdragsgivarens begäran. Här nedan har varje sida en egen rubrik med tillhörande stycke där sidan och dess funktion beskrivs. För mer ingående tekniska detaljer finns även kodstycken med.

Generellt sett används HTML-tabeller för att presentera data som hämtas från databasen och för att strukturera upp de olika menyerna. För att utföra åtgärder så som filtrering, tilläggning av objekt och annan manipulation av databasen så som objektredigering används post-formulär, som är en standardvariant av formulär i HTML. Vi använde get-data i länkadresserna för att de sidor som behövde skulle veta vilka tidigare val användaren hade gjort. Ett exempel är länkarna till objekthanteringssidan som är inbäddade i nästan alla rader i HTML-tabellerna där både objektet på radens typ och identifieringsnummer är med i adressen (se nedanstående kod för exempel).

```
onClick=\"document.location.href='manage_item.php?item=".$values['0']."&table=ITEM';\"
```

Kod 1 –JavaScript för länkning av tabellrader. I länken ovan är ".$values['0']." en PHP-variabel som innehåller ett id-nummer på ett item.

2.3.1.1 Login



Figur 3 – Loginsidan med behörighetsnivån ”admin” förvald och lösenordsfältet synligt

För att skilja på olika behörighetsnivåer av användarna av webbplatsen bestämde vi oss för att använda en simpel loginsida skriven i PHP. Anledningen till att vi inte försökte implementera något

säkrare alternativ var att servern redan ligger skyddad bakom brandväggar på det lokala nätverket. Den lösning vi valde går ut på att man använder sig av en så kallad sessionsvariabel i PHP-koden som anger behörighetsnivån som sedan skickas vidare mellan de olika sidorna på webbplatsen. Sidorna kontrollerar sedan värdet på sessionsvariabeln när de anropas av klienten och avgör om användaren har behörighet till sidans innehåll eller inte därefter.

Så här ser det ut när sessionsvariabeln definieras:

```
$_SESSION['phplogin'] = false;
```

Kod 2 - Definition av sessionsvariabeln.

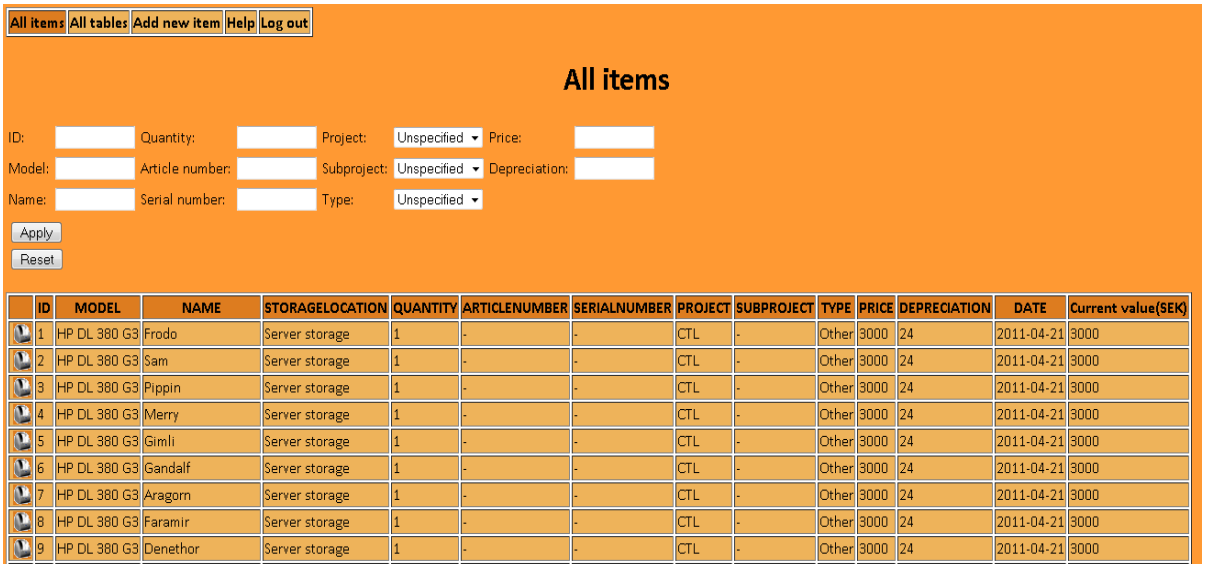
Så här kan det se ut när sessionsvariabeln kontrolleras:

```
if(!$_SESSION['phplogin'])
```

Kod 3 - Kontrollen om sessions variabel är sann.

För att få ett lite snyggare upplägg på loginsidan valde vi att använda oss av JavaScript (se delkapitel 2.3.4 för en översiktlig beskrivning av JavaScript) för att dölja och visa lösenordsfältet. Detta beror på att en gäst inte behöver ange något lösenord för att gå vidare till huvudsidan, medan en administratör gör det. Vi valde att inte utveckla något användarregister eller liknande system för att varje administratör skulle kunna ha en egen personlig inloggning. Detta på grund av att uppdragsgivaren inte hade något önskemål om det och att systemet var avsett för lokalt bruk där mer än två självförklarande användarklasser ansågs överflödigt.

2.3.1.2 Huvudsidan



	ID	MODEL	NAME	STORAGELOCATION	QUANTITY	ARTICLENUMBER	SERIALNUMBER	PROJECT	SUBPROJECT	TYPE	PRICE	DEPRECIATION	DATE	Current value(SEK)
	1	HP DL 380 G3	Frodo	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	2	HP DL 380 G3	Sam	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	3	HP DL 380 G3	Pippin	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	4	HP DL 380 G3	Merry	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	5	HP DL 380 G3	Gimli	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	6	HP DL 380 G3	Gandalf	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	7	HP DL 380 G3	Aragorn	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	8	HP DL 380 G3	Faramir	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	9	HP DL 380 G3	Denethor	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000

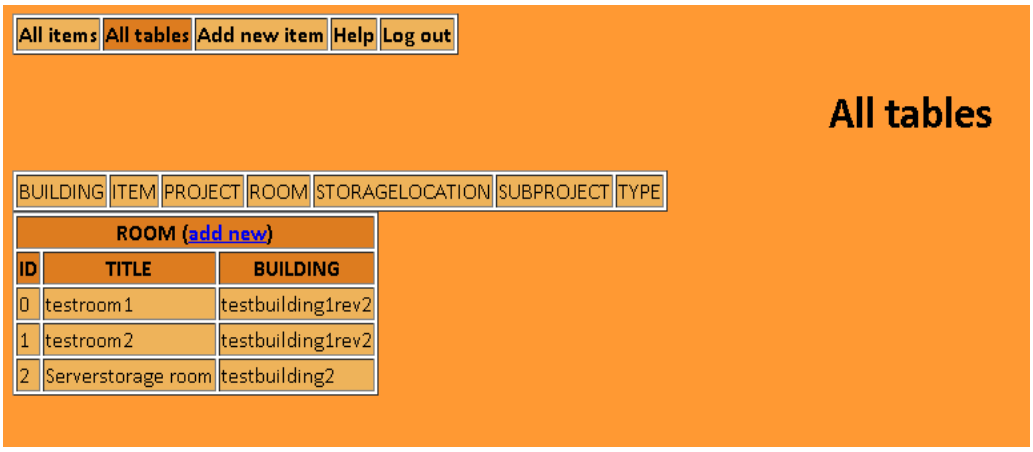
Figur 4 – Huvudsidan, med ofiltrerad lista över föremålen i databasen

Huvudsidan är den enda sidan på webbplatsen som är åtkomlig för alla användare oavsett behörighetsnivå. Vi valde i stället att dölja vissa länkar och funktioner för användare med gästinloggning medan de är synliga och går att använda som administratör. Syftet med huvudsidan var att presentera en lista över den mest väsentliga delen av databasen på ett sätt som var åtkomligt för alla. Vi valde därför att göra en PHP-sida som hämtade den viktigaste informationen från de olika tabellerna i databasen och presenterade den i en HTML-tabell. För att förbättra överskådligheten av sidan lade vi även till ett filtreringsformulär där användaren kan filtrera föremålen i listan efter värdena på deras olika attribut. Hela denna del av huvudsidan är åtkomlig för både administratör- och gästanvändare. Det endast en administratör kommer åt är länkarna till alla andra sidor (se Figur 6 nedan), uskriftsikonen på varje rad i tabellen för att skriva ut föremålets QR-kod (se Figur 5 nedan) och länkarna som är inbäddade i varje rad i tabellen som går direkt till föremålet på radens redigeringssida. Att dölja dessa funktioner gjordes genom en enkel if-sats i PHP-koden där värdet på sessionsvariabeln kontrollerades, varpå stycket hoppades över om användaren var inloggad som gäst. För att tabellen skulle förbli läsbar även vid presentation av stora mängder data använde vi oss av JavaScript för att belysa den rad i tabellen som muspekaren fördes över.

Figur 5 - Ikonen som används för utskrift av streckkod.

Figur 6 - Navigeringsrad med länkar till alla sidor på webbplatsen

2.3.1.3 Tabellhantering



Figur 7 – Sidan där alla tabeller och deras innehåll går att se

Sidan för tabellhantering är bara tillgänglig för administratören. För att inte tillåta användare som inte var inloggade som administratör att öppna sidan valde vi att inleda PHP-koden med en kontroll av sessionsvariabeln vars värde indikerar behörighetsnivån hos användaren. Om sessionsvariabeln påvisar att användaren är en gäst avslutas laddningen av sidan och man omdirigeras omedelbart till loginsidan. Koden för denna funktion såg ut så här:

```
if(!$_SESSION['phplogin']){
    header('Location: login.php');
    exit;
}
```

Kod 4 - Kontroll om användaren är inloggad, om inte skickas de tillbaka till loginsidan.

När väl användarens behörighet var bekräftad hämtades data från samtliga tabeller i databasen för att presenteras. För att göra det på ett överskådligt sätt valde vi att dölja alla tabellerna när sidan laddas, och sedan får användaren trycka på namnet till den tabell som önskas ses varpå den visas med hjälp av en JavaScript-funktion. Vill användaren se en annan tabell i stället är det bara att trycka på dens namn i stället så döljs den föregående tabellen och den nya tar dess plats på sidan. JavaScript-funktionen som användes för att dölja och visa tabellerna ser ut så här:

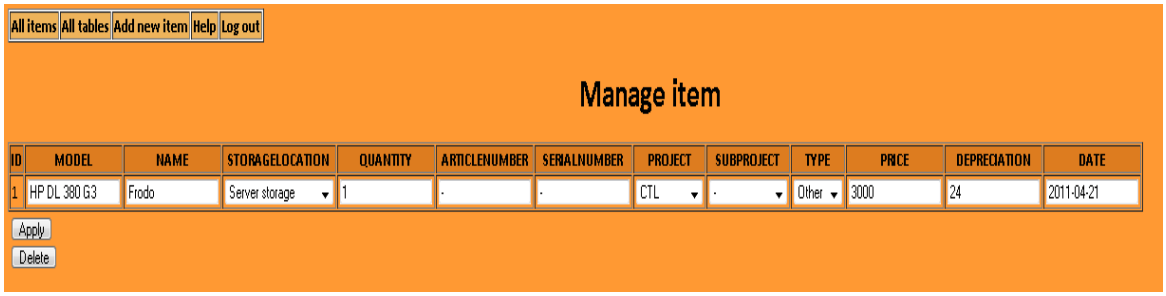
```
<script type="text/javascript">
//Variables to store data which decides the behaviour of this function
var state = 'none';
var currShow = 'none';
    //Function to toggle visability for the tables
    function showhide(layer_ref) {
        //if layer ref == currShow hide layer ref, set currshow=0 then break
        if (currShow == layer_ref) {
            state = 'none';
            currShow = 'none';
        }
        //if layer ref != currshow hide currshow, show layer ref, set currshow=layer ref then break
        else if (currShow != 'none') {
            state = 'none';
            document.getElementById(currShow).style.display = state;
            state = 'block';
            currShow = layer_ref;
        }
        else {
            state = 'block';
            currShow = layer_ref;
        }
        document.getElementById(layer_ref).style.display = state;
    }
</script>
```

Kod 5 - JavaScript för uppvisning av de olika tabellerna.

Precis som på huvudsidan är alla rader i tabellerna länkade till objekthanteringssidan för föremålet på den raden om behörighetsnivån hos användaren är indikerat som administratör av sessionsvariabeln. Eftersom en användare som öppnar sidan under behörighetsnivån gäst omdirigeras till en annan sida direkt är detta alltid fallet.

För att lägga till nya objekt i tabellerna trycker användaren bara på add new-länken på titelraden av HTML-tabellen med tabellens innehåll (se Figur 7). Man omdirigeras då till sidan för att lägga till objekt i den valda tabellen.

2.3.1.4 Objekthantering



Figur 8 – Sidan där objekt från de olika tabellerna kan hanteras

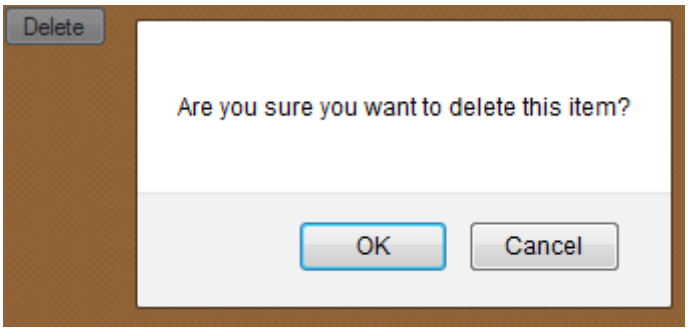
På sidan för att hantera objekt kan attribut hos ett föremål i någon tabell i databasen redigeras direkt från webbsidan. Detta genom att objektets alla attributnamn samt värden laddas när sidan öppnas, varpå användaren kan ändra på värdena och sedan spara ändringarna. Sparningen sker genom att värden användaren angett i fälten plockas ut och skickas till databasen i form av en SQL-fråga med

hjälp av en av de PHP-funktioner som fungerar direkt med PostgreSQL (se exempel på hur SQL-frågan kan se ut nedan).

```
UPDATE assetmanagement."ITEM" SET "MODEL" = 'HP DL 380 G3', "NAME" = 'Merry',
 "STORAGELOCATION" = '3', "QUANTITY" = '1', "ARTICLENUMBER" = '-', "SERIALNUMBER" = '-',
 "PROJECT" = '0', "SUBPROJECT" = '4', "TYPE" = '0', "PRICE" = '3000', "DEPRECIATION" = '24',
 "DATE" = '2011-04-21' WHERE "ID" = 4
```

Kod 6 - Exempel på en SQL-fråga för att uppdatera ett objekt i databasen.

Ytterligare en funktion som endast är tillgänglig på denna sida är att ta bort objekt ur databasen, vilket görs genom att trycka på delete-knappen på sidan och sedan bekräfta borttagningsåtgärden i den pop-up dialog som visas (se Figur 9 här nedan för en bild på delete-knappen och pop-up fönstret).



Figur 9 - Delete-knappen och pop-up frågan som måste godkännas för borttagning

Pop-up fönstret är skrivet med JavaScript (se exempelkod nedan för skriptets kod). Om användaren klickar på OK-knappen genomförs borttagningen och en SQL-fråga skickas från webbservern till databasen (ett exempel på en sådan SQL-fråga finns i exempelkoden nedan). Skulle användaren i stället ångra sig och klicka på Cancel-knappen avbryts åtgärden och man återgår till hanteringssidan utan att någon SQL-fråga skickas.

JavaScriptet för delete-popup:

```
<script type="text/javascript">
    function confSubmit(form) {
        if (confirm("Are you sure you want to delete this item?"))
        {
            form.submit();
        }
    }
</script>
```

Kod 7 - JavaScript för uppvisning av en popup ruta vid borttagning från databasen.

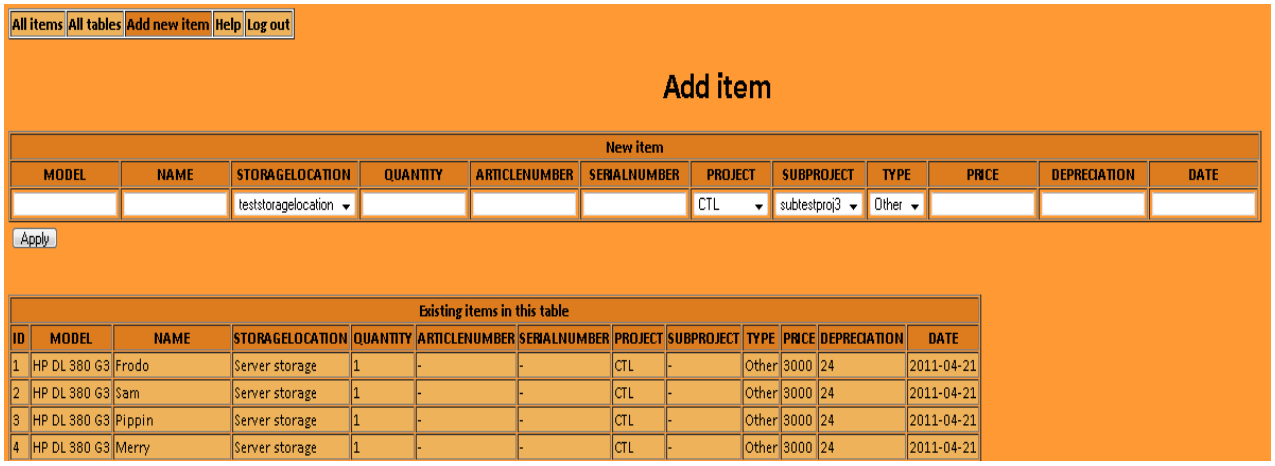
SQL-query för borttagning:

```
DELETE FROM assetmanagement."ITEM" WHERE "ID" = "3"
```

Kod 8 – SQL-fråga för borttagning av ett objekt ur databasen.

Precis som på tabellhanteringssidan så omdirigeras en användare med otillräcklig behörighetsnivå till loginsidan, dvs. om sessionsvariabeln för behörighetsnivån indikerar att användaren är en gäst.

2.3.1.5 Lägg till objekt



Figur 10 – Sidan för att lägga till ett objekt i någon av de olika tabellerna

Sidan för att lägga till nya objekt i databasen är åtkomlig endast från tabellhanteringssidan (se delkapitel 2.3.1.3 för beskrivning). Sidan ändrar utseende beroende på vilken tabell användaren begär att lägga till ett objekt i, på grund utav att objekten i de olika tabellerna har olika attribut (se Figur 11 för två exempel på hur sidan skiljer sig beroende på tabellval).

Figur 11 - Två olika varianter av sidan för att lägga till objekt i databasen

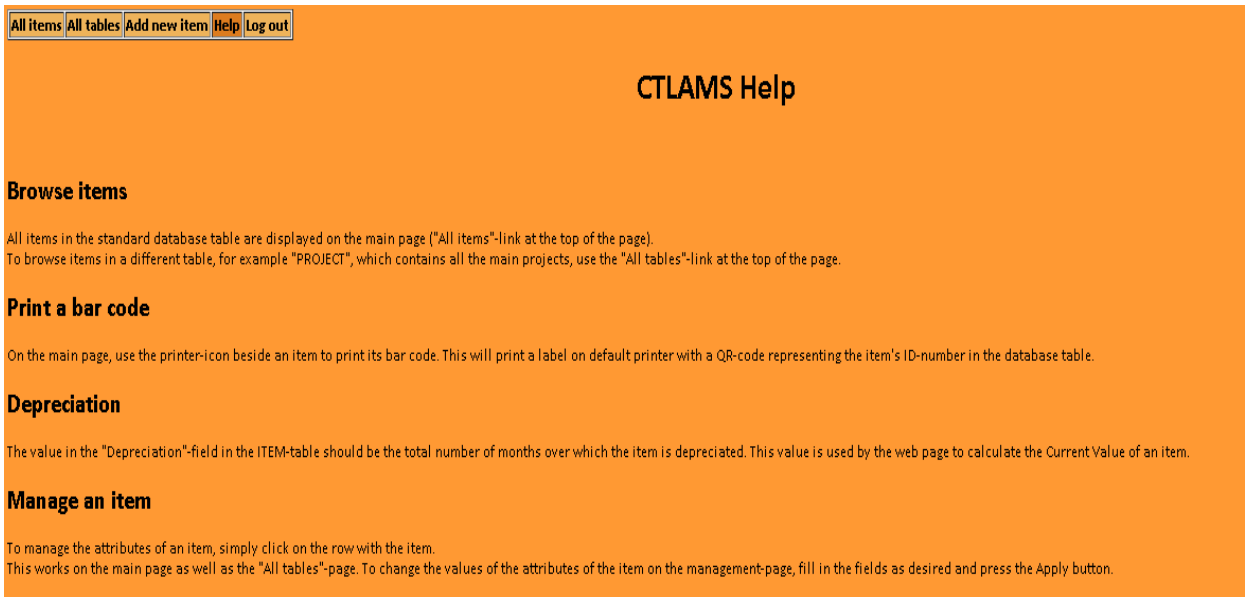
När användaren anser sig färdig med ifyllningen av fälten på sidan läggs objektet till genom att trycka på Add-knappen. En SQL-fråga konstrueras av PHP-sidan utifrån de användarangivna värdena i fälten och skickas till databasen (se exempel på hur en sådan SQL-fråga kan se ut nedan).

```
INSERT INTO assetmanagement."ITEM" VALUES ('21', 'somemodel', 'somename',
'0', null, null, null, '0', '1', '0', '10000', '24', '2011-05-17')
```

Kod 9 - SQL-fråga för att lägga till ett objekt i databasen.

Precis som på objekthanteringssidan så omdirigeras en användare med otillräcklig behörighetsnivå till loginsidan, dvs. om sessionsvariabeln för behörighetsnivån indikerar att användaren är en gäst.

2.3.1.6 Hjälpsidan



Figur 12 – Hjälpsidan där webbplatsens funktioner beskrivs kortfattat 2.3.2 PostgreSQL

Hjälpsidans syfte är som namnet indikerar att hjälpa användare som är osäkra på hur man använder vissa funktioner på webbplatsen. Sidan består av en kort beskrivning av varje webbsida på webbplatsen och en extra fördjupning på vissa funktioner. Nedan finns några exempel på hjälptext som finns att hitta på hjälpsidan.

Delete an item

To delete an item, go to the page where the attributes of the item can be managed (See the "Manage an item"-paragraph) and click the Delete-button.
The item will be permanently deleted from the database when the "Okay"-button is clicked on the popup-window that appears.

Manage an item

To manage the attributes of an item, simply click on the row with the item.
This works on the main page as well as the "All tables"-page. To change the values of the attributes of the item on the management-page, fill in the fields as desired and press the Apply button.

2.3.1.7 Sidan för smartphone

För att komma åt information från databasen från en vanlig smartphone använde vi oss av en simpel PHP-sida som presenterar information om ett objekt i en HTML-tabell. Ett antagande är alltså att användaren har en smartphone med kamera, en applikation för att läsa QR-kod och en webbläsare. Detta ansågs av både utvecklare och kund som en bättre lösning än att utveckla en ny applikation från grunden som alla som vill använda systemet genom smartphones hade varit tvungna att installera, som var den ursprungliga idén. För en bild på hur en typisk kodläsningsapplikation kan se ut, samt hur vår webbsida kan se ut på en iPhone se Figur 13 nedan.

Figur 13 – Skärmdump på smartphoneanvändning av systemet, till vänster en simpel qrkodläsningsapplikation och till höger webbsidan adressen i QR-koden hänvisar till.

Sidan tar emot en get-variabel direkt från adressen och hämtar utefter dens värde information om ett objekt i databasen och presenterar det. Dessa länkar är vad som är kodat i QR-koderna på streckkoderna som sitter på de olika fysiska föremålen ute på förvaringshyllorna, och tanken är att man med en enkel inskanning av denna kod skall kunna få information om föremålet direkt i handen. En typisk sådan kod kan exempelvis se ut så här:

- http://ctlams/item.PHP?item="4"

Vilket skulle presentera information om föremålet med ID 4.

2.3.1.8 Samlingsfilen för funktioner

Vissa funktioner återanvänds på flera olika sidor och kunde med fördel till slut läggas i en enskild fil som sedan inkluderas i de andra sidorna för att deras definition inte skall behöva skrivas på många olika ställen. Framförallt handlade det i vårt fall om långa SQL-frågor som användes på flera sidor och ändå inte är läsbara för någon som läser koden. Vi valde således att plocka ut några funktioner och SQL-frågor och lägga dem i en samlingsfil för funktioner.

Kortfattat fungerar det så att man i en PHP-fil skriver tillexempel:

```
include 'sql_queries.php';
```

Kod 10 - Kodexempel för inkludering av den PHP filen med alla gemensamma funktioner.

All den text som finns i filen SQL_queries.PHP inkluderas då precis där i den aktuella PHP-filen, så länge som SQL_queries.PHP existerar och filerna ligger i samma mapp i filsystemet.

Några av de funktioner som finns i samlingsfilen för funktioner är:

- Navigeringsmenyn för webbplatsen som ligger högst upp på varje sida (se Figur 6)
- Etableringen av koppling mellan webbservern och databasen;

```
$db_handle = pg_connect("dbname=<database> host=<address> user=<username>");
```

Kod 11 - Kod för anslutning till databasen.

- HTML-taggen <STYLE> för att ställa in webbplatsens färgschema, vilket för tillfället är inställt på Compares typiska orange-bruna färger som används på deras officiella webbplats.

```
<STYLE>
.normal { background-color: #eeb35a }
.highlight { background-color: #dd7c1f }
.background { background-color: #FF9933 }
</style>
```

Kod 12 - Styletaggen i HTML.

2.3.2 PostgreSQL

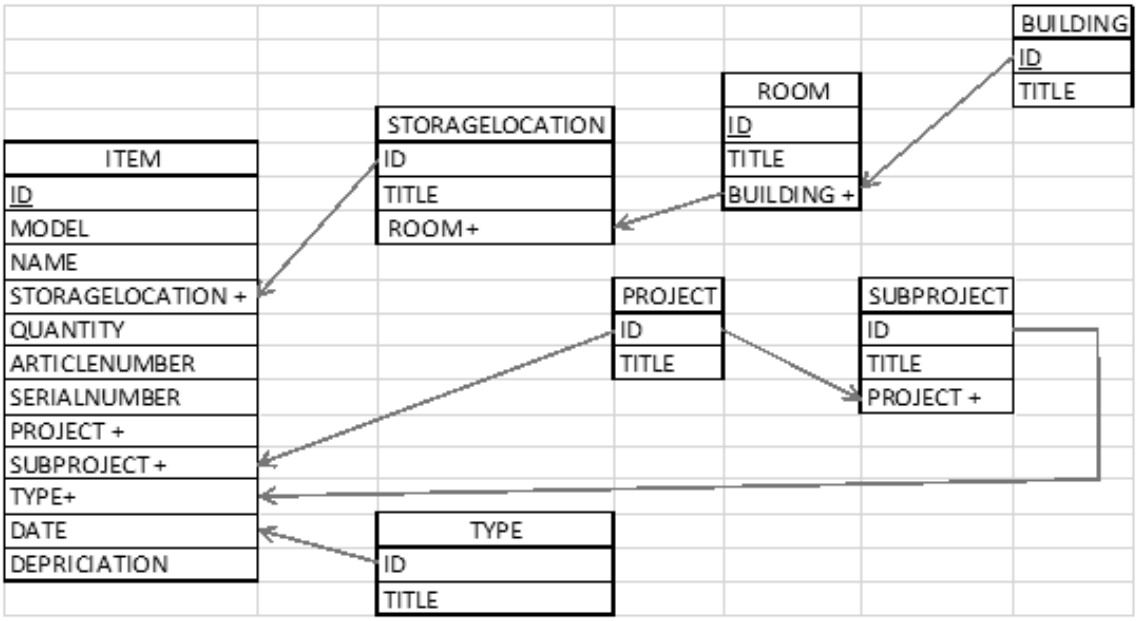
Som tidigare nämnt är PostgreSQL en open source version av MySQL som är en typ av relationsdatabas. Efter att MySQL AB köptes upp av Oracle föredrog Compare Testlab att använda sig av en open source databas.

På syntax basen är skillnaden emellan MySQL och PostgreSQL väldigt minimal vilket innebar att användningen av PostgreSQL inte var ett större problem. De första stegen som togs vid skapelsen av databasen var att försöka strukturera upp vilka tabeller som skulle tas med och vilka kolumner skulle vara mest relevanta i sammanhanget.

Varför PostgreSQL valdes över MySQL är ett val som gjordes av Compare Testlab tidigt i projektet och anledningen är att de vill allra helst använda sig av open source produkter. Vi en närmare titt på skillnaderna emellan MySQL och PostgreSQL anmärktes det att PostgreSQL är en mycket mer kraftfull databas och till skillnad från MySQL erbjuder den en mycket mer flexibel miljö. Eftersom databasen kommer att underhållas av erfarna programmerare sågs det inte problematiskt att PostgreSQL är även mer avancerad än MySQL. Majoriteten av arbetet som kommer att utföras på databasen är så kallade read only-operationer, operationer som endast hämtar data ur databasen och inte ändrar dess innehåll till exempel: tar bort, ändrar eller lägger till. En annan väldigt stor fördel som PostgreSQL har mot MySQL är att den är open source vilket innebär att den utvecklas vidare av sina användare vilket innebär att om administratören stöter på ett problem kan lösningen finnas på internet eller även kunna lösas inom kort, istället för att överlåta problemet ligga hos ett ägande företag som i sin tur tar en stund att släppa små uppdateringar som innehåller fler lagningar eftersom det aldrig är lönsamt att släppa endast en liten lagning. Databasens, PostgreSQLs, stöd för olika plattformar är också en variabel som väger tungt i detta sammanhang, eftersom MySQL är ursprungligen en Windows applikation och servern som skall hålla i databasen är en CentOS-server ger PostgreSQL ett klart övertag.

2.3.2.1 Detaljerad översikt av databasen

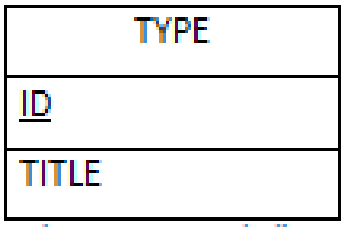
För att etablera databasen användes det PostgreSQL-anpassade administrativa programmet pgAdmin, som gör det möjligt att genom ett grafiskt gränssnitt konfigurera databasen. En översiktlig bild över databasen ger en lite inblick i hur det hela ser ut, se Figur 9.



Figur 14 - Databasstrukturen (Varje relation i databasen är ett 1->N relation.)

Figur 14 representerar en helhetsbild över databasen där de olika tabeller och deras relationer visa, ITEM tabellen håller i den samlade informationen från alla de underliggande tabellerna. STORAGELOCATION, ROOM och BUILDING tabellerna är de tabellerna som ansvarar för de olika lagringsutrymmen så som hyllor/lådor, rum respektive byggnader. Tabellerna PROJECT och SUBPROJECT håller i de olika projekten som hålls på Compare Testlab. Den sista tabellen, TYPE, är den tabellen som används för att hålla koll på alla olika typer på föremålen/resurser som matas in i systemet.

Figuren nedan visar hur TYPE tabellen ser ut i databasen. Tabellen innehåller två fält. Det första fältet står för ett löpnummer ID som är används för att lokalisera typen av ett föremål samt för att identifiera det i en relation. Det andra fältet är namnet på typen så som: ”server”, ”switch” osv.

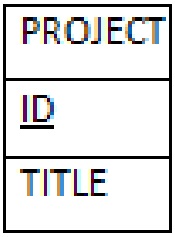


Figur 15– TYPE tabellen.

Tabellen i Figur 16 föreställer hur SUBPROJECT tabellen ser ut, denna tabell innehåller alla de subprojekt som för tillfället ligger aktiva hos Compare Testlab samt vilka projekt de tillhör. ID-fältet och TITLE fältet är tekniskt sett det samma som i TYPE tabellen ovan med skillnaden att de inte är relaterade på något sätt. Enda fältet som skiljer denna tabell är PROJECT fältet som samtidigt är en främmande nyckel som pekar emot ett fält i PROJECT tabellen som beskrivs längre ner i kapitlet.

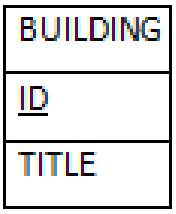
Figur 16– SUBPROJECT tabellen.

PROJEKT tabellen kan sen i figuren nedan, denna tabell innehåller endast två fält och ser principiellt ut som TYPE tabellen fast med annorlunda värden. ID-fältet är det fältet som SUBPROJECT tabellens PROJECT fält refererar till som en främmande nyckel.



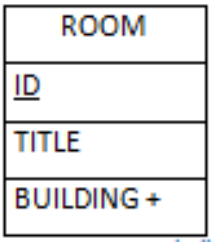
Figur 17 – PROJECT tabellen.

Eftersom Compare Testlab har ett flertal lagringsutrymmen i olika byggnader krävs det att de kan katalogisera de. I Figur 18 finns det en illustration av BUILDING tabellen. Denna tabell innehåller namnet på de olika byggnader samt ett ID nummer så som PROJECT och TYPE tabellerna gör.



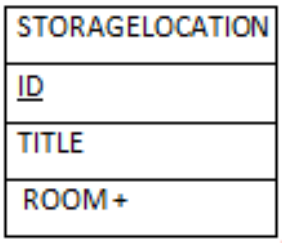
Figur 18- BUILDING tabellen.

ROOM tabellen är den tabell som håller i alla de lagringsutrymmen som ligger i olika byggnader. Tabellen refererar till de olika byggnaderna med hjälp av en främmande nyckel som ligger i BUILDING fältet. De tidigare fälten är samma som i alla de tidigare tabellerna, ID fältet håller i en numerisk variabel som är tupelns unika identifikation samt TITLE fältet som är ett textat namn på "föremålet".



Figur 19- ROOM tabellen.

Eftersom varje byggnad (BUILDING) har ett flertal rum (ROOM) och varje rum har ett antal hyllor eller andra förvaringsutrymmen valdes det att skapa en egen tabell för att hantera de. Som alla tabeller i databasen har även denna ett ID och ett TITLE fält som håller i ett identifikationsnummer samt ett textat namn men tabellen håller även i en främmande nyckel som pekar på vilket rum förvaringsutrymmet befinner sig i. Detta görs med hjälp av ett numeriskt fält.

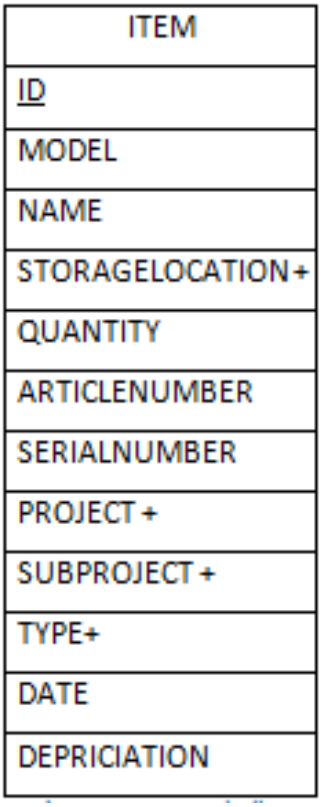


Figur 20 – STORAGELOCATION tabellen.

Tabellen i Figur 21 är den största tabellen och den innehåller alla förebål (ITEMs) i hela databasen. ITEM tabellen har så som alla andra tabeller i databasen ett ID fält för att systemet lätt skall kunna identifiera varje tupel, Däremot finns inte TITLE fältet med, istället döptes fältet med det textade namnet till NAME. Tabellen innehåller referenser till alla de andra tabellerna genom främmandenycklar som i sin tur refererar till ID fälten i respektive tabeller (STORAGELOCATION, PROJECT, SUBPROJECT, TYPE). Fältet QUANTITY innehåller ett värde endast om föremålet inte är en förbrukningsvara dvs. en förbrukningsvara är en vara som oftast finns i större mängder så som

skruvar, kablar eller toner patroner till skrivare. QUANTITY är ett numeriskt värde som representerar antalet av ett givet föremål som just för tillfäller befinner sig i förvar. ARTICKLENUMBER och SERIALNUMBER är de olika serie och artikel numren som datorer och andra elektroniska komponent har, detta gäller dock inte förbrukningsvaror eftersom användaren inte är intresserad av vilket artikel

eller serienummer en tummskruv eller en Cat6 (TP-kabel) har. Eftersom Compare Testlab gärna vill hålla koll på avskrivningen och nuvarande värdet på alla sina tillhörigheter skapades det ett fält för att kontrollera hur många månader föremålet avskrivs under. Detta fält innehåller ett numeriskt värde som motsvarar antalet månader föremålet avskrivs under vilket i sin tur används för att räkna ut det nuvarande värdet.



Figur 21 – ITEM tabellen.

2.3.3 Androidapplikation & QR-kod

På grund av tidsbrist tvingades framtagandet av Androidapplikationen för en smartphone att sänkas i prioritet. Enligt Compare Testlabs önskemål skulle applikationen scanna in en streckkod och returnera viktig information om föremålet användaren hade i handen. Detta innebär att mobilen måste skapa en uppkoppling mot databasen och skicka en SQL-fråga som i sin tur interpretteras och behandlas av databasen. Databasen skall returnera relevant information som skall pressenteras av

applikationen på ett enkelt sätt. Detta är inte allt, för att applikation skulle ha någonting att scanna in skapades det även, i början, en streckkod liknande den som kan ses i Figur 22.

Figur 22 - EAN-13 sträckkod exempel.

För detta användes EAN standarden. EAN står för European Article Number och är en standard som används över hela världen för att. Numera har denna standard döpts om till GS1-nummer eftersom företaget bytt namn efter att Amerikanska UCC och den Kanadensiska motsvarigheten gick ihop med EAN. Den mest använda standarden heter EAN-13 numera GTIN-13 (Global Trade Item Information). Som namnet antyder består den utav 13 siffror, en del av siffrorna är statiska beroende på vilket land man kommer ifrån samt den 13:e siffran som är en kontrollsiffra. Kontrollsiffran används för att kontrollera att koden är korrekt. Bland de icke statiska delar i koden finns det utrymme för företagets prefix samt ett antal siffror för lokal användning, som charkvaror har sin vikt och kilopris inkodat i GTIN-13 koden för att enkelt kunna beräkna priset i kassan.

Denna standard var en som skulle kunna möjliggöra för Compare Testlab att använda sina koder utanför företaget men implementeringen av denna standard visade sig vara för komplicerad av ett flertal anledningar. En av anledningarna var att GTIN-13 endast kan innehålla siffror vilket innebar att ingen riktigt relevant information kunde kodas in i dem. Ett annat problem var att koderna skulle bli alldeles för långa för att räknas in i standarden detta innebar att idén övergavs för en mer användbar, QR-kod.

QR-kod (Quick Response) visade sig vara en mycket smidigare och enklare lösning på applikationen. Denna lösning innebar att allt som behövdes var en enkel scanner som användaren kan ladda ner för just sin smartphone och sedan endast scanna in den koden som skrevs ut. QR standarden låter användaren skapa en grafisk representation av både siffror och tecken, som kan ses på Figur 23.

Figur 23 - Ett exempel på QR-kod (innehåll:" I denna standard kan användaren skapa en grafisk motsvarighet till både ett antal siffror(117256276541) samt text och länkar (http://www.qrstuff.com/)").

QR är en tvådimensionell kod som utvecklades av Toyota för att enkelt hålla koll på alla bildelar inom företaget. Målet med koden var att användaren skall enkelt och snabbt kunna komma åt data som låg i den, som Denso Wave själva uttryckte "Code read easily for the reader". Kodens tvådimensionella egenskaper innebär att den innehåller värdefull information både på höjden och bredden till skillnad från en vanlig streckkod som innehåller data endast på bredden. Koden är även väldigt stryktålig, detta innebär att koden kan läsas av även när den är nedsmutsad eller i sämre skick, som kan ses på Figur 24. För en vidare förklaring hur själva kodningen sker i systemet se kapitel 2.3.8

Figur 24 - QR kod är väldigt tåligt när det gäller läsbarhet.

Detta vägde tungt i valet eftersom alla enheter som kommer tilldelas en sådan kod kommer också att hanteras mycket dagligen vilket innebär risken av koder som skadas eller smutsas ner.

Varje QR-etikett innehåller en länk till en PHP sida. På sidan hämtas all relevant information om föremålet och visas upp i form av en tabell, i adressen till sidan står ID till det föremålet som användaren scannar in, till exempel: "httpd://ctlams/item.PHP?ID=1". Beroende på vilket ID som är

medskickat i länken visas det olika information, exempelfallet returnerar en hemsida med all relevant information om föremålet i databasen som i ITEM tabellen har ID 1.

2.3.4 JavaScript

JavaScript är ett script språk som oftast körs på klientsidan det vill säga i webbläsaren. Skriptspråket är objektorienterat och utvecklats av Netscape. Netscape är det företaget som utvecklade den första grafiska webbläsaren Mosaic. Vanligtvis inbäddas JavaScript i HTML koden vilket görs även i detta fall. Scriptet låter skaparna utföra en stor del av kontroller redan hos användaren innan HTML formuläret skickas in, till exempel kontrollera om olika fält är ifyllda eller korrekt ifyllda så som de obligatoriska fälten hos vissa hemsidor. Detta hjälper att avlasta servern och skapa mycket dynamiska sidor.

JavaScript lämpar sig allra bäst för hantering av små händelser som inträffar lokalt i användarens webbläsare. Eftersom JavaScript körs i webbläsaren så arbetar det mot DOM (Document Object Model). Oftast används JavaScript för att tillexempel visa upp pop-ups eller ändra i hemsidan beroende på vad användaren gör lokalt i sin webbläsare. En annan fördel med JavaScript är att den, till skillnad från serverbaserade språk som PHP, kan uppfatta användarinteraktion till exempel vilka knappa användaren trycker på eller vart muspekaren befinner sin i webbläsaren samt vad användare har tryckt på. Servern kan också hantera det senast nämnda men detta kommer att belasta den och samtidigt ta mer tid med om samma funktion utförs med hjälp av JavaScript görs den lokalt i webbläsaren och resultera i nästan realtidrespons och mindre belastning på servern.

Denna funktionaliteten gjorde att JavaScript användes i systemet för att till exempel lysa upp vilken rad användaren hade muspekaren över eller vilken av tabellerna skulle visas upp beroende på vilken länk användaren tryckte på utan att behöva hämta tabellen för varje klick. Den senare nämnda funktionaliteten avlastade servern väldigt mycket och snabbade upp processen eftersom tabellerna kommer växa sig väldigt stora till slut vilket skulle förlänga processen otroligt mycket. När det kommer till den förstnämnda lösningen, att lysa upp de raderna i tabellen användaren hade muspekaren över, valdes den eftersom den informationen som Compare Testlab önskade sig skapade en väldigt stor tabell och denna i sin tur var inte speciellt användarvänlig. För att underlätta användandet och avläsningen lyses en rad upp med en ljusare ton av bakgrundfärgen så att användaren kan klart och tydligt se vilken rad den nu läser.

2.3.5 Virtuell server

En virtuell server är mjukvara som körs på en serverdator som agerar som en fristående server. Det vill säga; det är en viss mängd av en serverdators resurser som agerar som en egen serverdator. På

den virtuella servern installeras sedan ett operativsystem, i detta fall CentOS, för att den skall kunna konfigureras att serva klienter som kommunicerar med den.

Fördelarna med en virtuell server är att en enda serverdator kan vara värd för flera ändamål samtidigt. Detta är perfekt för vårt projekt, då den server det är implementerat på inte behöver ha stora mängder resurser, men samtidigt bör vara dedikerad enbart till projektet så att konfigurationerna på den kan anpassas fullkomligt.

2.3.6 SSH-tunnel

SSH-tunneln mellan servern och utvecklingsdatorerna användes för att köra shell-kommandon på servern över nätverket. Detta innebar att nödvändiga åtgärder, så som att ändra i konfigurationsfiler, installera paket och flytta eller ändra i filsystemet gick att göra från vilken dator som helst på nätverket som hade SSH installerat om rätt användaruppgifter angavs. För det mesta kopplades utvecklingsdatorerna till servern över SSH-tunneln som root-användare, för att ha fullständig kontroll över den och kunna anpassa den fritt utan restriktioner.

2.3.7 CentOS server

CentOS (Community Enterprise Operating System) är en fri distribution av Linux, baserad på Red Hat Enterprice Linux (RHEL), vilket i sin tur är en kommersiell distribution som kräver licens. Anledningen till att det finns fria varianter på RHEL är att utvecklaren är bunden, enligt licenser för öppen källkod, till att hålla källkoden allmänt åtkomlig. CentOS skapades genom ”The CentOS Project” med mål att hålla operativsystemet fullt kompitabelt med all den mjukvara som finns tillgänglig för RHEL. Eftersom RHEL i sin tur är en, sedan länge, väldigt populär distribution och även väldigt väl underhållen, gör det att CentOS är det mest använda operativsystemet för servrar, med öppen källkod.

När CentOS installeras på en ny server kommer det med en mängd för installerad mjukvara. I just detta projekt visade det sig vara till både för- och nackdel, då vissa program var sådana vi behövde under utvecklingen, t.ex. PHP, medan andra medför mer säkerhetsspärrar och liknande än behövligt vid användning på ett internt nät, t.ex. SELinux (Security-Enhanced Linux), vilket var vad denna server skulle användas till. Funktionen SELinux har på operativsystemet är att det blockerar tillgången till filer och processer på servern i väldigt hög utsträckning. För att gå runt dessa blockeringar är administratören tvungen att införa undantag i konfigurationen för SELinux, vilket är ganska besvärligt. Eftersom vi ansåg att körning av servern utan SELinux aktiverat var ofarligt vid endast lokal användning var det vårt beslut att stänga av funktionen helt.

2.3.8 QRencode

För att skapa de QR koder (se kapitel 2.3.3) användes det ett litet bibliotek som efter att ha laddats ner och installerats, kan kommas åt med hjälp av ett bash-kommando för att koda in vald text, som skickades med som en parameter, till en QR motsvarighet. Ursprungskommandot som används för att utföra denna uppgift är följande:

- `qrencode -o ~/Desktop/google.png -s 6 'http://google.com'`

Kommandot tar in ett flertal parameter som bestämmer hur utskriften kommer se ut. De flaggorna som används ovan är –o och –s. Flaggan –o bestämmer vart filen skalla sparas samt med vilket namn där flaggan –s bestämmer storleken för slutkoden som sparas. Slutligen i exemplet ovan matas in strängen som användaren önskar koda inom citationstecken, i fallet ovan har man valt att mata in `'http://google.com'`. Dock använder systemet en extra flagga till parametrar vid tillverkningen av QR-koderna. Den extra flaggan som används är –m denna markerar parametern som står för marginalen emellan QR–koden och kanten av bilden till exempel:

- `qrencode -o ~/Desktop/google.png -s 6 'http://google.com' –m 2`

Om flaggan inte anges kommer qrencode anta standard värdet som är 4. Eftersom qrencode inte är ett standard bibliotek måste den därför laddas ner och installeras innan den kan användas.

2.3.9 Apache

Apache är en open source webbserver som introducerades till marknaden år 1995 och sedan dess blivit den mest använda webbservern i välden. Apaches open source karaktär gör den enkel att anpassa till användarens behov vilket vägde tungt vid valet av just denna server.

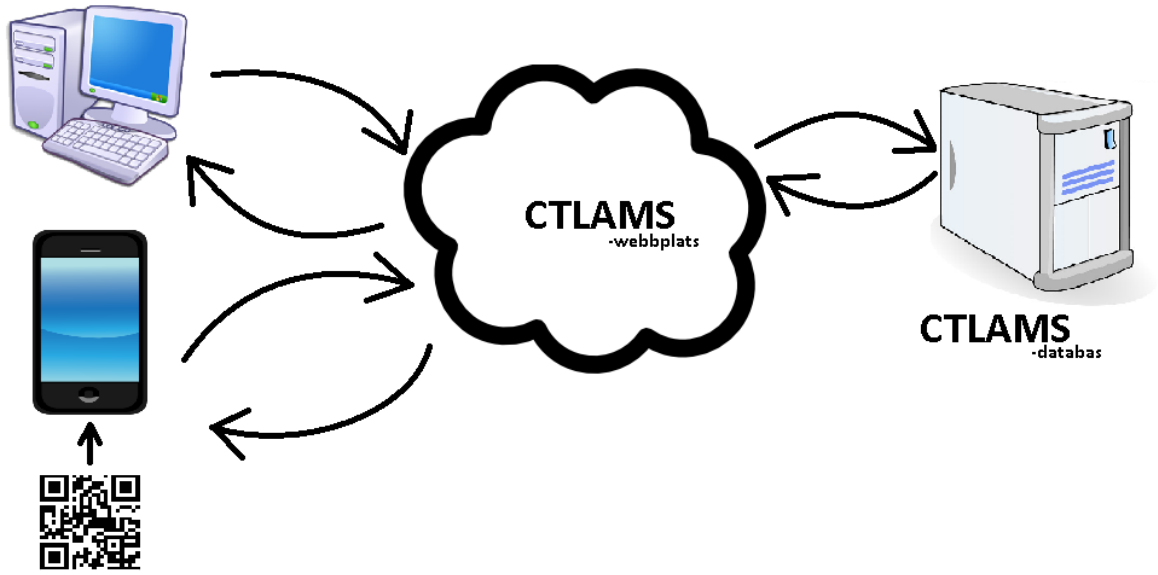
Den initiala installationen av apache medförde alla de plug-in moduler som behövdes dock krävdes det en hel del konfiguration. Detta visade sig vara väldigt problematiskt men i sin tur fanns det en väldigt utförlig dokumentation över hela konfigurationen ute på nätet (länken kan hittas i referenserna). Hela konfigurationsprocessen gick dock väldigt problemfritt vilket ledde till att vidare stegen i utvecklingen kunde tas inom kort.

2.4 Kapitelsammanfattning

Eftersom projektet kom till väldigt hastigt ledde detta till att en kravspecifikation inte existerade i början. Detta gav inga klara grunder eller önskemål mer än behovet av ett system som skall kontrollera alla tillgångar. I en liten grad är det tillräckligt för att börja med men det försvårar framtagningsprocessen väldigt mycket. Det är dock inte ett förlorat slag att inte ha en kravspecifikation klar i början eftersom utvecklingsgruppen får vara med och påverka valen vilket i sin tur minskar bollandet av demoversioner och framtagandet av onödiga funktioner. Detta på grund av att systemutvecklarna (vi) och kunderna (Compare Testlab) kan föra en dialog väldigt tidigt och tillsammans ta fram en väldigt detaljerad och korrekt kravspecifikation som håller länge. Med tanke på att projektet inte var välplanerat från själva början var det inte alls enkelt att börja vilket i sin tur ledde till en väldigt sakta uppploppsträcka. Efter att ha kommit över det initiala träsket av oklara krav och tagit fram en klar och tydlig kravspecifikation påbörjades framtagningsprocessen. Compare Testlab tillgodosåg en virtuell server och arbetsstationer samt en lokal och miljö att utveckla systemet i vilket underlättade processen och gjorde det möjligt att arbeta väldigt agilt. Eftersom utvecklarna (vi) var vägg i vägg med kunderna (Compare Testlab) kunde man föra en kontinuerlig dialog och fånga in onödiga funktioner och misstag tidigt vilket gjorde hela processen mycket mer smidig och billig, rent tidsmässigt eftersom den tid som tilldelades till framtagandet inte slösades bort på att reparera djupgående misstag.

Valet av implementeringar och miljöer gjordes till en del av Compare Testlab men valen av språken och ett fåtal lösningar var upp till utvecklarna (oss). Till exempel: användningen av en simpel scanner istället för att skapa en egen applikation som går igenom närmare i kapitel 2.3.3 eller användningen av QR-kod istället för GTIN (EAN) som kan ses i samma kapitel. De miljöer som användes under utvecklingen var redan valda av Compare Testlab eftersom de som företag använder sig av en specifik standard som i detta fall är Linux och andra open source produkter. Detta är den största anledningen till varför de valde att använda PostgreSQL istället för MySQL, eftersom MySQL inte längre är open source, mer om just detta val i kapitel 2.3.2.

I Figur 25 presenteras den övergripande funktionaliteten av systemet, där CTLAMS är en samverkan mellan webbplatsen och databasen.



Figur 25 - Översiktlig bild över Systemet.

För en mer detaljerad sammanfattning av hela framtagningsprocessen samt de utvecklingsmöjligheter som lämnades öppna för systemet se kapitel 4 med dess underliggande subkapitel.

3 Resultat & utvärdering

Eftersom alla problem under utvecklingen av ett system bidrar till framdriften av utvecklingen gjordes valet att ägna detta kapitel åt alla de problemen som bidrog mest till att systemet kom till.

3.1 Introduktion

Ett problem som placeras bäst i introduktionen är det problemet som Compare Testlab hade: ”Avsaknaden av ett riktigt system för att hålla sina tillgångar i schack”. Compare Testlab konstruerade en temporär lösning på detta problem med ett excelark som låg lokalt eller skickades runt med email, en mer utförlig beskrivning av Compare Testlabs lösning kan ses i kapitel 1.3.1.

3.2 Problem

Under projektets gång stötte vi på en mängd problem där de flesta av dem var tekniska som till slut blev lösta för att gå vidare i utvecklingen. Listan på alla problem sträcker sig lång och därför gjordes valet att selektivt plocka bort vissa problem och presentera dem som representerar de olika stadier och milstolpar i projektets gång. I första hand beskrivs det initiala problemet att komma åt och skapa en stabil uppkoppling mot databasen. Problemen med PHP beskrivs i ett senare subkapitel och slutligen presenteras problemen med SQL som var det sista steget i projektet borträknat från de kosmetiska frågorna.

3.2.1 pgAdmin problem

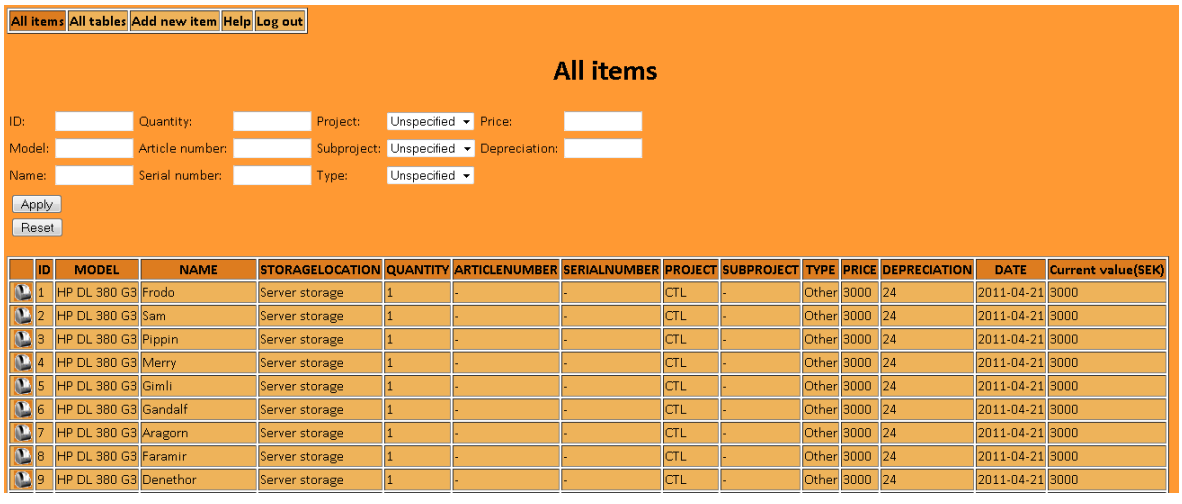
Det allra första problemet som uppkom var anslutningen till den virtuella servern (se kapitel 2.3.5). Efter att ha införskaffat administrationsprogrammet, pgAdmin III, till PostgreSQL från deras egen hemsida uppstod det genast en mängd problem vid anslutnings försök. Efter otaliga antal felmeddelanden vid anslutning och validerings förfrågningar resulterade felsökningen i att filen postgreSQL.conf, filen som håller i alla parametrar för databasen så som sökvägar såväl som logdestinationer med mera, pekades ut. I denna fil krävdes det ett antal ändringar för att tillåta en utomstående anslutning. Efter ändringen av listen_adress till ’*’, vilket innebär att databasen skulle istället för att lyssna på endast en specifik adress lyssna på alla adresser som kunde tänka sig kopplas upp. Detta innebar en stor risk med tanke på att nu kan alla IP adresser koppla upp sig mot databasen och manipulera dess tabeller men samtidigt var det inte en så stor risk för Compare Testlab eftersom de skulle använda databasen endast lokalt och de antog att inga av deras anställda skulle vilja sabotera deras egna system. Den andra ändringen som utfördes var att ta bort det förinställda lösenord skyddet igenom att ändra ”METHOD” till trusted vilket innebar att databasen skulle lita på alla uppkopplingar och inte be dem om något lösenord. Denna ändring utfördes i samma fil som ändringen för vilka adresser som skulle avlyssnas.

3.2.2 PHP-problematik

Kommande problemet var ett av de mer tidskrävande. PHP sidorna kunde inte komma åt informationen som låg i databasen vilket resulterade i att de inte hade någonting att pressentera för användaren. Felsökningen visade att SELinux (se kapitel 2.3.7) blockerade httpd's anslutning mot databasen. Detta var inte den enda anledningen till varför PHP sidorna inte kom åt någon information, när databasen skapade gjordes det en användare med alla rättigheter för att kunna modifiera all information igenom pgAdmin III. Denna användare antogs ha följt med till Apache (se kapitel 2.3.9), vilket i sin tur visade sig vara fel. När Apache försökte komma åt databasen försökte den göra det som en ny användare, detta gjordes möjligt igenom ändringarna som nämns i tidigare punkt. Eftersom en ny användare inte har några rättigheter alls innebar det att användare Apache inte kunde komma åt någonting alls. Detta åtgärdades igenom att ge användare Apache alla de rättigheter som krävs och i sin tur möjliggjorde det för PHP sidorna att komma åt den nödvändiga informationen.

3.2.3 SQL-komplikationer

Vid skapelsen av filtreringsfunktionen uppstod det ett par mindre problem. Ett av de första problemen var att SQL frågan som hämtar ut relevanta data skulle byggas upp allt eftersom användaren fyllde in fler filteralternativ. Detta skapade problemet med en fråga som skall byggas upp dynamiskt, lösningen var att göra en samling IF-satser som beroende på vad användaren hade fyllt in, till exempel om användaren fyllde in ett artikel nummer skulle frågan bara inkludera artikel nummer och sedan fyllas på med ytterligare data så som: ”Namn”, ”Pris”, ”Antal” och så vidare. I Figur 26 kan man se ett exempel på filtreringen.



	ID	MODEL	NAME	STORAGELOCATION	QUANTITY	ARTICLENUMBER	SERIALNUMBER	PROJECT	SUBPROJECT	TYPE	PRICE	DEPRECIATION	DATE	Current value(SEK)
	1	HP DL 380 G3	Frodo	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	2	HP DL 380 G3	Sam	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	3	HP DL 380 G3	Pippin	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	4	HP DL 380 G3	Merry	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	5	HP DL 380 G3	Gimli	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	6	HP DL 380 G3	Gandalf	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	7	HP DL 380 G3	Aragorn	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	8	HP DL 380 G3	Faramir	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000
	9	HP DL 380 G3	Denethor	Server storage	1	-	-	CTL	-	Other	3000	24	2011-04-21	3000

Figur 26 – Filter sidan i systemet.

3.2.4 Kaskad borttagning & främmande nycklar

Efter en kortvarig testfas som Compare Testlabs stod för själva upptäcktes det ett litet problem. När administratören tog bort ett subprojekt visades inte tillhörande föremålen i tabellen på framsidan. Detta berodde på att databasen, vid borttagning av ett subprojekt, satta alla andra värden som använde denna subprojektets id som främmande nyckel till ”null” istället för att ändra det till ett standard projekt. Detta problem var simpelt att åtgärda eftersom allt som behövde ändras var den raden kod som satt värden på främmande nycklar från ”null” till ett standars värde som i detta fall är ett projekt som kallas CTL (Compare Testlab).

3.3 Kapitelsammanfattning

Grundprojektet var relativt litet och omfattade att skapa en stadig grund till vidare utveckling. Detta var så klart inte den enda anledningen till varför projektet kom till dock var det största kravet. Kravet att skapa en grund som Compare Testlab kunde utveckla ett större och mer globalt system på. Vilket i sin tur innebar att systemet skulle vara tåligt och klara av att köras samtidigt som det utvecklades vidare. Alla de små problemen och de stora har bidragit till att systemet blev tåligt och robust så länge Compare Testlab inte väljer att använda systemet på det offentliga nätverkat, world wide web. Vid det fallet krävs det en del säkerhetsåtgärder så som en extra del av databasen med de olika inloggningarna samt ett lite mer avancerade säkerhetsåtgärder vid login och verifikation.

Den delen av projektet som visade sig vara mest problematisk var serversidan av systemet där det uppkom flest problem och fel. Allt från småsaker som utskrifter som tack vare sin felplacering försökte komma åt det data som inte fanns just i anropets scope, till de riktigt stora som fel behörigheter som resulterade i att användaren inte kom åt någonting i databasen. Som tidigare nämnt är alla projekt problemlösnings processer som drivs fram av just problem och kan både falla ner o marken och aldrig tas upp igen eller blomstra. I detta fall var det, det senare nämnda alternativet.

4 Slutsatser

Detta kapitel kommer att ägnas åt att sammanfatta hela processen, alla problemen och resultatet.

4.1 Sammanfattning

I början an använde Compare Testlab sig av ett excelark för att hålla koll på alla sina resurser. Detta fungerade ett tag fram tills antalet tillgångar överskred gränsen av en A4 sida, detta innebär att användaren som ville söka bland alla resurser efter en specifik fick söka igenom detta för hand. Detta är inte ett optimalt sätt att arbeta på och inte heller ett optimalt sätt att kontrollera alla sina resurser på. Ur det ”systemet” föddes behovet av ett mer sofistikerat system med mer funktionalitet och skalbarhet.

När vi kom till Compare Testlab fanns det inte mer än en idé och ett fåtal förslag på användarfall. Efter ett kort tag togs det fram en kravspecifikation och arbetet påbörjades.

4.1.1 Prototyp

Systemet har igenomgått tre faser i sin utveckling. Den första fasen var en beta fas som inkluderade tester av databasen, olika implementeringsspråk samt olika designalternativen för databasen. Efterkommande fasen ägnades åt att ta fram en fungerande version av systemet, en version 0.1. Detta innebär, en fungerande databas med ett gränssnitt som går att använda även om alla finesser och all funktionalitet inte finns på plats ännu. Det enda som skulle finnas tillgängligt var en grundläggande funktionalitet. Den slutgiltiga fasen var fullständigt ägnad åt en version 1.0 på systemet. Med detta menas att den färdiga varan skulle vara klar att implementeras och en dokumentation för kommande administratörer skulle finnas på plats och tillgänglig.

4.1.1.1 Beta-fas

I denna fas las mesta delen av tiden på att ha klart ett skelett, en ”benstomme” att bygga resten av systemet på. Denna stomme bestod av en provisorisk databas i PostgreSQL (se kap. 2.3.2), en enkel HTML-sida med PHP-script som tar fram all den informationen som finns i databasen och kastar ut den på hemsidan utan någon vidare igenomtänkt struktur. Samt ett stort antal av testfall för att täcka in alla möjliga problem och behov för systemet. Under denna fas finslipades även kravspecifikationen och fylldes på med de framtagna testfallen.

4.1.1.2 Version 0.1

När beta-fasen avslutades stod den stadiga grunden och höll upp systemet så att påbyggnaden av alla viktiga funktioner kunde börjas. Under denna fas implementerades de livsviktiga funktionerna så som:

- Att lägga till ett föremål till databasen,
- Ta bort ett föremål ur databasen,
- Ändra i ett befintligt föremål,
- Sök/filtrera bland föremålen,
- Visa alla föremål i databasen,
- Utskrift av QR-koder.

Där föremålen var resurser, lagerutrymmen, de olika projekten Compare Testlab sköter samt de subprojekten som hör till.

Denna funktionaliteten ansågs vara viktigast att ta fram utifall systemet skulle implementeras för användning innan dess utsatta deadline hade löpt ut.
Efter att denna fas klarades av hade både en stadig bas skapats likväl en grund funktionalitet vilket gjorde att den återstående utvecklingen var endast en finslipning av de grova kanterna och implementering av de mindre viktiga funktionerna som Compare Testlab hade önskat. Dessa funktioner var:

- Kontrollera nuvarande värde på alla föremålen,
- Räkna ut avskrivningsvärde baserat på dagens datum samt avskrivningsreglerna,
- Räkna ut det totala värdet på alla föremålen som visas på hemsidan för tillfället.

4.1.1.3 Version 1.0

Den sista fasen systemet gick igenom var en ”testfas”. I denna fas började Compare Testlabs anställda att använda systemet och stresstesta den så väl som kontrollerar om funktionaliteten var uppdaterad och gjorde det som begärdes.

När testingen påbörjades upptäckte en av anställde ett mindre problem:

- När ett subprojekt togs bort medans det hade ett föremål skrivet till sig, uteslöts det föremålet från listan på alla föremål på startsidan.

Anledningen till problemet visade sig vara ett gammalt kodstycke som fortfarande låg kvar i systemet efter den allra första fasen. När detta åtgärdades var Compare Testlab väldigt nöjda med webbapplikationen och valde att börja använda den på heltid.

4.1.2 Androidapplikation

Efter att de tidigare faserna var avlutade lades tiden på att utveckla en Androidapplikation. Den grundliga idén var att använda en specifik miljö och skriva en hel applikation från grunderna. Tyvärr eftersom tiden var knapp och licensen till miljön skulle kunna ta väldigt lång tid att ta fram.
Efter ett slutligt möte och en kort presentation av systemet, till arbetsgivarna samt de anställda, framtogs idén att istället för att skapa en helt ny applikation skulle det användas en vanlig läsare, som finns att ladda ner till alla smartphones, för att läsa av koderna och kontakta en egen PHP-sida som skapades beroende på vilket ID skickades med i länken. IDn skulle kodas in i QR-koden tillsammans med länken till hemsidan. Figuren nedan (figur 27) illustrerar adressfältet på webbläsaren Mozilla med en länk till hemsidan för ett specifikt objekt.

Figur 27 – länkadressen till hemsidan för enskilda föremål i databasen.

4.2 Framtida utveckling

Det slutgiltiga systemet var komplett sett till den funktionalitet som önskats. Vi hann implementera allt som var planerat under projektet och även hunnit testköra det samt åtgärdat de fel som upptäckts. Så länge uppdragsgivaren inte önskar lägga till ytterligare funktionalitet i systemet skall utvecklingen vara klar. Skulle detta inträffa har vi vidtagit vissa åtgärder för att underlätta eventuellt utvecklingsarbete, mestadels genom dokumentation och beskrivning av systemet.

I projektets slutskede såg det först ut som att applikationen för smartphones skulle behöva lämnas helt och hållet till framtida utveckling av systemet. Vilket inte sågs som en helt dålig idé då Compare Testlab ofta erbjuder projektuppgifter för praktikanter från både gymnasium och högskola, och en sådan applikation säkerligen hade varit ett intressant projekt. Men i stället för att lämna applikationen för smartphones löste vi det på ett enklare sätt genom en PHP sida (se delkapitel 2.3.1.7 för beskrivning).

Det kan tänkas att en vidareutveckling av systemet kan innebära att funktionaliteten som är åtkomlig från smartphones utökas. För tillfället är den enda interaktion som är anpassad för smartphones att hämta ut information om ett föremål i databasen genom att scanna dess streckkod. Det är även

möjligt att komma åt övrig funktionalitet genom webbläsaren i telefonen, men det är tämligen osmidigt och webbsidorna är inte anpassade för en så liten skärm. I ställen skulle man kunna skapa en applikation, förslagsvis för Androidtelefoner, där gränssnittet passar skärmen bättre. En sådan applikation skulle förslagsvis kunna kommunicera direkt med databasen via SQL-frågor och på så sätt utföra samma åtgärder som en användare kan göra på webbplatsen. För att använda sig av vårt arbete för att underlätta en sådan utveckling skulle utvecklarna behöva tillgång till källkoden som vi har överlämnat till uppdragsgivaren. I källkoden skulle de sedan behöva läsa och till viss del förstå hur vi har använt SQL-frågor och databasens svar för att presentera och manipulera objekten. De kommentarer som finns till i stort sett varenda funktion i källkoden och relation i databasen skulle underlätta detta avsevärt, men mer än så skulle inte vårt system vara till nytta för utvecklarna.

I övrigt skulle en anledning till att vidareutveckla systemet vara behov av utökad funktionalitet. Även då skulle utvecklarna behöva sätta sig in i källkoden och förstå vad vi har gjort för att kunna dra nytt av vårt arbete. Vi har försökt underlätta det genom att kommentera koden så utförligt som möjligt. Det kan även tänkas att vår källkod används som hjälp med förslag på hur man kan använda PHP-funktioner och SQL-frågor för att manipulera databasen vid tilläggning av ytterligare webbsidor på webbplatsen.

5 Akronym & Ordlista

ASP – Active Server Pages.

AWK – är ett hjälpprogram för datautvinning och rapportering.

Bash – kommandotolken för Unix system.

CGI – står för: Common Gateway Interface.

DOM – står för Document Object Model som är en plattform och språkoberoende konversion för objekt i bland annat HTML.

EAN – European Article Number.

Flagga – användardefinierade värden vid bash anrop för utökning, ändring och reducering av funktionaliteten.

GTIN – det nya namnet som ersätter EAN, står för Global Trade Item Number.

HTML –Hyper Text Markup Language.

ICT –Information and communication technology vilket I sin tur betyder informations och kommunikations teknologi. Ett längre namn på IT.

IT- Informations Taknologi, den teknologin som hjälper oss bearbeta all information.

Null – en karaktär som är 0 bits stor.

OOP – står för: Object Oriented Programming, objekt orienterad programmering.

PHP – ett rekursivt akronym för: PHP Hypertext Preprocessor.

QR – ett akronym för: Quick Responce, en kodstandard för kodning av texter i en grafisk motsvarighet.

RHEL – Red Hat Enterprise Linux.

Sed – en strömeditor.

SELinux – Security-Enhanced Linux som är en funktion som möjliggör användning av avancerade säkerhetsimplementeringar i Linux kärnan.

SSH – Secure shell, ett nätverksprotokol för säker kommunikation emellan två enheter.

SQL – förkortning för: Structured Query Language.

TP- står för: Twisted Pair den teknologin som används i nutida internetkablar. Teknologin negerar den negativa effekten av elektromagnetiska störningarna.

UCC – står för: Uniform Commercial Code.

Xml – denna förkortning är den filändelsen som tilldelas excel filer.

Yum – står för Yellowdog Updader, Modified. En opensource kommandorad för nedladdning och installation av RPM paket till Linuxbaserade system.

6 Källor

[1] http://compare.se/index.asp?fu_id=153 [Senast besökt:2011-05-09] – Compare Testlabs egna hemsida med företags information och historia.

[2]http://www.webmasterworld.com/forum91/441.htm [Senast besökt:2011-05-09] - guiden för dynamisk användning av taggar i HTML.

[3]http://www.permadi.com/tutorial/cssHighlightTableRow/index.HTML [Senast besökt:2011-05-09] - guide för implementering av belysta rader i tabellerna.

[4]http://www.bizfive.com/articles/web-design/comparing-PHP-and-asp.net/ [Senast besökt:2011-05-09] – jämförelse av ASP.NET med PHP.

[5]http://www.xenocafe.com/tutorials/linux/centos/apache_web_server/index.PHP [Senast besökt:2011-05-09] - guide för installation av en Apache server.

[6]https://ascherconsulting.com/what/are/the/advantages/of/using/postgreSQL/versus/mySQL/ [Senast besökt:2011-05-09] – sammanfattning av fördelarna med PostgreSQL över MySQL.

[7]http://www-css.fnal.gov/dsg/external/freeware/pgSQL-vs-mySQL.HTML [Senast besökt:2011-05-09] - granskning av skillnader och likheter emellan PostgreSQL och MySQL.

[8]http://sv.wikipedia.org/wiki/European_Article_Number [Senast besökt:2011-05-09] - information om EAN koden samt dess historia och utveckling.

[9]http://www.denso-wave.com/qrcode/index-e.HTML [Senast besökt:2011-05-09] - Denso wave hemsidan, informationskälla för QR-koden.

[10]http://httpd.apache.org/ABOUT_APACHE.HTML [Senast besökt:2011-05-09] - hemsidan för Apacheorganisationen.

[11]http://www.linuxcertif.com/man/1/qrencode/ [Senast besökt: 2011-05-11] – man sidan för qrencode biblioteket.

[12]http://www.gentoo.org/doc/sv/handbook/handbook-x86.xml?part=2&chap=2 [Senast besökt: 2011-05-11] – definition av ”flagga” i Linux sammanhang.

[13]http://www.postgreSQL.org/docs/8.1/static/runtime-config.HTML [Senast besökt:2011-05-09] - dokumentation för PostgreSQL databasen.

[14]http://docs.python.org/howto/webservers.HTML [Senast besökt:2011-05-09] - användningsguide och beskrivning av språket PYTHON.

[15]http://www.scribd.com/doc/9668146/5-Steps-To-Write-the-Abstract-of-Scientific-Writing [Senast besökt: 2011-05-20] - Instruktioner för hur man skriver en Abstract i vetenskapliga uppsatser

7 Bilagor

Bilaga 1 – Kravspecifikation

Kravspecifikation

- **Åtkomst:**
    - Webb UI,
    - Androidapplikation:
        - Scanna in en sträckkod →namn på föremålet och platsen den skall befinna sig på.
- **Behörighet:**
    - Administratör:
        - Redigera befintliga inlägg,
        - Lägg till/ta bort föremål/projekt,
        - Sök efter föremål/projekt.
    - Gäst:
        - Söka efter föremål/projekt.
- **Lägg till:**
    - Nya projekt/subprojekt,
    - Enskild inmatning av investeringsvaror,
    - Inmatning av flera förbrukningsvaror samtidigt,
    - Nya lagringsplatser.
- **Sökning:**
    - Sök efter föremål,
    - Räkna ut:
        - Nuvarande värdet (med hänsyn till avskrivningsregler),
        - Inköpsvärdet för alla föremål i sökningen.
- **Redigering/borttagning:**
    - Investeringsvaror,
    - Förbrukningsvaror (minska antalet eller ta bort helt),
    - Projekt & subprojekt (kaskad borttagning),
    - Lagringsplatser.

Bilaga 2 - Källkod

add.php

```
<!--
*** The page to add items to the different tables
*** This page takes two get-paramaters, 'table' and 'item'
-->
<?php
include 'sql_queries.php';
//Define standard query with the get-parameter identifying the table
$query = "SELECT * FROM assetmanagement.\"".$_GET['table']. "\" ORDER BY
 \"".$_GET['table']."\".\"ID\" ASC";

//Store result for the standard query
$result = pg_exec($db_handle, $query);
//Sql query getting all the foreign keys in the database
$fkeys = pg_exec($db_handle, get_fkeyquery($_GET['table']));


//If the if-statement is true, the user has pressed the Submit-button to add the item
if(isset($_POST['submit'])){
                        //Query to inser the item
                        $addquery = "INSERT INTO assetmanagement.\"".$_GET['table']."\" VALUES (";
                          // This loop extracts the name of the different fields in the database table
                          // and builds the values-part of the sql query to insert the values from the form-fields into the fields
    $values = pg_fetch_row($result, pg_num_rows($result)-1);
    for ($col = 0; $col < count($values); $col++) {
                                                        if($col == 0) $addquery .= "'".($values[0]+1)."'";
                                                        else if($_POST[pg_field_name($result, $col)] == '') $addquery .= ", null";
                                                        else $addquery .= ", '". $_POST[pg_field_name($result, $col)]."'";
 }
                        $addquery .= ")";
                        //Execute the INSERT-query
                        $addresult = pg_exec($db_handle, $addquery);
                        //If the if-statement is true, the INSERT-query failed
                        if(!$addresult){
                                        //Display error pop-up with the error message from the database
                                        ?>
                                        <script type="text/javascript">
                                        alert("<? echo addcslashes(pg_errormessage($db_handle),"\"");?>")
                                        </script>
                                        <?
                        }
                        //else: update the list of items
                        else $result = pg_exec($db_handle, $query);
}
//If the if-statement is true, the user has pressed the logout-button
if(isset($_POST['logout'])){
                        session_start();
   $_SESSION['phplogin'] = false;
   header('Location: login.php');
   exit;
}
?>
<html>
 <head>
  <title>Add new item</title>
 </head>
 <body class="background">
<center>
<h1>Add item</h1>
<? #Functions
//Function to construkt a drop-down menu which is used for selecting value in the fields with foreign-keys
//See the documentation for further explaination
function mkDropdown($_fkeys, $_fkeyRow,  $db_handle){
                        $fvalues = pg_fetch_row($_fkeys, $_fkeyRow);
                        $dropresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"".$fvalues[2]."\"");
                        echo "\n\t<select name=\"".$fvalues[1]."\">";
                                        for ($row = 0; $row < pg_numrows($dropresult); $row++) {
```

```
                                                                $dvalues = pg_fetch_row($dropresult, $row);
                                                                echo "\n\t\t<option value=\"".$dvalues[0]."\"";
                                                                echo ">" . $dvalues[1] . "</option>";
                                                        }
                                                echo "\n\t</select>";
}

//Getting data from database to fill the drop-down menus
//This part will not adjust if further foreign keys are added
$projectresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"PROJECT\"");
$subprojectresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"SUBPROJECT\"");
$typeresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"TYPE\"");

//If the if-statement is true, the standard query was successful
if ($result) {
  echo "</center>";
  //Table with fields for the new item
  echo "<table border=\"1\">";
  echo "<tr class='highlight'><th colspan=".pg_num_fields($result).">";
  echo "New item";
  echo "</th></tr>";
  echo "<tr class='highlight'>";
  for ($j = 1; $j < pg_num_fields($result); $j++)
  {
    $fieldname = pg_field_name($result, $j);
    echo "\t<th>". $fieldname . "</th>";
  }
  echo "\n</tr>";
  echo "\n<form method=\"post\" action=\"\">";
     echo "\n<tr>\n";
     for ($col = 1; $col < pg_num_fields($result); $col++) {
                                                        echo "\t<td>";
                                                if(($fkeyRow = isFkey(pg_field_name($result, $col),$fkeys))!=(-1)){
                                                        mkDropdown($fkeys, $fkeyRow, $db_handle);
                                                }
                                                else{
                                                $idvalues = pg_fetch_row($result, pg_num_rows($result)-1);
                                                        ?><input size="15" type="text" maxlength="40" name="<?echo
pg_field_name($result, $col); ?>" <? if(pg_field_name($result, $col) == 'DATE') echo "title=\"YYYY-MM-DD\""; ?>><?
                                                        }
                                                        echo "</td>\n";
     }
                                                echo "</tr>\n";
                                                echo "</table><table>";
                                                echo "<tr><td><input name=\"submit\" type=\"submit\" value=\"Apply\">";
                                                echo "</form>";
                                                echo "</td></tr>";
                                                echo "</table>";
                                                echo "<br>";
                                                echo "<br>";

                                                //Table presenting existing items
                                                echo "<table border=\"1\">";
                                                echo "<tr class='highlight'><th colspan=".pg_num_fields($result).">";
                                                echo "Existing items in this table\n";
                                                echo "</th></tr>\n";
                                                echo "<tr class='highlight'>";
                        //Loop for the table-headers
  for ($j = 0; $j < pg_num_fields($result); $j++)
  {
    $fieldname = pg_field_name($result, $j);
    echo "<th>". $fieldname . "</th>";
  }
  echo "</tr>\n";
  //Loop for the table rows
for ($row = 0; $row < pg_numrows($result); $row++) {
     $values = pg_fetch_row($result, $row);
     //Making the rows clickable and hyper-linked
                                                echo "<tr
onClick=\"document.location.href='manage_item.php?item=".$values[0]."&table=".$_GET['table']."';\" class='normal'
onMouseOver=\"this.className='highlight'\" onMouseOut=\"this.className='normal'\">";
```

```
    for ($col = 0; $col < count($values); $col++) {
       if($values[$col] == null) { echo "\n\t<td> - </td>"; }
       else {
                                                    echo "\n\t<td>";
                                                    //If the if-statement is true, the field is a foreign key
                                                    if(isFkey(pg_field_name($result, $col), $fkeys)!= (-1)){
                                                                // Print the TITLE-field from the table containing the
foreign key
                                                                // instead of the ID-field as stored in the current table
                                                                        $fvalues = pg_fetch_row($fkeys,
isFkey(pg_field_name($result, $col), $fkeys));
                                                                        $fkeyresult = pg_exec($db_handle,
"SELECT \"".$fvalues[2]."\".\"TITLE\" FROM assetmanagement.\"".$fvalues[2]."\" WHERE \"ID\" = ".$values[$col]);
                                                                        $nvalues = pg_fetch_row($fkeyresult,
0);
                                                                        echo $nvalues[0];
                                                                         }
                                                    else echo $values[$col];
                                                    echo "</td>";
                                                     }
    }
    echo "\n</tr>\n";
  }
                        echo "</table>";
}
//else: the standard query resulted in an error
else {
  echo "The query failed with the following error:<br>\n";
  echo pg_errormessage($db_handle);
}
//Close the database-handle
pg_close($db_handle);
?>
 </body>
</html>
```

help.php

```
<!--
***This help page contains a brief description of all the basic functions of the CTLAMS web page
-->
<?php
include 'sql_queries.php';

//Return to login.php if the user isn't logged in as admin
if(!$_SESSION['phplogin']){
   header('Location: login.php');
   exit;
}
?>
<html>
 <head>
 <title>Help</title>
 </head>
 <body class="background">
 <!-- The help text as paragraphs and headers -->
                        <center><h1>CTLAMS Help</h1></center><br>
                                        <h2>Browse items</h2>
                                                        <p>
                                                                        All items in the standard database table are displayed on
the main page ("All items"-link at the top of the page).<br>
                                                                        To browse items in a different table, for example
"PROJECT", which contains all the main projects, use the "All tables"-link at the top of the page.
                                                        </p>
                                                        <h2>Print a bar code</h2>
                                                        <p>
                                                                        On the main page, use the printer-icon beside an item to
print its bar code. This will print a label on default printer with a QR-code representing the item's ID-number in the database table.
                                                        </p>
                                                        <h2>Depreciation</h2>
                                                        <p>
                                                                        The value in the "Depreciation"-field in the ITEM-table
should be the total number of months over which the item is depreciated. This value is used by the web page to calculate the Current
Value of an item.
                                                        </p>
                                        <h2>Manage an item</h2>
                                                        <p>
                                                                        To manage the attributes of an item, simply click on the
row with the item.<br>
                                                                        This works on the main page as well as the "All tables"-
page.
                                                                        To change the values of the attributes of the item on the
management-page, fill in the fields as desired and press the Apply button.
                                                        </p>
                                        <h2>Add an item</h2>
                                                        <p>
                                                                        To add an item to the default table, use the "Add item"-
link at the top of the page. Fill in the desired fields and press Apply.
                                                                        To add an item to a different table, navigate to the table-
selection page by clicking the "All tables"-link at the top of the page.<br>
                                                                        Then click the desired table in the list and click the "(add
new)" link by the name of the table that appears on the page.
                                                        </p>
                                        <h2>Delete an item</h2>
                                                        <p>
                                                                        To delete an item, go to the page where the attributes of
the item can be managed (See the "Manage an item"-paragraph) and click the Delete-button.<br>
                                                                        The item will be permanently deleted from the database
when the "Okay"-button is clicked on the popup-window that appears.
                                                        </p>
                                        <h2>Browse tables</h2>
                                                        <p>
                                                                        To browse the different tables in the database and their
containing items, use the "All tables"-link at the top of the page than click on the desired table.<br>
                                                                        Here you can also add items to the table and manage the
existing items.
                                                        </p>
```

```
                    <h2>Manage tables</h2>
                        <p>
                                                    Managing the tables, not the items in the tables, can only be done through direct sql-queries to the database or with a stand-alone GUI, for example <a href="http://www.pgadmin.org/">pgAdmin III</a>.
                        </p>

 </body>
</html>
```

index.php

```
<?
header('Location: login.php');
?>
```

item.php

```
<!--
*** This page is used to display 1 item in the database.
*** The QR-codes which are printed with the print-function on the main page contain direct links to this page
*** One GET-parameter is needed, which is the ID-number of the item that is to be displayed
-->
<?php
include 'sql_queries.php';
$_SESSION['phplogin'] = false;

//If the if-statement is true, there were no get-parameters in the URL
if($_GET['item'] == null ){
                        header('Location: list_all.php');
                        exit;
}
                        //Fetch data from database
                        $result = pg_exec($db_handle, get_itemquery("ITEM",$_GET['item']));
                        $fkeys = pg_exec($db_handle, get_fkeyquery("ITEM"));
                        // If the if-statement is true, there were 0 rows in the result from the get_itemquery
                        // This means the name of the table or id-number of the item was invalid,
                        // or that the item no longer exists in the database
                        if(pg_numrows($result)==0){
                                        ?>
                        <script type="text/javascript">
                                        this.location.href = "list_all.php";
                        </script>
 <?
                        }
?>
<html>
 <head>
  <title>Display item</title>
 </head>
 <body class="background">
<h1>Display item</h1>
<? #Functions


//If the if-statement is true, the sql-query execution was successful
if ($result) {
  echo "</center>";
  //Table presenting sql-query results
  $tot_price = 0;
  echo "<table border=\"1\" bgcolor=\"white\">";
  echo "<tr class=\"highlight\">";
  //Build all the table-headers on the first row

  //First <th> is for the printer-image if the user is logged in as admin
  if($_SESSION['phplogin']) echo "<th></th>";
  for ($j = 0; $j < pg_num_fields($result); $j++)
  {
                        $fieldname = pg_field_name($result, $j);
    echo "<th>". $fieldname . "</th>";
  }
  //Add a header for the current-value field, which is not from the database
  echo "<th>Current value(SEK)</th>";
  echo "</tr>";

  //Add all the rows with the items from the database
for ($row = 0; $row < pg_numrows($result); $row++) {
    $values = pg_fetch_row($result, $row);
    $curr_date = null;
    $curr_dep = null;
    $curr_price = null;
    ?>
    <tr class='normal'">
    <?
```

```
    for ($col = 0; $col < count($values); $col++) {
       if($values[$col] == null) { echo "<td> - </td>"; }
       else {
                                                                    echo "<td>";
                                                                    //If the if-statement is true, we are in the first column and should add the
hyper-linked printer-icon
                                                                    if($col == 0 && $_SESSION['phplogin'])
                                                                    {
                                                                                  echo "<a href=print_label.php?item=".$values[$col]."
title=\"Print lable for this item\"><img src=\"print_label.jpg\" align=right></a>";
                                                                                  echo "</td>";
                                                                                  echo "<td>";
                                                                                  echo  $values[$col];
                                                                    }
                                                                    //If the if-statement is true, the field is a foreign key
                                                                    else if(isFkey(pg_field_name($result, $col), $fkeys)!= (-1)){
                                                                                  // Print the TITLE-field from the table containing the
foreign key
                                                                                  // instead of the ID-field as stored in the current table
                                                                                                $fvalues = pg_fetch_row($fkeys,
isFkey(pg_field_name($result, $col), $fkeys));
                                                                                                $fkeyresult = pg_exec($db_handle,
"SELECT \"".$fvalues[2]."\".\"TITLE\" FROM assetmanagement.\"".$fvalues[2]."\" WHERE \"ID\" = ".$values[$col]);
                                                                                                $nvalues = pg_fetch_row($fkeyresult,
0);
                                                                                                echo $nvalues[0];
                                                                                                 }
                                                                    else{
                                                                                  echo $values[$col];
                                                                                  // If either of these if-statements are true, we are in one of
the fields
                                                                                  // which's value is used to calculate the current price.
                                                                                  // This value should be stored in a variable
                                                                                  if(pg_field_name($result, $col)=="PRICE") $curr_price =
$values[$col];
                                                                                  if(pg_field_name($result, $col)=="DATE") $curr_date =
$values[$col];
                                                                                  if(pg_field_name($result, $col)=="DEPRECIATION")
$curr_dep = $values[$col];
                                                                                  }
                                                                    echo "</td>";
                                                                     }
    }
                                                //Function to print a <td> with curr_price of the item
                                                //Calculates price in relation to curr_date, curr_price and curr_dep
                                                if ( $curr_date != null){
                                                // The runnable c-program file age_calc is executed
                                                // and the resulting printed string is stored in the $age variable
                                                // For this to work, age_calc must be located in the same folder as this php-file
                                                // and be executeable by the web-server user, "apache" by default
                                                $age = intval(shell_exec("./age_calc ".$curr_date));
                                                echo "<td>";
                                                if (($curr_price-(($curr_price/$curr_dep))*$age) > 0)
                                                {
                                                                  echo intval(($curr_price-(($curr_price/$curr_dep))*$age));
                                                                  $tot_price += intval(($curr_price-(($curr_price/$curr_dep))*$age));
                                                }
                                                else echo "0";
                                                echo "</td>";
                                                }
                                                else echo "<td>-</td>";
    echo "</tr>";
  }
                                  echo "</table>";
}
//else: the sql-query resulted in an error
 else {
                                                //Display error pop-up with the error message from the database
                                                ?>
                                                <script type="text/javascript">
                                                alert("<? echo addcslashes(pg_errormessage($db_handle),"\"");?>")
```

```
                                        </script>
                                        <?
}
//Close the database handle
pg_close($db_handle);
?>
 </body>
</html>
```

list_all.php

```
<!--
*** The page displayes all the items in the items-table and allows the user to filter the list
*** If the user is logged in as an admin, all rows in the table are hyper-linked to manage_item.php
*** This page can be referred to as the "standard page" of the web site
-->
<?php
include 'sql_queries.php';
//If the if-statement is true, the user clicked the logout-button
if(isset($_POST['logout'])){
   $_SESSION['phplogin'] = false;
   header('Location: login.php');
   exit;
}

//Sql query getting all the foreign keys in the database
$fkeys = pg_exec($db_handle, get_fkeyquery("ITEM"));

?>
<html>
 <head>
  <title>List all</title>
 </head>
 <body class="background">
<center><h1>All items</h1></center>
<? #Functions

//Function building the <option> part of a drop-down menu
function options_loop($_result,$type){
                        for ($row = 0; $row < pg_numrows($_result); $row++) {
                                                                                $values = pg_fetch_row($_result, $row);
                                                                                echo "<option value=\"".$values[0]."\"";
                                                                                if ( $values[0] == $_POST[$type]) echo " selected ";
                                                                                echo ">" . $values[1] . "</option>";
                                                        }
                        }


//Getting data from database to fill the drop-down menus
$projectresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"PROJECT\"");
$subprojectresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"SUBPROJECT\"");
$typeresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"TYPE\"");
$roomresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"ROOM\"");
$buildingresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"BUILDING\"");

   ?>
  <!--
  *** The filter form
  *** This part will not adjust well if fields are added, removed or adjusted in the ITEM-table in the database
  *** Contains a lot of hard-coded functionality
  -->
  <form method="post" action="">
  <table>
                        <tr>
                                                <td>ID:</td> <td> <input title="Database ID of an item" type="text" size="10" maxlength="40"
name="id" value="<? echo $_POST['id']; ?>"> </td>
                                                <td>Quantity:</td> <td> <input title="Quantity between ½ and 1½ times this value" type="text"
size="10" maxlength="40" name="quantity" value="<? echo $_POST['quantity']; ?>"> </td>
                                                <td>Project:</td> <td> <select name="project" >
                                                                        <option value = ""> Unspecified </option>
                                                <?
                                                                        options_loop($projectresult, "project");
                                                ?>

                                                </select></td>
                                                <td>Price:</td> <td> <input type="text" size="10" maxlength="40" name="price" value="<? echo
$_POST['price']; ?>"> </td>
                                                <td>Building:</td> <td> <select name="building">
                                                <option value = ""> Unspecified </option>
                                                <?
```

```
                                                options_loop($buildingresult, "building");
                                        ?>
                        </tr>
                        <tr>
                                        <td>Model:</td> <td> <input type="text" size="10" maxlength="40" name="model" value="<?
echo $_POST['model']; ?>"> </td>
                                        <td>Article number:</td> <td> <input type="text" size="10" maxlength="40"
name="articlenumber" value="<? echo $_POST['articlenumber']; ?>"> </td>
                                        <td>Subproject:</td> <td> <select name="subproject">
                                        <option value = ""> Unspecified </option>
                                        <?
                                                options_loop($subprojectresult, "subproject");
                                        ?>

                                         </select></td>
                                        <td>Depreciation:</td> <td> <input type="text" size="10" maxlength="40" name="depreciation"
value="<? echo $_POST['depreciation']; ?>"> </td>
                        </tr>
                        <tr>
                                        <td>Name:</td> <td> <input type="text" size="10" maxlength="40" name="name" value="<?
echo $_POST['name']; ?>"> </td>
                                        <td>Serial number:</td> <td> <input type="text" size="10" maxlength="40"
name="serialnumber" value="<? echo $_POST['serialnumber']; ?>"> </td>
                                        <td>Type:</td> <td> <select name="type">
                                        <option value = ""> Unspecified </option>
                                        <?
                                                options_loop($typeresult, "type");
                                        ?>
                                        <td>Room:</td> <td> <select name="room">
                                        <option value = ""> Unspecified </option>
                                        <?
                                                options_loop($roomresult, "room");
                                        ?>
                                         </select></td>
                        </tr>
  </table>
  <table>
                        <tr>
                                        <td> <input name="submit" type="submit" value="Apply">
  </form>
  <form><input type="submit" value="Reset"> </form>
  </td>
                        </tr>
  </table>
  <center>
  <?

 $standardquery = "SELECT assetmanagement.\"ITEM\".*, assetmanagement.\"ROOM\".\"TITLE\" as \"ROOM\",
assetmanagement.\"BUILDING\".\"TITLE\" as \"BUILDING\" FROM assetmanagement.\"ITEM\", assetmanagement.\"ROOM\",
assetmanagement.\"BUILDING\", assetmanagement.\"STORAGELOCATION\"
                                                                                WHERE
assetmanagement.\"ITEM\".\"STORAGELOCATION\" = assetmanagement.\"STORAGELOCATION\".\"ID\" AND
assetmanagement.\"STORAGELOCATION\".\"ROOM\" = assetmanagement.\"ROOM\".\"ID\" AND
assetmanagement.\"ROOM\".\"BUILDING\" = assetmanagement.\"BUILDING\".\"ID\" ";

 //If the if-statement is true, the user has NOT submitted a filter
if(!isset($_POST['submit'])){
$query = $standardquery;
$query .= "ORDER BY
                                        \"ITEM\".\"ID\" ASC";
$result = pg_exec($db_handle, $query);
}
// else: a filter has been submitted and the standardquery should be replaced
// This part also contains a lot of hard-coded functionality
else
{
                $query = $standardquery;
                if($_POST['id']!="") {$query .= "AND \"ITEM\".\"ID\" LIKE ".$_POST['id']." ";}
                if($_POST['model']!="") { $query .= "AND \"ITEM\".\"MODEL\" LIKE '%".$_POST['model']."%' ";}
                if($_POST['name']!="")  { $query .= "AND \"ITEM\".\"NAME\" LIKE '%".$_POST['name']."%' ";}
```

```
                        if($_POST['quantity']!="")  { $query .= "AND \"ITEM\".\"QUANTITY\" BETWEEN ".intval($_POST['quantity']*0.5)." AND
".intval($_POST['quantity']*1.5)." ";}
                        if($_POST['articlenumber']!="")  { $query .= "AND \"ITEM\".\"ARTICLENUMBER\" LIKE '%".$_POST['articlenumber']."%'
";}
                        if($_POST['serialnumber']!="") { $query .= "AND \"ITEM\".\"SERIALNUMBER\" LIKE '%".$_POST['serialnumber']."%' ";}
                        if($_POST['project']!="") { $query .= "AND \"ITEM\".\"PROJECT\" = '".$_POST['project']."' ";}
                        if($_POST['subproject']!="") { $query .= "AND \"ITEM\".\"SUBPROJECT\" = '".$_POST['subproject']."' ";}
                        if($_POST['type']!="") { $query .= "AND \"ITEM\".\"TYPE\" = '".$_POST['type']."' ";}
                        if($_POST['price']!="")  { $query .= "AND \"ITEM\".\"PRICE\" BETWEEN ".intval($_POST['price']*0.75)." AND
".intval($_POST['price']*1.25)." ";}
                        if($_POST['depreciation']!="") { $query .= "AND \"ITEM\".\"DEPRECIATION\" BETWEEN
".intval($_POST['depreciation']*0.8)." AND ".intval($_POST['depreciation']*1.2)." ";}
                        if($_POST['room']!="") { $query .= "AND assetmanagement.\"ROOM\".\"ID\" = '".$_POST['room']."' ";}
                        if($_POST['building']!="") { $query .= "AND assetmanagement.\"BUILDING\".\"ID\" = '".$_POST['building']."' ";}
                        $query .= "ORDER BY
                                        \"ITEM\".\"ID\" ASC";
                        $result = pg_exec($db_handle, $query);
}
//If the if-statement is true, the sql-query execution was successful
if ($result) {
  echo "</center>";
  //Table presenting sql-query results
  $tot_price = 0;
  echo "<table border=\"1\" bgcolor=\"white\">";
  echo "<tr class=\"highlight\">";
  //Build all the table-headers on the first row

  //First <th> is for the printer-image if the user is logged in as admin
  if($_SESSION['phplogin']) echo "<th></th>";
  for ($j = 0; $j < pg_num_fields($result); $j++)
  {
                $fieldname = pg_field_name($result, $j);
    echo "<th>". $fieldname . "</th>";
  }
  //Add a header for the current-value field, which is not from the database
  echo "<th>Current value(SEK)</th>";
  echo "</tr>";

  //Add all the rows with the items from the database
for ($row = 0; $row < pg_numrows($result); $row++) {
    $values = pg_fetch_row($result, $row);
    $curr_date = null;
    $curr_dep = null;
    $curr_price = null;
    ?>
    <!-- Make the row highlighted on mouse-over and hyper-linked -->
    <tr <? if($_SESSION['phplogin']){ echo " onClick=\"document.location.href='manage_item.php?item=".$values['0']."&table=ITEM';\" "; }
?> class="normal" onMouseOver="this.className='highlight'" onMouseOut="this.className='normal'">
    <?
    for ($col = 0; $col < count($values); $col++) {
      if($values[$col] == null) { echo "<td> - </td>"; }
      else {
                                                echo "<td>";
                                                //If the if-statement is true, we are in the first column and should add the
hyper-linked printer-icon
                                                if($col == 0 && $_SESSION['phplogin'])
                                                {
                                                        echo "<a href=print_label.php?item=".$values[$col]."
title=\"Print lable for this item\"><img src=\"print_label.jpg\" align=right></a>";
                                                        echo "</td>";
                                                        echo "<td>";
                                                        echo  $values[$col];
                                                }
                                                //If the if-statement is true, the field is a foreign key
                                                else if(isFkey(pg_field_name($result, $col), $fkeys)!= (-1)){
                                                        // Print the TITLE-field from the table containing the
foreign key
                                                        // instead of the ID-field as stored in the current table
                                                                $fvalues = pg_fetch_row($fkeys,
isFkey(pg_field_name($result, $col), $fkeys));
```

```
                                                                                $fkeyresult = pg_exec($db_handle,
"SELECT \"".$fvalues[2]."\".\"TITLE\" FROM assetmanagement.\"".$fvalues[2]."\" WHERE \"ID\" = ".$values[$col]);
                                                                                $nvalues = pg_fetch_row($fkeyresult,
0);
                                                                                echo $nvalues[0];
                                                                                 }
                                                        else{
                                                                        echo $values[$col];
                                                                        // If either of these if-statements are true, we are in one of
the fields
                                                                        // which's value is used to calculate the current price.
                                                                        // This value should be stored in a variable
                                                                        if(pg_field_name($result, $col)=="PRICE") $curr_price =
$values[$col];
                                                                        if(pg_field_name($result, $col)=="DATE") $curr_date =
$values[$col];
                                                                        if(pg_field_name($result, $col)=="DEPRECIATION")
$curr_dep = $values[$col];
                                                                        }
                                                        echo "</td>";
                                                         }
    }
                                        //Function to print a <td> with curr_price of the item
                                        //Calculates price in relation to curr_date, curr_price and curr_dep
                                        if ( $curr_date != null){
                                        // The runnable c-program file age_calc is executed
                                        // and the resulting printed string is stored in the $age variable
                                        // For this to work, age_calc must be located in the same folder as this php-file
                                        // and be executeable by the web-server user, "apache" by default
                                        $age = intval(shell_exec("./age_calc ".$curr_date));
                                        echo "<td>";
                                        if (($curr_price-(($curr_price/$curr_dep))*$age) > 0)
                                        {
                                                        echo intval(($curr_price-(($curr_price/$curr_dep))*$age));
                                                        $tot_price += intval(($curr_price-(($curr_price/$curr_dep))*$age));
                                        }
                                        else echo "0";
                                        echo "</td>";
                                        }
                                        else echo "<td>-</td>";
    echo "</tr>";
  }
                        if(pg_numrows($result)>0){
                        echo "<tr class=\"normal\">";
                        if($_SESSION['phplogin']) echo "<td colspan=".(count($values)+1)." align=right>Total value:";
                        else echo "<td colspan=".(count($values))." align=right>Total value:";
                        echo "<td>".$tot_price."</td>";
                        echo "</tr>";
                        }
                        echo "</table>";
}
//else: the sql-query resulted in an error
 else {
                                        //Display error pop-up with the error message from the database
                                        ?>
                                        <script type="text/javascript">
                                        alert("<? echo addcslashes(pg_errormessage($db_handle),"\"");?>")
                                        </script>
                                        <?
}
//Close the database-handle
pg_close($db_handle);
?>
 </body> </html>
```

login.php

```
<!--
*** The login-page is where the users choose wether to log on as an guest or an admin
*** A session variable ($_SESSION['phplogin']) is used on the other pages to keep track
*** of the user's role during the session
```

```
-->
<?php
session_start();
//Set the phplogin-session-variable to false, this means we're not logged in
$_SESSION['phplogin'] = false;
//If the if-statement is true, the login-button was clicked by the user
if(isset($_POST['login']))
{
$password = $_POST['pswd'];
// If the if-statement is true, the users chose to log on as a Guest
// and is redirected to the main page instantly
if($_POST['lvl'] == 0) { header('Location: list_all.php'); exit; }
// else: the user chose to log on as an admin and a password check must be performed
// if: the password is correct
else if ( $password == "exa11" ) { // exa11 is the only valid password for an admin-login, this can be changed here
   $_SESSION['phplogin'] = true;
   header('Location: list_all.php');
   exit;
}
// else: wrong password
else{
?>
<script type="text/javascript">
alert('Wrong Password, Please Try Again')
</script>
<?php
}
}
?>

<html>
<head>
<title> Log in </title>
<script>
// JavaScript-function to toggle show or hide the 'pass' element
// which is the text field where the admin-pass should be intered
function ShowMenu(num)
{
if(num == 1)
document.getElementById('pass').style.display = 'block';
else
document.getElementById('pass').style.display = 'none';
}
</script>
</head>
<body bgcolor="FF9933">
<center>
<h1>Log in</h1>
<form method="post" action="">
Login as:
<!-- Drop-down menu to chose log in role -->
<select name="lvl" id="level" onChange="javascript: ShowMenu(document.getElementById('level').value);">
<option value='0'>Guest</option>
<option value='1'>Admin</option>
</select>
<br>
<!-- Div-part for the pass-field -->
<div id="pass" style="display: none;">
Password:
<input type="password" name="pswd">
</div>

<input type="submit" name="login" value="Login">
</form>
</center>
<script language="JavaScript">
// Hide the pass-field by default
document.getElementbyId("<divid>").style.display:"hidden";
</script>
</body>
</html>
```

manage_item.php

```
<!--
*** This is the page where items in the different tables in the database can be adjusted or deleted
*** The page takes two get-parameters, one with a string-identifier for the table and one integer for the item id
-->
<?php
include 'sql_queries.php';

//If the if-statement is true,the items should be deleted
 if(isset($_POST['delete'])){
                        //If the if-statement is true, the DELETE-query was successful
  if(pg_exec($db_handle, "DELETE FROM assetmanagement.\"".$_GET['table']."\" WHERE \"ID\" = ".$_GET['item'])){
  ?>
                        <script type="text/javascript">
                                history.go(-1)
                        </script>
 <?
}
// else: the DELETE-query failed
else{
                        ?>
                        <script type="text/javascript">
                                alert("The item could not be removed!")
                        </script>
                        <?
}
}
//If the if-statement is true, the user is not logged in as an admin
if(!$_SESSION['phplogin']){
   header('Location: login.php');
   exit;
}
//If the if-statement is true, there were no get-parameters in the URL
if($_GET['item'] == null || $_GET['table'] == null ){
                        header('Location: list_all.php');
                        exit;
}

                        //Fetch data from database
                        $result = pg_exec($db_handle, get_itemquery($_GET['table'],$_GET['item']));
                        $fkeys = pg_exec($db_handle, get_fkeyquery($_GET['table']));
                        // If the if-statement is true, there were 0 rows in the result from the get_itemquery
                        // This means the name of the table or id-number of the item was invalid,
                        // or that the item no longer exists in the database
                        if(pg_numrows($result)==0){
                                ?>
                        <script type="text/javascript">
                                this.location.href = "list_all.php";
                        </script>
 <?
                        }

                        //If the if-statement is true, the user has submitted the changes made to the item
if(isset($_POST['submit'])){
                        //Base for the UPDATE-query
                        $updatequery = "UPDATE assetmanagement.\"".$_GET['table']."\" SET ";
     $values = pg_fetch_row($result, 0);
     //For-loop to loop through all the options in the form
     for ($col = 1; $col < count($values); $col++) {
                                        //Procedure to build GET-clause for the UPDATE-statement
                                                if($col > 1) $updatequery .= ", ";
                                                if($_POST[pg_field_name($result, $col)] != null) $updatequery .= "\"" .
pg_field_name($result, $col). "\" = '". $_POST[pg_field_name($result, $col)]."'";
                                                else $updatequery .= "\"" . pg_field_name($result, $col). "\" = null";
  }
  //Add the end of the UPDATE-query
       $updatequery .= " WHERE \"ID\" = ".$_GET['item'];
       //Execute the query
                        $updateresult = pg_exec($db_handle, $updatequery);
                        //If the if-statement is true, the UPDATE-query failed and retunred an error
```

```
                        if(!$updateresult){
                                        ?>
                                        <script type="text/javascript">
                                        alert("<? echo addcslashes(pg_errormessage($db_handle),"\"");?>")
                                        </script>
                                        <?
                        }

                        //else: update the item on the page to correspond to the changes
                        else $result = pg_exec($db_handle, get_itemquery($_GET['table'],$_GET['item']));
}
?>
<html>
 <head>
  <title>Manage items</title>

<!-- JavaScript-function to display a pop-up to confirm deletion of an item -->
<script type="text/javascript">
function confSubmit(form) {
if (confirm("Are you sure you want to delete this item?")) {
form.submit();
}
}
</script>

 </head>
 <body class="background">
 <center><h1>Manage item</h1></center>
<? #Functions
//Function to construct a drop-down menu for the options in a foreign-key field
//The parameters are:
// (<sql-query result with all the foreign-keys>, <which row in the result is this perticular foreign-key>, <the value for the displayed item in
this field>, <standard database handle>)
function mkDropdown($_fkeys, $_fkeyRow, $_default, $db_handle){
                        $fvalues = pg_fetch_row($_fkeys, $_fkeyRow);
                        $dropresult = pg_exec($db_handle, "SELECT * FROM assetmanagement.\"".$fvalues[2]."\"");
                        echo "\n<select name=\"".$fvalues[1]."\">";
                                        for ($row = 0; $row < pg_numrows($dropresult); $row++) {
                                                        $dvalues = pg_fetch_row($dropresult, $row);
                                                        echo "<option value=\"".$dvalues[0]."\"";
                                                        if ( $dvalues[0] == $_default) echo " selected ";
                                                        echo ">" . $dvalues[1] . "</option>";
                                        }
                                        echo "</select>\n";
}

//If the if-statement is true, the sql-query was successful
if ($result) {
  //Table presenting sql query results
  echo "<table border=\"1\">";
  echo "<tr style=\"background-color:#dd7c1f;\">\n";
  for ($j = 0; $j < pg_num_fields($result); $j++)
  {
                        //Function building all the table-headers on the first row
    $fieldname = pg_field_name($result, $j);
    echo "<th>". $fieldname . "</th>";
  }
  echo "\n</tr>";
  echo "\n<form method=\"post\" action=\"\">";
  //For-loop building the rows with the items from the sql-query (there should really only be one)
for ($row = 0; $row < pg_numrows($result); $row++) {
     $values = pg_fetch_row($result, $row);
     echo "\n<tr>\n";
     //For-loop building the cells on the row
     for ($col = 0; $col < count($values); $col++) {
                                        //If the if-statement is true, we are at a row that is not #0 (the first)
                                        if($col!=0){
                                                        echo "<td>\n";
                                                        // If the if-statement is true, the field is a foreign-key
                                                        // and a drop-down menu should be built
                                        if(($fkeyRow = isFkey(pg_field_name($result, $col),$fkeys))!=(-1)){
```

```
                                                mkDropdown($fkeys, $fkeyRow, $values[$col], $db_handle);
                                        }
                                        //else: build a text field with the item's current value in this perticular field as default-value
                                        else{
                                        ?>
                                         <input size="15" type="text" maxlength="40" name="<?echo pg_field_name($result, $col); ?>"
value="<? if($values[$col] != null) echo $values[$col]; ?>">
                                                                <?
                                                                }
                                                                echo "</td>\n";
                                                                }
                                                                //else: we are at row 0 which is the ID-field, which should not be adjustable
                                                                else echo "<td>".$values[$col]."</td>";
    }
    echo "</tr>";
  }
  echo "</table>";
  //A separate table for the Apply-button for layout simplicity
  echo "<table>";
                        echo "<tr><td><input name=\"submit\" type=\"submit\" value=\"Apply\">";
                        echo "</form>";
                        ?>
                        <!-- The form for the Delete-button -->
                        <form method="post" action="">
                        <input type="hidden" name="delete" value="delete">
                        <input onClick="confSubmit(this.form);" type="button" value="Delete"> </form>
                        <?
                        echo "</td></tr>";
                        echo "</table>";
}
//else: the query to get the current item from the database failed
else {
  echo "The query <br>". get_itemquery($_GET['table'],$_GET['item'])  ."<br> failed with the following error:<br>\n";
  echo pg_errormessage($db_handle);
}
//Close the database handle
pg_close($db_handle);
?>
 </body>
</html>
```

print_label.php

```
<?php
/* This page will print a label with the value of the $_GET-parameter in the url
 * using a shell-command.
 * */

session_start();

//If the if-statement is true, the user is logged in as guest and does not have access to this page
if(!$_SESSION['phplogin']){
   header('Location: login.php');
   exit;
}
//If the if-statement is true, the needed $_GET-parameter is not defined in the URL; redirecting
if($_GET['item'] == null ){
                        header('Location: list_all.php');
                        exit;
}
//shell_exec-commands are executed as if it were entered in the linux-terminal by the same user as the apache server ("apache" by
default)

//First command creates an png-image of the QR-code
//This command will not work unles the library "qrencode" is installed properly on the server
//The data encoded in the QR-code is a http-link to the item.php-page on the web site with the ID
//of the item as a GET-parameter. This will allow the user scanning the bar code to open the page
//in the default web browser on the phone with most common bar code scanning apps.
shell_exec("qrencode -o ".$_GET['item'].".png -margin=0 \"http://ctlams/item.php?item=".$_GET['item']."\" -s 3");

//Second command sends the QR-code image to the printer
//To change printer location adjust the -d option in the shell_exec string below.
//The IP-address should be that of the computer the printer is connected to.
//example: "lp -d QL-500@<ip-address of the computer with the printer> -o media=24Dia ".$_GET['item'].".png"
shell_exec("lp -d QL-500@192.168.146.129 -o media=24Dia ".$_GET['item'].".png");

//Third command removes the png-image that was printed
//shell_exec("rm ".$_GET['item'].".png -f");

?>
<html>
 <head>
  <title>Printing...</title>
  <script>
  // JavaScript to redirect to the previous page,
  // if this code works as intended this entire php file
  // should be executed in practically no time and the user
  // returned to where the print-link was used
  javascript: history.go(-1)
  </script>
 </head>
</html>
```

select_table.php

```
<!--
*** This page displays all the items in the different tables, without any filtering-options as in list_all.php
*** All the tables are loaded as the page is accessed by a client, but are hidden on the page with JavaScript
*** When the user clicks one of the cells containing the different table names the corresponding table,
*** with its containing items, is displayed. Only one table is displayed at a time.
-->
<?php
include 'sql_queries.php';
//If the if-statement is true, the user is a "Guest" and does not have access to this page; redirecting
if(!$_SESSION['phplogin']){
   header('Location: login.php');
   exit;
}
                        //Fetch data from database
                        $tables = pg_exec($db_handle, get_tablesquery());
?>
<html>
 <head>
  <title>Select table</title>

<script type="text/javascript">
//Variables to store data which decides the behaviour of this function
var state = 'none';
var currShow = 'none';
                        //Function to toggle visability for the tables
                        function showhide(layer_ref) {
                                        //if layer ref == currShow hide layer ref, set currshow=0 then break
                                        if (currShow == layer_ref) {
                                                        state = 'none';
                                                        currShow = 'none';
                                        }
                                        //if layer ref != currshow hide currshow, show layer ref, set currshow=layer ref then break
                                        else if (currShow != 'none') {
                                                        state = 'none';
                                                        document.getElementById(currShow).style.display = state;
                                                        state = 'block';
                                                        currShow = layer_ref;
                                        }
                                        else {
                                                        state = 'block';
                                                        currShow = layer_ref;
                                        }
                                        document.getElementById(layer_ref).style.display = state;
                        }
</script>
 </head>
 <body class="background">
 <center><h1>All tables</h1></center>
<? #Functions
                        //Table with JavaScript-calls to show all forms
                        echo "<table border=\"1\" bgcolor=\"white\">";
                        echo "<tr class=\"normal\">";
                        for ($tablesRow = 0; $tablesRow < pg_num_rows($tables); $tablesRow++)
                        {
                                        $currTable = pg_fetch_row($tables, $tablesRow);
                                        echo "<td onMouseOver=\"this.className='highlight'\" onMouseOut=\"this.className='normal'\"
onclick=\"showhide('".$currTable[0]."');\">".$currTable[0]."</td>";
                        }
                        echo "\n</td>";
                        echo "\n</tr>";
                        echo "\n</table>";


                        //For-loop to add <table> for each table
                        for ($tablesRow = 0; $tablesRow < pg_num_rows($tables); $tablesRow++)
                        {
                                        $currTable = pg_fetch_row($tables, $tablesRow);
                                        $query = "SELECT * FROM assetmanagement.\"".$currTable[0]. "\" ORDER BY
                                        \"".$currTable[0]."\".\"ID\" ASC";
```

```
                                        $result = pg_exec($db_handle, $query);
                                        $fkeys = pg_exec($db_handle, get_fkeyquery($currTable[0]));


                                        //Table presenting existing items, the <div> tag is for easy toggling of visability
                                        echo "\n<div id=\"".$currTable[0]."\" style=\"display: none;\">";
                                        echo "<table bgcolor=\"white\" border=\"1\">";
                                        echo "<tr class='highlight'><th colspan=".pg_num_fields($result).">";
                                        echo $currTable[0]." (<a href=\"add.php?table=".$currTable[0]."\">add new</a>)\n";
                                        echo "</th></tr>\n";
                                        echo "<tr class='highlight'>";
                                        //For-loop to build the table headers on the first row
                                        for ($j = 0; $j < pg_num_fields($result); $j++)
                                        {
                                                $fieldname = pg_field_name($result, $j);
                                                echo "<th>". $fieldname . "</th>";
                                        }
                                        echo "</tr>\n";
                                        //For-loop to add every item in the table on a new row
                                        for ($row = 0; $row < pg_numrows($result); $row++) {
                                                $values = pg_fetch_row($result, $row);
                                                //Table-row tag that makes the row highlighted on mouse-over and hyper-link
to manage_item.php
                                                echo "<tr
onClick=\"document.location.href='manage_item.php?item=".$values[0]."&table=".$currTable[0]."';\" class='normal'
onMouseOver=\"this.className='highlight'\" onMouseOut=\"this.className='normal'\">";
                                                //For-loop to add all cells on the row
                                                for ($col = 0; $col < count($values); $col++) {
                                                        // If the if-statement is true, the value for this field is null
according to the database,
                                                        // "-" is printed in this cell for layout reasons
                                                        if($values[$col] == null) { echo "\n\t<td> - </td>"; }
                                                        else {
                                                                echo "\n\t<td>";
                                                                // If the if-statement is true, the field is
a foreign-key and the title-value
                                                                // from the corresponding table should
be displayed instead of the id-value
                                                                if(isFkey(pg_field_name($result, $col),
$fkeys)!= (-1)){
                                                                        $fvalues =
pg_fetch_row($fkeys, isFkey(pg_field_name($result, $col), $fkeys));
                                                                        $fkeyresult =
pg_exec($db_handle, "SELECT \"".$fvalues[2]."\".\"TITLE\" FROM assetmanagement.\"".$fvalues[2]."\" WHERE \"ID\" = ".$values[$col]);
                                                                        $nvalues =
pg_fetch_row($fkeyresult, 0);
                                                                        echo $nvalues[0];
                                                                }
                                                                else echo $values[$col];
                                                                echo "</td>";
                                                        }
                                                }
                                                echo "\n</tr>\n";
                                        }
                                        echo "</table>";
                                        echo "\n</div>";
                        }
                        //Close the database handle
pg_close($db_handle);

?>
 </body>
</html>
```

sql_queries.php

```
<?php
/* This file contains functions and code that is re-used in many of the PHP-files on the web site.
 * This includes, the standard database handle, the quick-link menu at the top of each page,
```

```
 * functions to build some frequently used sql-queries and the basic CSS-styles.
 * */

// Sandard database handle, if the postgresql database is ever moved to
// another server which is not hosting the apache server, this ip-address should be changed
$db_handle = pg_connect("dbname=postgres host=127.0.0.1
user=postgres");

//$parts will contain the URL to this page split by every "/"
$currentFile = $_SERVER["PHP_SELF"];
$parts = Explode('/', $currentFile);

//Start the php-session, this is for the $_SESSION-variables
session_start();
?>
<!-- Styles defining the different colors used on the page, these can be changed to adjust the general color-scheme -->
<STYLE>
  .normal { background-color: #eeb35a }
  .highlight { background-color: #dd7c1f }
  .background { background-color: #FF9933 }
</style>
<?
//If the if-statement is true, the users is logged in as admin
if($_SESSION['phplogin'] == true){
?>
<!--
*** This is the table with the quick-link menu displayed at the top of almost all pages on the web site
*** The if-statements are to decide wether the link is to the current page.
*** If it is, the cell is not hyper-linked and will be displayed in another color.
*** (This menu is only displayed if the user is logged in as an admin)
-->
<table  border="1" bgcolor="white">
                        <tr class='normal'>
                                        <? if($parts[count($parts) - 1] == "list_all.php"){ ?>
                                        <th class='highlight'>All items</th>
                                        <?}
                                        else{ ?>
                                        <th onClick="document.location.href='list_all.php';" onMouseOver="this.className='highlight'"
onMouseOut="this.className='normal'">All items</th>
                                        <? } ?>

                                        <? if($parts[count($parts) - 1] == "select_table.php"){ ?>
                                        <th class='highlight'>All tables</th>
                                        <?}
                                        else{ ?>
                                        <th onClick="document.location.href='select_table.php';"
onMouseOver="this.className='highlight'" onMouseOut="this.className='normal'">All tables</th>
                                        <? } ?>

                                        <? if($parts[count($parts) - 1] == "add.php"){ ?>
                                        <th class='highlight'>Add new item</th>
                                        <?}
                                        else{ ?>
                                        <th onClick="document.location.href='add.php?table=ITEM';"
onMouseOver="this.className='highlight'" onMouseOut="this.className='normal'">Add new item</th>
                                        <? } ?>

                                        <? if($parts[count($parts) - 1] == "help.php"){ ?>
                                        <th class='highlight'>Help</th>
                                        <?}
                                        else{ ?>
                                        <th onClick="document.location.href='help.php';" onMouseOver="this.className='highlight'"
onMouseOut="this.className='normal'">Help</th>
                                        <? } ?>

                                        <th onClick="document.location.href='login.php';" onMouseOver="this.className='highlight'"
onMouseOut="this.className='normal'">Log out</th>
                        </tr>
</table>

<?
```

```
}
//else: the user is logged in as a guest, the only link in the menu is to the log in page
else{
                        ?>
<table  border="1" bgcolor="white">
                        <tr style="background-color:#eeb35a;">
                                        <th onClick="document.location.href='login.php';">Login as admin</th>
                        </tr>
</table>
<?

}
                        //Function to find out wether a field is a foreign-key
                        function isFkey($field, $_fkeys){
                                        $_return = -1;
                                        for($row = 0; $row < pg_numrows($_fkeys); $row++) {
                                                        $values = pg_fetch_row($_fkeys, $row);
                                                        if($values[1] == $field) $_return = $row;
                                        }
                                        return $_return;
                        }

//Function that returns the standard sql-query as a string. The query was built with help from the pgAdminIII graphical sql-query tool
function get_standardquery(){
                        return "SELECT
 \"ITEM\".\"ID\",
 \"ITEM\".\"MODEL\",
 \"ITEM\".\"NAME\",
 \"ITEM\".\"QUANTITY\",
 \"ITEM\".\"ARTICLENUMBER\",
 \"ITEM\".\"SERIALNUMBER\",
 \"ITEM\".\"PRICE\",
 \"ITEM\".\"DEPRECIATION\",
 \"STORAGELOCATION\".\"TITLE\" AS \"STORAGELOCATION\",
 \"SUBPROJECT\".\"TITLE\" AS \"SUBPROJECT\",
 \"PROJECT\".\"TITLE\" AS \"PROJECT\",
 \"TYPE\".\"TITLE\" AS \"TYPE\",
 \"BUILDING\".\"TITLE\" AS \"BUILDING\",
 \"ROOM\".\"TITLE\" AS \"ROOM\",
 \"ITEM\".\"DATE\"
FROM
 assetmanagement.\"ITEM\",
 assetmanagement.\"PROJECT\",
 assetmanagement.\"SUBPROJECT\",
 assetmanagement.\"TYPE\",
 assetmanagement.\"ROOM\",
 assetmanagement.\"BUILDING\",
 assetmanagement.\"STORAGELOCATION\"
WHERE
 \"ITEM\".\"PROJECT\" = \"PROJECT\".\"ID\" AND
 (\"ITEM\".\"STORAGELOCATION\" = \"STORAGELOCATION\".\"ID\" OR
 \"ITEM\".\"STORAGELOCATION\" = null) AND
 (\"ITEM\".\"SUBPROJECT\" = \"SUBPROJECT\".\"ID\" OR
 \"ITEM\".\"SUBPROJECT\" = null) AND
 (\"ROOM\".\"BUILDING\" = \"BUILDING\".\"ID\" OR
 \"ROOM\".\"BUILDING\" = null) AND
 (\"STORAGELOCATION\".\"ROOM\" = \"ROOM\".\"ID\" OR
 \"STORAGELOCATION\".\"ROOM\" = null) AND
 \"ITEM\".\"TYPE\" = \"TYPE\".\"ID\"";
                        }

//Function returning the base for the sql-query to get all items from a table as a string
function get_itemquery($table, $item){
                        return "SELECT * FROM assetmanagement.\"".$table."\" WHERE \"".$table."\".\"ID\" = ".$item;
}

//Function returning the sql-query to get a list of all the foreign keys for a certain table in the database as a string.
//The query will also return information of which table and field the keys link to
function get_fkeyquery($table){

                        return "SELECT        tc.table_name,
```

```
                kcu.column_name,

                ccu.table_name AS references_table,
                ccu.column_name AS references_field
                FROM information_schema.table_constraints tc

                LEFT JOIN information_schema.key_column_usage kcu
                ON tc.constraint_catalog = kcu.constraint_catalog
                AND tc.constraint_schema = kcu.constraint_schema
                AND tc.constraint_name = kcu.constraint_name

                LEFT JOIN information_schema.referential_constraints rc
                ON tc.constraint_catalog = rc.constraint_catalog
                AND tc.constraint_schema = rc.constraint_schema
                AND tc.constraint_name = rc.constraint_name

                LEFT JOIN information_schema.constraint_column_usage ccu
                ON rc.unique_constraint_catalog = ccu.constraint_catalog
                AND rc.unique_constraint_schema = ccu.constraint_schema
                AND rc.unique_constraint_name = ccu.constraint_name

                WHERE lower(tc.constraint_type) in ('foreign key')
                AND tc.table_name = '".$table."'";
}

//Function returning the sql-query to get all the names of the tables in the user-defined part of the database as a string
function get_tablesquery(){

                return "SELECT
                                                distinct columns.table_name
                                                FROM
                                                information_schema.columns
                                                WHERE
                                                columns.table_schema = 'assetmanagement'";
}
?>
```

time_compare.c

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
int to_seconds(const char *date)
{
                                        char now[80];
                                        struct tm *ts;
    int retval=0;
                                        time_t d2;
                                        d2 = time(0);
                                        ts = localtime(&d2);

                                        strftime(now, 80, "%Y-%m-%d", ts);
    retval += ((atoi(&now[0])) - atoi((date+(sizeof(char)*0))))*12;
    retval += ((atoi(&now[5])) - atoi(date+(sizeof(char)*5)))*1;
    if( atoi(&now[8]) < atoi(date+(sizeof(char)*8))) retval--;

    return retval;
}

int main(int argc, char *argv[])
{
                int d1=to_seconds(argv[1]);
                printf("%d",d1);
                return 0;
}
```