Utveckling av slutgranskning i produktionsprocessen

Fler användningsmöjligheter i slutgranskningsprogrammet Quality Review

Mathias Mäntysaari

Examensarbete för ingenjör (YH)-examen

Utbildningsprogrammet för informationsteknik

Vasa 2012

EXAMENSARBETE

Författare:	Mathias Mäntysaari
Utbildningsprogram och ort:	Informationsteknik, Vasa
Handledare:	Kaj Wikman

Titel: *Utveckling av slutgranskning i produktionsprocessen*

Datum 9.4.2012 Sidantal 63

Sammanfattning

Detta examensarbete har utförts åt karosserifabriken Oy Närko Ab i Närpes. Målet med arbetet var att införa fler användningsmöjligheter i den applikation som används vid slutgranskningen av nyproducerade vagnar vid företaget. Arbetet bestod i huvudsak av att införa användande av kamera vid granskningen samt kontroll av gjord axeljustering i produktion. En applikation för operativsystemet Android utvecklades för att använda kameran i en smarttelefon. Den befintliga webbapplikationen gjord i programmeringsspråket PHP byggdes ut och integrerades i den nya Android-applikationen. Resultatet av arbetet blev en applikation där slutgranskarna vid kvalitetsgranskningen kan fotografera vagnar före leverans. I applikationen har det blivit lätt att komma åt protokollet över gjord axeljustering, för att se att den blivit gjord och för kontroll av att de justerade värdena är inom angivna toleranser.

Språk: svenska Nyckelord: Android, PHP, slutgranskning

Förvaras: Theseus.fi och Tritonia, Vasa vetenskapliga bibliotek

OPINNÄYTETYÖ

Tekijä:	Mathias Mäntysaari
Koulutusohjelma ja paikkakunta:	Tietotekniikka, Vaasa
Ohjaaja:	Kaj Wikman

Nimike: *Laaduntarkastuksen kehitys tuotannossa*

9.4.2012 63 sivua

Tiivistelmä

Tämä opinnäytetyö on suoritettu koritehdas Oy Närko Ab:lle, Närpiössä. Työn tavoite oli lisätä käyttömahdollisuuksia sovelluksessa, jota käytetään uusien vaunujen laaduntarkastuksessa. Työn pääpaino oli kameran käyttöönotto laaduntarkastuksessa, erityisesti akselien säätöjen osalta tuotannossa. Sovellus kehitettiin Android-käyttöjärjestelmää varten, joka mahdollisti älypuhelimen kameran käytön. Nykyistä nettisovellusta, joka on ohjelmoitu PHP:llä laajennettiin ja se integroitiin uuteen Android-sovellukseen. Työn lopputulos on uusi sovellus, jonka avulla tarkastajat laaduntarkastuksessa voivat kuvata vaunuja ennen toimitusta. Sovelluksen myötä on entistä helpompi löytää akselisäätöjen mittapöytäkirja, jotta tarkastajat voivat todeta, että säädöt on tehty ja säädetyt arvot ovat toleranssien sisällä.

Kieli: ruotsi Avainsanat: Android, PHP, laaduntarkastus

BACHELOR'S THESIS

Author:	Mathias Mäntysaari
Degree programme:	Information Technology, Vaasa
Supervisor:	Kaj Wikman

Title: *Improvements of Quality Control in the production process*

Date 9.4.2012 Number of pages 63

Abstract

This thesis work was made for Oy Närko Ab, a transport vehicle manufacturer in Närpes, Finland. The purpose of the assignment was to introduce new possibilities of usability in the application used for Quality Control of the company's brand new produced trailers. The assignment essentially consisted of introducing the use of a camera in the Quality Control and verifying that the axis adjustment had been performed during production. An application was developed to make it possible to use the camera of Android smart phones. The currently used web application programmed in PHP was developed to be integrated into the new Android application. The result of this work is an application where quality controllers can take photographs of trailers before delivery. In the application it has also become easy to access measurement records of performed axis adjustments, to verify that it has been done and to control that the values are within specified tolerances.

Language: Swedish Key words: Android, PHP, Quality Control

Innehållsförteckning

Förkortningar och begrepp

API	Application Programming Interface. Källkod som innehåller regler för hur en programvara kan kommunicera med en annan. Beskriver funktionalitet utan att berätta hur den gjorts.
Bugg	Ett fel som finns i programmets källkod.
CGI	Common Gateway Interface. Protokoll för webbservrar som används för att delegera genereringen av webbsidor till exekverbara skript.
Debugging	Används vid testning av program för att spåra och avlägsna fel i programvaran.
Emulator	Hårdvara eller mjukvara avsedd att efterlikna funktionen av annan mjukvara eller hårdvara. Används ofta för att testa programvara på hårdvara som kan vara dyr eller svår att få tag på.
Encoding	Översätts som teckenkodning på svenska. Ett sätt att representera en vald uppsättning tecken när man vill överföra text på tekniskt sätt. Vanliga exempel på encoding är ASCII och Unicode.
ERP-system	Enterprise Resource Planning. Ett IT-system där ett företags information hanteras. Innehåller vanligen moduler för kund-, logistik-, inköps- och personaladministration.
Exekvera	Innebär att ett program körs på en dator.
IIS	Internet Information Services. Webbserverprogramvara för datorer som kör operativsystemet Microsoft Windows.
Interface	Utformningen av kommunikation mellan olika mjukvarumoduler och/eller hårdvarumoduler.
JSON	JavaScript Object Notation. Ett format för att överföra data på ett snabbt och enkelt sätt. Formatet innehåller mycket litet överflödig information, såkallad overhead. Läs mer i kapitel 3.3.3.

Klass	Programkod som samlar en mängd relaterade attribut och funktioner. En användardefinierad datatyp.
Kompilering	Innebär att datorn översätter programkod från ett språk till ett annat. Vanligtvis översätts koden till en lägre nivå, så kallad maskinkod.
Källkod	Den kod som bygger upp ett program.
Objekt	Ett objekt är en instans av en klass. När man använder en klass som t.ex. representerar en bok så har den attribut som titel, författare o.s.v.
Objektorienterat	En programmeringsmetod som bygger på användning av objekt som interagerar med varandra. Gör det lättare att återanvända funktionalitet mellan olika program.
ODBC	Open DataBase Connectivity. En standardiserad åtkomstmetod för databaser. Målsättningen med ODBC är att applikationer ska kunna komma åt data oavsett vilken databashanterare eller operativsystem som används. Läs mer i kapitel 3.3.2.
PERL	Ett skriptspråk som vanligen används tillsammans med protokollet CGI för webbutveckling.
PHP	PHP: HyperText Preprocessor. Ett skriptspråk baserat på öppen källkod som främst används för att driva webbsidor med dynamiskt innehåll. Läs mer i kapitel 3.1.4.
Primitiva datatyper	Datatyper som programmeringsspråket erbjuder direkt, och inte klarar sig utan. De privitiva datatyperna är t.ex. heltal (int), flyttal (double) och booleska värden (bool).
Pseudo-klass	Används i CSS för att bestämma specialeffekter på vissa HTML-element som reagerar när man gör vissa saker som berör elementet.
Quality Review	Namnet på det slutgranskningsprogram som används inom Oy Närko Ab. Programmet är en webbapplikation kodad i PHP.
UML	Unified Modeling Language. En serie olika diagram som kan användas för att beskriva och designa speciellt objektorienterad programvara.

Unicode	Teckenkodningssystem som används inom datakommunikation för att representera text. Ersätter i många fall det vanligt förekommande ASCII.
URL	Uniform Resource Locator. Webbadress på svenska, används för att identifiera en webbsida och hur den kan hittas.
UTF-8	Åtta bitars teckenencoding som används för att representera text kodad i Unicode. Text representeras som en sekvens av byte.
Vektor	Ett begrepp inom programmering som avser en lista med en serie av värden.
Widget	Innebär de grafiska komponenterna som visas i gränssnittet, t.ex. etiketter, textrutor och knappar. Kan även kallas view i Android.
Viewgroup	Används för att strukturera användargränssnitt och gruppera widget-komponenter i förhållande till varandra.
WiFi	En teknik för trådlösa nätverk. Ett annat motsvarande begrepp är WLAN som står för Wireless Local Area Network.
XML	eXtensible Markup Language. Ett utbyggbart märkspråk där användaren kan definiera vilka element och attribut som anges. Används för att utbyta data mellan olika informationssystem.

1 Inledning

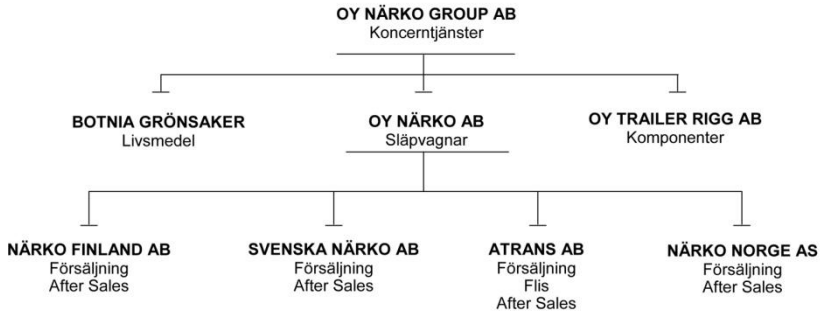
Detta examensarbete utfördes på uppdrag av företaget Oy Närko Ab, i fortsättningen enbart Närko. Kontaktpersoner vid företaget var kvalitetschef Ann-Cathrine Lassas och IT-konsult Kenneth Gullans. Som handledare fungerade Kaj Wikman vid Yrkeshögskolan Novia.

Närko har sedan år 2006 använt handdatorer som hjälp vid slutgranskningen av sina produkter. Som ett steg i kvalitetscertifieringen ska processerna regelbundet förbättras. Ett förbättringsförslag med fotografering hade diskuterats tidigare men nu kändes tiden mera mogen. Mitt examensarbete blev därför att förbättra och införa fler användningsmöjligheter i slutgranskningsprogrammet.

1.1 Företaget

Närko Group koncernen grundades i Närpes av Egil Gullström år 1959. Egil Gullström ägde sedan tidigare Gullström Grönsaker som idag heter Botnia Grönsaker. Ännu idag ägs koncernen helt av familjen Gullström. Företagets huvudkontor och största delen av produktionen är koncentrerad till Närpes. Förutom i Finland har Närko även verksamhet i Sverige, Norge, Danmark, Tyskland, Ryssland, Island och Baltikum.

Närko Group har totalt cirka 280 anställda som arbetar i något av koncernens företag. I skrivande stund fungerar Hannu Louhi som verkställande direktör. I figur 1 ses en överblick av organisationsstrukturen inom Närko Group. (NOMS - Närko Operational Management System, u.d.)



Figur 1. Organisationsstruktur inom Oy Närko Group Ab.

1.1.1 Oy Närko Ab

Oy Närko Ab är koncernens släpvagnstillverkare. Släpvagnstillverkningen är koncernens huvudverksamhet och därmed den största enskilda arbetsgivaren inom koncernen. Närko utvecklar, tillverkar och marknadsför släpvagnar och bilpåbyggnader. Marknadsföring, reservdelsförsäljning och service sköts i de nordiska länderna av dotterbolag till Närko. I Finland sköts detta av Närko Finland, i Sverige av Svenska Närko och i Norge av Närko Norge. Närko har utöver dessa även kontraktsverkstäder som kan reparera Närkos produkter. I Norge sköts servicen endast av kontraktsverkstäder. Närko äger även Atrans i Sverige som tillverkar flisvagnar. Till Atrans levereras färdiga chassier som det därefter monteras flisskåp på.

Tillverkningen sker i Närpes där Närkos ca 30 000 m^2 stora produktionshall finns. I fabriken arbetar ungefär 150 personer med allt från att rita och planera till att slutligen montera delarna på en färdig produkt. Närko är i dagsläget en av Nordens ledande tillverkare av släpvagnar. Närkos logotyp som kan ses i figur 2 är därför en vanlig syn, speciellt på de nordiska vägarna. (NOMS - Närko Operational Management System, u.d.)

Figur 2. Närkos logotyp

1.1.2 Oy Trailer Rigg Ab

Trailer Rigg utvecklar, tillverkar och distribuerar skräddarsydda komponenter för släpvagnar och bilpåbyggnader. Trailer Rigg inledde sin verksamhet redan år 1909. På 1960-talet flyttades en del av komponenttillverkningen från Närko över till Trailer Rigg.

Produktionen av stålkonstruktioner, dörrmontering och sidolämmar finns i Närpes. I Teuva tillverkas element för skåppåbyggnader samt i Kaskö kapell, flaklås, surrningsöglor och andra lastsäkringskomponenter. Trailer Rigg levererar förutom till Närko även lösningar till ett flertal andra företag

inom transport- och metallbranschen. (NOMS - Närko Operational Management System, u.d.)

Figur 3. Trailer Riggs logotyp

1.1.3 Botnia Grönsaker Ab

Gullströms Grönsaker grundades 1957 då Egil Gullström började handla med trädgårds- och jordbruksprodukter vid sin hemgård. Verksamheten går idag under namnet Botnia Grönsaker och omfattar i huvudsak partihandel med tomat, gurka, potatis och morötter. Botnia Grönsaker har två verksamhetspunkter. I Närpes sköts sorteringen, förpackning och vidareförädlingen av tomat, gurka och andra grönsaker. I Lappfjärd hanteras potatisprodukterna som säljs under namnet Pom-Frit. Under högsäsong har företaget drygt 30 anställda. Marknadsområdet är främst Finland där de största kunderna är Inex Partners och Kesko. (NOMS - Närko Operational Management System, u.d.)

Figur 4. Botnia Grönsakers logotyp

2 Bakgrund och uppgiftsbeskrivning

2.1 Bakgrund

Närko vill att deras kunder ska kunna lita på att de levererade produkterna håller en jämn och hög kvalitet. Slutgranskningsprogrammet Quality Review används vid slutgranskningen av produkterna som sista steg i produktionsprocessen för att säkerhetsställa detta. Varje producerad enhet granskas enligt ett på förhand uppgjort protokoll med programmet Quality Review, före leverans till kund. Slutgranskaren kontrollerar främst funktion och trafiksäkerhet med hjälp av programmet. Från programmet fås rapporter såsom de vanligaste bristerna och statistik över dessa, e-postmeddelanden om granskningsresultatet och diverse mätare.

2.2 Produktionsprocessen

När en vagn planerats färdigt med ritningar, alla komponenter har köpts in och en tidsplan fastställts, påbörjas produktionen av vagnens chassi. Denna process som tar vid innehåller vanligtvis sex steg. Antalet steg kan variera beroende på om underleverantörer står för något av produktionsstegen.

Som första steg börjar komponenter för vagnens ram att tillverkas. När ramkomponenterna är färdiga börjar de svetsas ihop till en ramkonstruktion. När själva ramen är klar sätts den antingen i lager eller går direkt vidare till Närkos produktionslinje Masterline. Masterline togs ibruk i mars 2004 och där utförs arbetsmomenten som leder till ett färdigt släpvagnschassi. Vissa av arbetsmomenten på Masterline är robotiserade medan andra kräver manuell montering. Vid full produktionskapacitet kan ett färdigt chassi lämna produktionslinjen med två timmars mellanrum.

Som första steg på Masterline utförs ytbehandling. Vagnen blästras först av en automatiserad robot. När blästringen slutförts grundmålas och ytbehandlas vagnen, dessa steg är manuella. Därefter när ramen torkat monteras vagnens el- och bromsutrustning samt axlar och däck. För att förbättra rostskyddet görs till sist en Dinitrol-behandling av alla hålutrymmen i ramen.

Nu har man ett färdigt chassi som kan lagras i en chassibuffert eller så går det direkt vidare till monteringen av överbyggnad. Beroende på vilken typ av vagn det är fråga om går chassiet antingen vidare till flak- eller skåpavdelningen. Vagnens flak- eller skåputrustning monteras på chassiet.

2.2.1 Axeljustering och kontroll

Inom Närkos produktion förekommer i huvudsak två fabrikat av axlar, BPW och SAF. Fabrikaten har hos Närko specificerade felgränser för snedställning och plogning ”toe”. Toe beskriver hur hjulen är riktade, ”toe-out” utåt från vagnen eller ”toe-in” inåt mot vagnen. Felgränserna bygger på däckleverantörernas rekommendationer och axeltillverkarnas uppnåeliga tillverkningstoleranser.

Axeljusteringen utförs som sista moment i slutmonteringen av montörer på avdelningarna flak eller skåp. Justeringen av axlarna sker med olastad släpvagn där axeln är i körläge på en plan yta. Efter gjord justering kontrollmäts värdena. Snedställningen är det enda som går att påverka vid axeljusteringen på Närko. Förekommer andra fel som inte går att påverka med justering av snedställningen behöver inköparen kontaktas som kontaktar respektive leverantör.

2.2.2 Slutgranskning

Slutgranskningen eller kvalitetsgranskningen är det sista steget i produktionsprocessen där man kontrollerar att vagnen är klar för leverans. För att garantera detta kontrolleras vagnens trafiksäkerhet och övriga brister som uppkommit vid monteringen. Närkos slutgranskare ska alltså med kundens kritiska ögon granska och godkänna produkten innan den levereras.

I slutskedet av vagnens montering, installeras beroende på vagnens typ flak- eller skåputrustning. När vagnen är färdigt monterad, körs den ut på gården, där den står tills den ska granskas. Alla vagnar har ett planerat datum för slutgranskningen. Detta datum baserar sig på vagnens leveransdatum. När vagnen slutligen tas in för slutgranskning körs den in på en servicegrop så att man har möjlighet att kontrollera bl.a. bromsar och

fjädring. Här utförs även programmering av ABS- och EBS-bromssystemen.

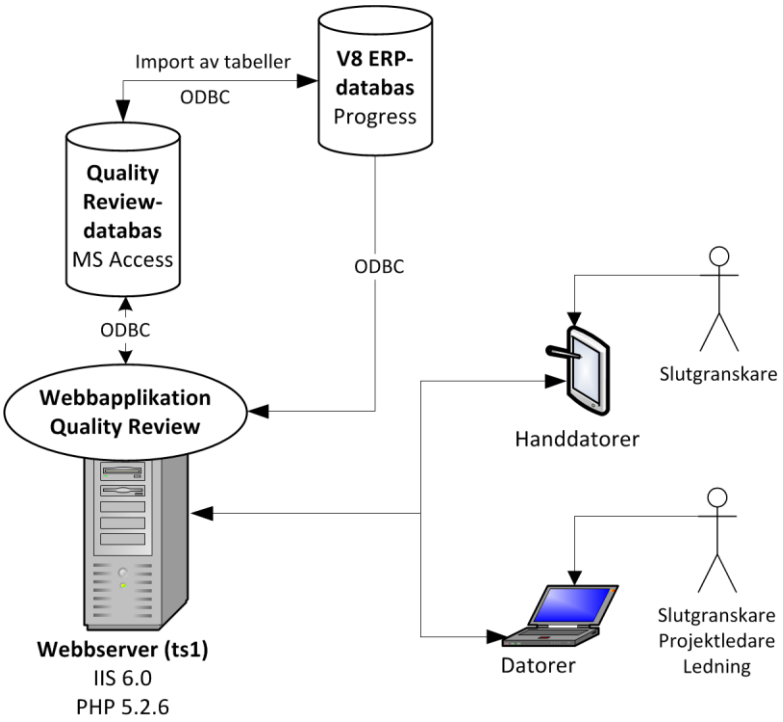
Det första personalen gör när vagnen kommer in är att leta reda på VIN-koden. VIN-koden identifierar vagnen, bland annat ser man tillverkningsland, och vagnens chassinummer eller så kallade vagnsnummer framgår sist i koden. Slutgranskaren anger vagnsnumret i webbapplikationen Quality Review i sin handdator eller alternativt i en bärbar dator. Handdatorn är kopplad till det interna nätverket via WLAN och den bärbara datorn med Ethernet. Uppgifter om vagnen hämtas därefter från ERP-systemet och granskaren kontrollerar att uppgifterna i applikationen stämmer överens med vagnen som granskas.

När vagnsuppgifterna granskats kan slutgranskaren, enligt granskningsprotokollet, börja gå runt vagnen och se så allt är i skick och fungerar. Just i det här skedet är handdatorn extra praktisk då granskaren direkt kan mata in en brist på någon av kontrollpunkterna i programmet. När vagnen har genomgått granskningen och inga brister påträffats, går den vidare till leverans. Om brister som behöver åtgärdas hittats, körs vagnen undan, en rapport skrivs ut och delas ut till ansvarig förman på den avdelning som bristerna berör. Rapporten kan endast skrivas ut från den bärbara datorn. Avdelningarna åtgärdar bristerna och en ny granskning utförs.

2.2.3 Slutgranskningsprogrammet Quality Review

Slutgranskningsprogrammet Quality Review är en webbapplikation utvecklad i skriptspråket PHP. Detta program som används vid slutgranskning av nyproducerade vagnar är utvecklat i slutet av år 2005 och togs i bruk i början av år 2006. Applikationen körs på en server (ts1) i det interna nätverket med Windows Server 2003 och webbservern IIS version 6.0 installerad.

Figur 5 visar en grov skiss hur kommunikationen fungerar mellan programmet, databaser och användarnas datorer.



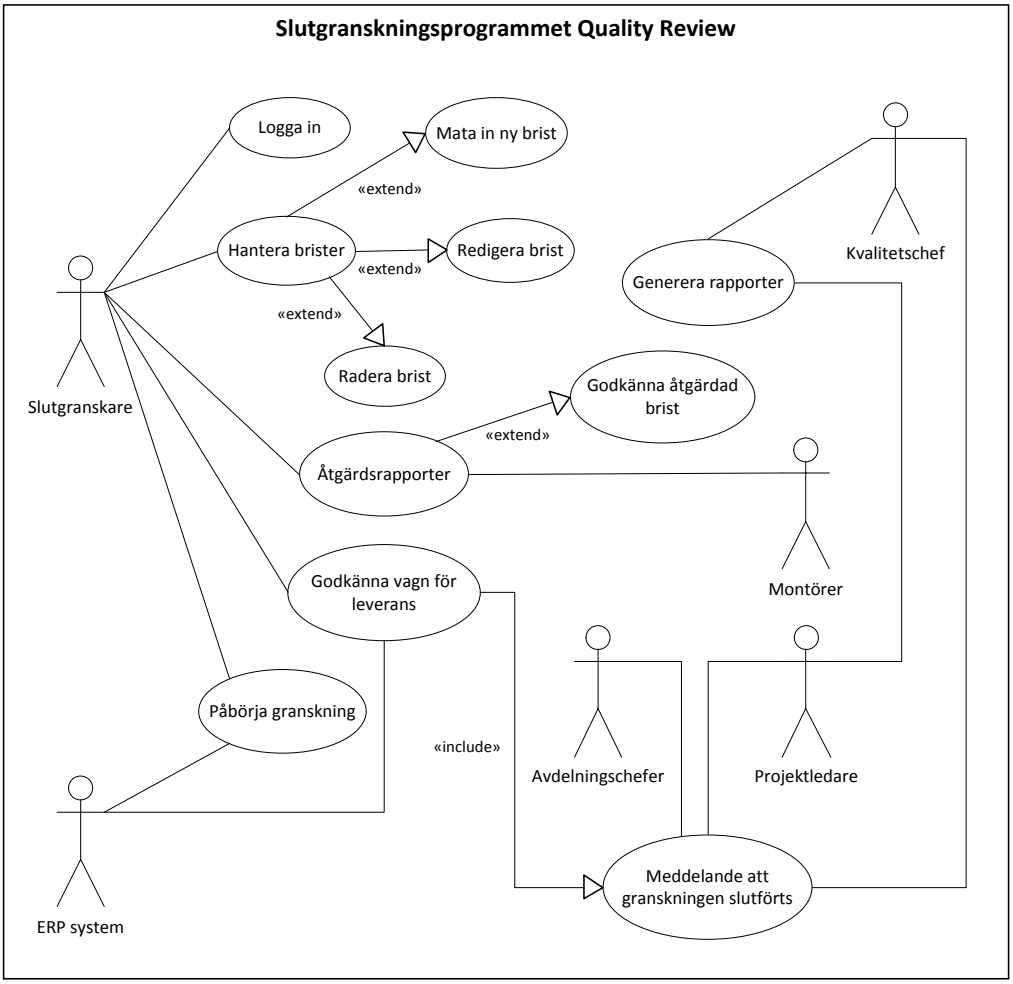
Figur 5. Grov skiss av slutgranskningsprogrammets uppbyggnad och databaskommunikation.

Programmet är kompatibelt med PHP version 5. Den specifika versionen av PHP som körs på webbservern är version 5.2.6. Denna version är flera år gammal och därför är en uppdatering planerad i samband med att examensarbetet slutförs.

Slutgranskningsprogrammet avläser och sparar information i en Microsoft Access-databas via en konfigurerad ODBC-koppling på webbservern. Vissa tabeller i Access-databasen är länkade från ERP-systemets databas med ODBC. Länkningen har gjorts för att kunna göra hämtningar från de båda databaserna smidigare. Vissa hämtningar från de länkade tabellerna via Access-databasen går långsamt. Vid testning har det visat sig att hämtningar mot ERP-systemets egen ODBC-koppling i dessa fall kan gå snabbare. Några hämtningar görs därför direkt mot ERP-systemets egen ODBC-databaskoppling. Programmet har utvecklats under årens lopp och idag används version 2.0, som inte genomgått några större förändringar under de senaste åren.

Det finns ett antal personer inom Närko som berörs av slutgranskningsprogrammet. Först och främst är det slutgranskarna som

påbörjar granskningen, matar in eventuella brister och godkänner vagnar för leverans. Beroende på om brister hittas eller inte, berörs avdelningschefer och montörer på avdelningarna som de inmatade bristerna anses ha orsakats av och bör åtgärdas av. När granskningen avslutas får avdelningscheferna, projektledaren och kvalitetschefen, ifall brister hittats, en vagnsspecifik granskningsrapport per e-post. Kvalitetschefen och projektledaren får en fullständig rapport med alla funna brister listade, medan avdelningscheferna endast får brister som berör sin egen avdelning. I figur 6 visas ett UML Use Case-diagram där det illustreras hur aktörerna, personer eller system, kommer i kontakt med programmet.



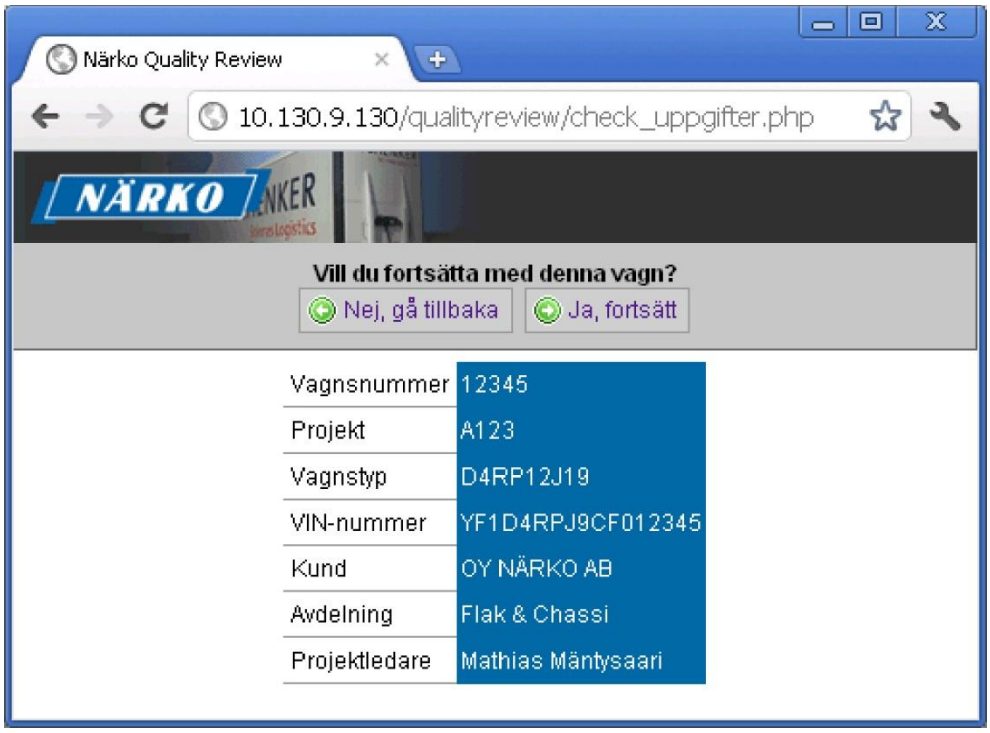
Figur 6. Use Case-diagram som visar hur olika aktörer berörs av slutgranskningsprogrammet.

När granskningen av en vagn ska påbörjas matas vagnens vagnsnummer första gången in i en textruta i programmet. Eftersom alla arbetstagare som känner till HTTP-adressen inom det interna nätverket kan visa programmet,

t.ex. för att visa rapporter, måste slutgranskaren först logga in för att få slutföra vissa steg i programmet. Dessa steg är bl.a. påbörjande av granskning, inmatning och godkännande av brister samt godkännande för leverans. Detta första steg illustreras i figur 7 där granskning är på väg att påbörjas för vagn 12345.

Figur 7. Granskning på väg att påbörjas för vagn 12345.

När granskaren tryckt på knappen ”Till granskning” görs diverse kontroller. Vagnsnumrets innehåll som bör vara numeriskt kontrolleras först, därefter kontrolleras att vagnen existerar i ERP-systemets databas och information om vagnen hämtas in. Informationen om vagnen presenteras för granskaren som kontrollerar den och väljer att fortsätta eller återgå till startsidan. Visningen av informationen kan ses i figur 8.

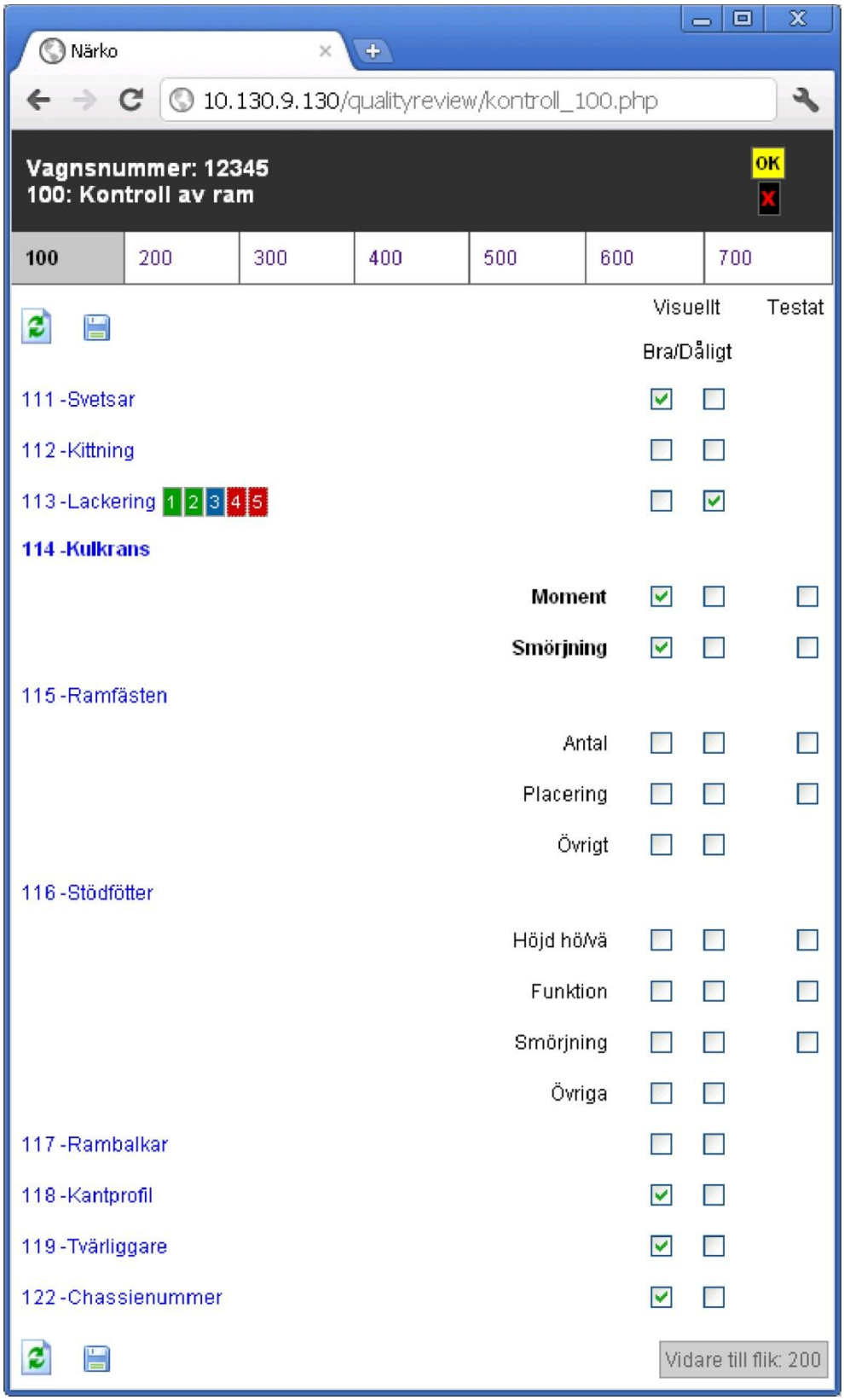


Figur 8. Visning av vagnsspecifika uppgifter för vagnsnummer 12345.

Om informationen stämmer överens med den vagn som ska granskas väljer granskaren att fortsätta till granskningsformuläret. På granskningsformuläret finns sju olika flikar, 100–700, som innehåller de kontrollpunkter som ska kontrolleras. Varje kontrollpunkt har en kod som börjar på kontrollflikens hundratal. Vissa punkter behöver inte granskas på alla vagnstyper, men de punkter som är obligatoriska att granska har angetts med fet stil.

Granskaren börjar med en överblick av den punkt som granskas. Beroende på om brister hittas, markeras först en kryssruta att kontrollpunkten antingen är visuellt dåligt eller bra. Beroende på vagnstyp, kräver vissa kontrollpunkter att testning ska utföras, då markeras även den kryssrutan. I figur 9 visas granskningsformuläret för en vagn där flera brister skapats på kontrollpunkten 113 – Lackering.

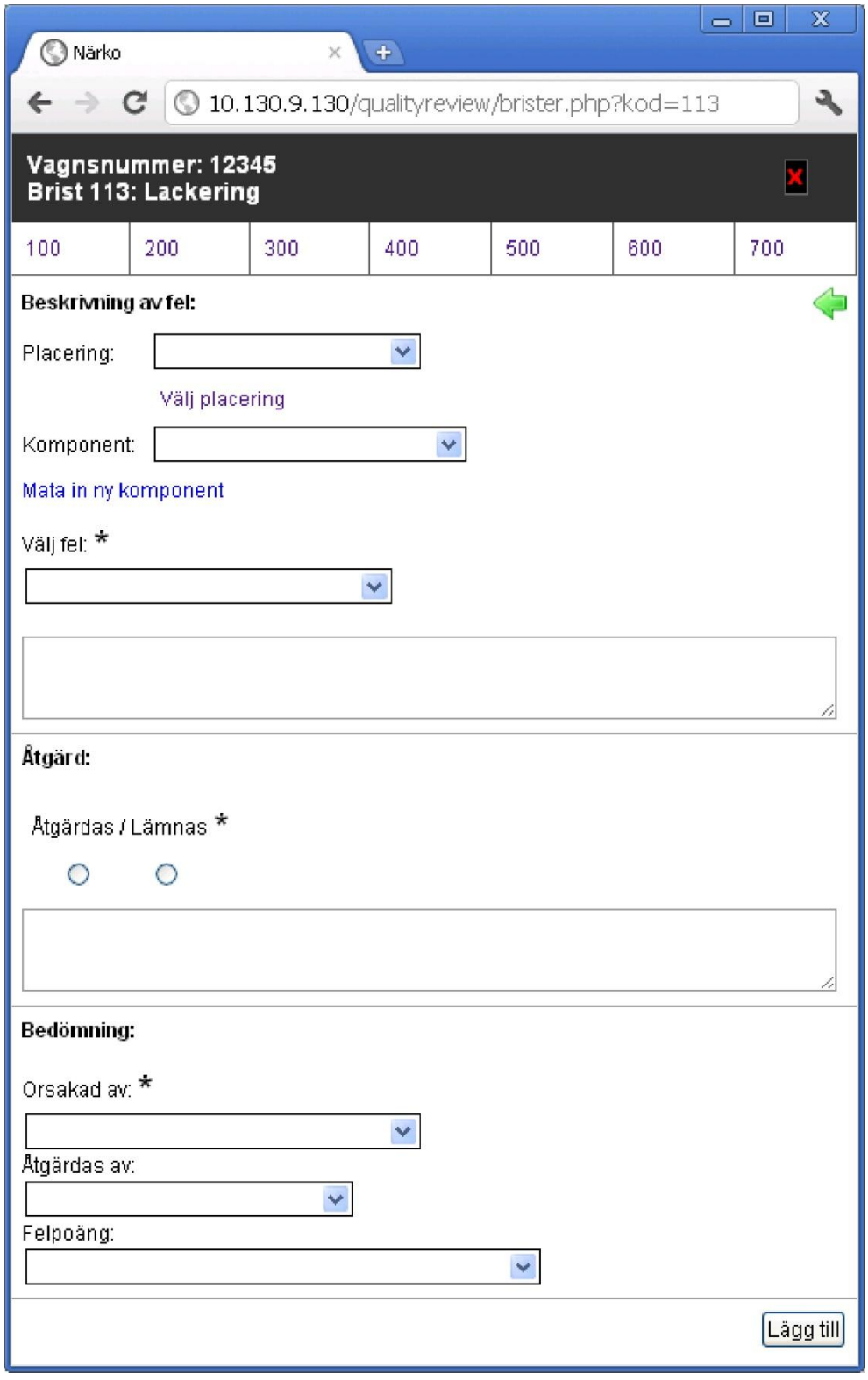
Bristerna har olika färgkodning för att de lättare ska kännas igen. Brister som ska åtgärdas har röd färg, de brister som åtgärdats är gröna. I de fall det är frågan om mindre allvarliga brister som lämnats har de en blå färg.



Figur 9. Granskningsformuläret för vagn 12345.

När nya brister ska läggas till så klickar granskaren på själva kontrollpunkten för att lägga till en ny för ifrågavarande punkt. Brister som infogas får ett nytt unikt löpande nummer för varje kontrollpunkt. Detta

nummer används sedan för att identifiera bristerna, uppdatera information eller ändra status. För att uppdatera information eller ändra status klickar man på rutan som innehåller bristens nummer. I figur 10 visas inmatnings- och redigeringsformuläret för en brist på kontrollpunkten 113 - Lackering.



Figur 10. Formulär för inmatning och redigering av brister.

Formuläret är indelat i tre sektioner, beskrivning av fel, åtgärd och bedömning. I sektionen för beskrivning av fel har man möjlighet att välja från en lista av positioner, men också sätta en visuell placering på en vagnstypsspecifik bild. Den visuella placeringen möjliggör en exaktare positionering av felet, utan att desto mera behöva beskriva detta i felbeskrivningsrutan. Vilken typ av fel måste också väljas från en fördefinierad lista.

Listan för val av komponent kan beskrivas som en dynamisk lista. Komponenterna i listan finns kvar om de används regelbundet, komponenter som inte använts på tre månader döljs automatiskt. Saknar granskaren någon komponent, kan han direkt från formuläret där brister matas in mata in en ny komponent. Ifall att samma komponent redan finns sparad, aktiveras den komponenten igen. Listan kan även redigeras av slutgranskarna på en skild administrationssida.

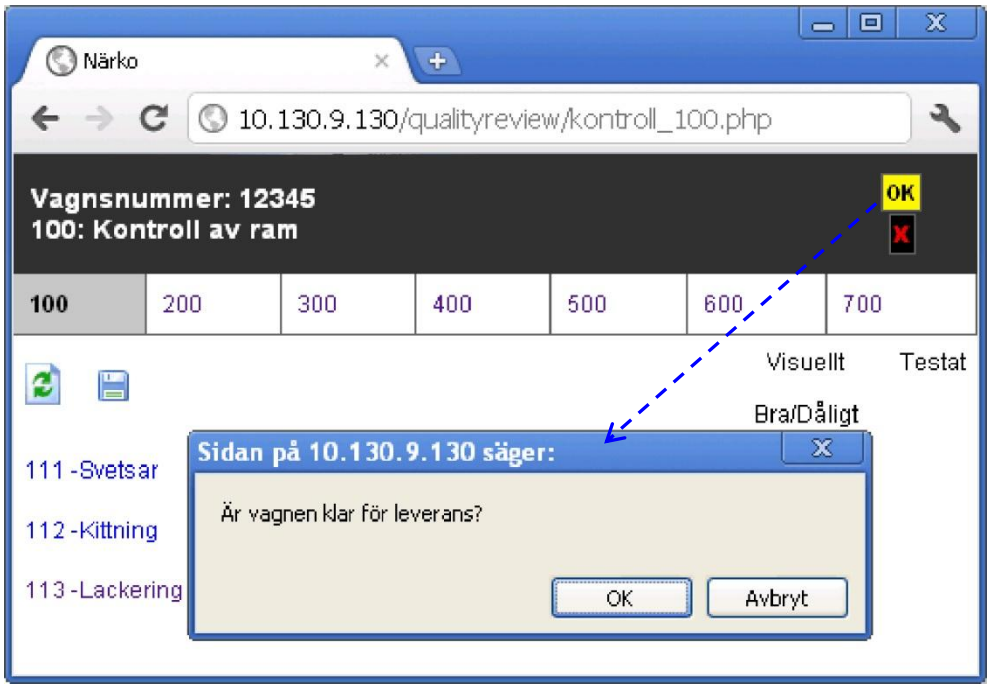
I sektionen åtgärd anger slutgranskaren om den brist han matar in ska åtgärdas eller inte. En beskrivning av förslag på åtgärd kan också anges. I den sista sektionen, bedömning, anges på vilken avdelning bristen uppstått, vilken avdelning som eventuellt ska åtgärda den, och en intern felpoäng för Närkos interna uppföljning. Närkos felpoäng är specificerade som:

- 1: Marginell betydelse, ingen inverkan på körduglighetен
- 2: Seriös reklamation, ingen inverkan på körduglighetен
- 3: Allvarlig reklamation, inverkar på körduglighetен
- 9: Säkerhetsrisk (trafik- eller personfara)

När alla vagnens kontrollpunkter granskats, skriver slutgranskaren ut en granskningsrapport som innehåller eventuella brister som hittats. Bristerna grupperas och ordnas i rapporten efter vilka avdelningar som ska åtgärda dem. När vagnens alla brister har blivit åtgärdade, granskats och godkänts, kan slutgranskaren godkänna vagnen för leverans.

Vagnen godkänns för leverans genom att klicka på den gula knappen med texten OK i programmet. Uppgifter om att slutgranskningen är klar överförs därefter också till ERP-systemet. Därför måste slutgranskaren bekräfta att granskningen verkligen är klar när han klickar på knappen OK. I figur 11

visas den bekräftelseruta som visas när slutgranskningen slutförs och vagnen anges leveransklar.



Figur 11. Bekräftelse att slutgranskningen har slutförts och vagnen är klar för leverans.

2.3 Uppgiftsbeskrivning

Uppgiften kan delas in i två större delar, där den ena delen omfattar utveckling av den befintliga webbapplikationen i PHP och den andra en nyutvecklad Android-applikation för att kunna använda en Android-enhets kamerafunktioner. Dessutom behöver funktionalitet utvecklas för att kommunikationen och utbyte av information mellan dessa två delar ska fungera. Utbytet av information kommer att genomföras med hjälp av teknikerna PHP, JSON och JavaScript.

När arbetet slutförts dokumenteras de viktigaste delarna av applikationen. Detta gäller både webbapplikationen och applikationen för Android.

2.3.1 Vidareutveckling av webbapplikation

Den befintliga webbapplikationen i PHP utökas med funktionalitet som kontrollerar ifall axeljusteringen är gjord för den vagnsnummer som

granskas. Om justeringen gjorts ska slutgranskaren ha möjlighet att visa värdena före och efter gjord justering. Axeljusteringsprotokoll behöver alltså kunna läsas och datat behandlas för att visa ett protokoll anpassat för webbläsare. Om justeringsprotokollet inte hittas så ska programmet även visa att justeringen är ogjord och att den bör kontrolleras med ansvariga på slutmonteringen.

Webbapplikationen anpassas även för moderna webbläsare som finns i dagens datorer och mobila enheter. Detta omfattar layoutförbättringar, modernare design och förbättrad funktionalitet. Skärmupplösningen har blivit högre på moderna datorer och smarta telefoner, varför storleken på knappar och text behöver ses över.

Rapporterna som skickas ut per e-post till projektchefen, kvalitetschefen och berörda avdelningschefer ska utvecklas. För att göra sökning enklare i e-postprogrammen ska hela granskningsrapporten eller nyckelord skrivas ut i meddelandetexten. På så sätt kan man lättare hitta t.ex. brister som matats in på kontrollpunkten 113 - Lackering.

2.3.2 Nyutvecklad Android-applikation

En Android-applikation byggs upp från grunden för en enkel åtkomst till funktionalitet som slutgranskningen har behov av. Webbapplikationen integreras i Android-applikationen och integration mellan applikationerna görs för att underlätta granskningen. Funktionalitet för att använda kameran i Android-enheten byggs in i applikationen. Kameran ska kunna användas för att i första skedet kunna fotografera granskade vagnar och ha bilderna arkiverade på en filserver inom det interna nätverket. Bilderna ska kunna användas av personalen för olika ändamål och bör vara av god bildkvalitet.

För att kunna använda den färdiga applikationen behöver en Android-enhet köpas in till Närko. Eftersom det är en miljö med ganska mycket smuts, metallspånor o.s.v. behöver enheten tåla en del slitage. Enheten måste även ha smidigt fungerande anslutbarhet med WiFi-nätverk, samt en bra batteritid när WiFi är aktiverat.

2.3.3 Kommunikation mellan applikationerna

För utbyte av information mellan Android-applikationen och webbapplikationen behöver interface utvecklas. Interfacen exponerar det som det båda applikationer behöver komma åt sinsemellan. Dessa förverkligas i webbapplikationen med teknikerna JavaScript, PHP och JSON. En basklass utvecklas för webbapplikationen som innehåller grundläggande funktionalitet som behövs i hela webbapplikationen. Detta innebär bl.a. identifiering, start av sessioner, aktuell granskad vagnsnummer och vilken slutgranskare som är inloggad i programmet.

För Android skapas ett JavaScript-interface för tvåvägskommunikation mellan de båda applikationerna. Större mängder data hämtas av Android-applikationen med HTTP request. Från Android kan data också postas med HTTP POST. Dessa två metoder körs mot nyutvecklade API:n i webbapplikationen.

3 Språk och tekniker

I detta kapitel presenteras de språk och tekniker som huvudsakligen använts vid genomförande av examensarbetet. Allmän information och historik presenteras först, därefter presenteras andra eventuella möjliga alternativ. I de fall där flera alternativ kunde ha valts, motiveras valet av just detta språk eller teknik. Till sist ges ett exempel på hur varje språk och teknik kan tillämpas för att ge ett liknande resultat när resultatet presenteras med respektive språk eller teknik. I vissa av exemplen blandas flera tekniker, i dessa fall är den huvudsakliga tekniken som visas i exemplet representerad med röd text. Kodkommentarer i respektive språk och teknik har skrivits med grön färg.

3.1 Webb- och skriptspråk

Under denna rubrik behandlas de viktigaste språk och standarder som används för att bygga webbsidor och webbapplikationer där resultatet kan visas i en webbläsare. De webb- och skriptspråk som presenteras är HTML, CSS, JavaScript på klientsidan och på serversidan PHP för generering av dynamiskt innehåll.

3.1.1 HTML

HTML står för HyperText Markup Language. Precis som namnet antyder är HTML inte ett programmeringsspråk utan tillhör familjen märkspråk. Exempel på andra märkspråk är XML och textfilsformatet RTF. Ett märkspråk utmärker sig med att det innehåller så kallade taggar där innehållet infogas. (Refsnes, Refsnes, & Refsnes, 2010, s. 1)

HTML används för att skapa HTML-dokument som med flera dokument tillsammans bildar en hel webbsida. Som med alla märkspråk är det inte meningen att taggarna ska visas ut vid presentation av informationen. När det gäller HTML så är det en webbläsare som tolkar informationen i taggarna och visar informationen inuti taggarna. Vanliga webbläsare idag är Microsoft Internet Explorer, Mozilla Firefox och Google Chrome. (Refsnes, Refsnes, & Refsnes, 2010, s. 2)

HTML-dokument byggs upp av taggar som består av både start- och sluttaggar. Dessa delar bygger upp så kallade HTML-element. De båda delarna kan också kallas öppningstagg och sluttagg. Varje elements innehåll ska placeras emellan dess start- och sluttagg. Vissa HTML-element kan också sakna innehåll, dessa stängs då i samband med starttaggen. Ett exempel på element som saknar innehåll är elementet för radbyte
, som avslutas direkt efter elementets namn. HTML-element kan också nästlas, d.v.s. att man placerar ett annat element före ett elements sluttagg. (Refsnes, Refsnes, & Refsnes, 2010, s. 13−14)

De flesta HTML-element kan också innehålla det som kallas HTML-attribut. Attributen ger möjlighet att specificera mer specifik information om ett element. Attribut specificeras alltid på ett elements starttagg inom citationstecken. Olika element kan ha olika attribut, det finns dock ett antal standardattribut som är tillgängliga på de flesta element. Dessa är:

- Class, används för att definiera en klass för ett element, användbart t.ex. vid användning av CSS.
- Id, används för att definiera ett unikt id för element, användbart t.ex. vid användning av JavaScript och CSS.
- Style, används för definiering av CSS direkt på ett element.
- Title, används för att sätta en titel på ett element, visas när man för muspekaren ovanför elementet.

(Refsnes, Refsnes, & Refsnes, 2010, s. 17−18)

Kommentarer kan infogas i HTML-koden för att förklara vad element eller sektioner av koden innehåller och har för betydelse. För att specificera att det gäller en kommentar inleds kommentarer med ”<!—” och avslutas med ”-->”. Kommentarerna visas inte ut när webbsidan visas i en webbläsare. (Refsnes, Refsnes, & Refsnes, 2010, s. 21)

Den vanligaste versionen av HTML som används idag kallas XHTML. XHTML är mera strikt definierad än tidigare HTML-versioner och är baserat på definitionen av XML. Till definitionen hör t.ex. att alla element bör avslutas och bildelement ska alltid innehålla ett beskrivande alt-attribut. En nyare HTML5-standard är under utveckling och kan användas i de

senaste webbläsarna till vissa delar. Standarden är inte fastställd ännu vid skrivtillfället och saknar en hel del planerade delar.

I kodexempel 1 skapas ett HTML-dokument som innehåller de grundläggande elementen som bygger upp HTML-kod. Dessa element är namngivna html, head och body. Först i koden har också specificerats vilken HTML-version som används i dokumentet. Elementet head innehåller ett meta-element som specificerar sidans teckenkodning och ett title-element där sidans titel i webbläsarens titelrad bestäms. Själva sidans innehåll består av en kommentar och ett rubrikelement och finns i body-elementet. Resultatet visas i figur 12.

Kodexempel 1. En HTML-sida innehållande de grundläggande elementen, ett rubrikelement och en kommentar.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type"
         content="text/html; charset=utf-8" />
   <title>Min XHTML-sida</title>
</head>
<body>
   <!-- Kommentar i HTML-kod -->
   <h1>Hello World!</h1>
</body>
</html>
```

Figur 12. HTML-sidan visad i en webbläsare för koden i kodexempel 1.

3.1.2 CSS

CSS är stilmallar och är förkortningen för Cascading Style Sheets. CSS används för att ställa utseende på HTML-dokument, d.v.s. oftast webbsidor, men kan även vara t.ex. e-postmeddelanden skrivna i HTML. CSS kan

användas för att bestämma det mesta som gäller webbsidors layout, t.ex. färger, teckensnitt, avstånd och positioner. Man kan göra inställningar som gäller element, specificerade klasser, id:n eller så kallade pseudo-element. (Powell, 2010, s. 429)

CSS-inställningarna kan göras på olika ställen. Det som brukar rekommenderas är en enskild fil med filändelsen .css, eftersom det ger cachningsfördelar. Det är också möjligt att definiera direkt i HTML-dokument inom taggen style eller direkt på enskilda HTML-element i attributet style. Definiering direkt på element bör endast användas om samma stil inte förekommer på några andra element.

För att ange stilen på enskilda element kan man använda HTML:s id-attribut som bara får förekomma en gång på en sida enligt HTML-standarden. I CSS definerar man stilmallen för ett id genom att inleda med nummertecken och hänvisa till ett id i HTML-dokumentet, t.ex. ”#uniktelement”.

Klasser i CSS används för att bestämma hur flera element ska presenteras. För att definiera en klass inleder man med en punkt och sätter dess namn direkt efter, t.ex. ”.cssklass”. Klasser kan också kedjas med andra klasser, specifika element eller id:n, t.ex. paragrafelementet p ”p.cssklass”. De klasser som man definierat behöver därefter sättas på de element i HTML-dokumenten där man vill använda dem. (Powell, 2010, s. 433–437)

Pseudo-element kan användas för vissa HTML-element, det vanligaste är länkar. Med pseudo-element i CSS kan man bl.a. bestämma hur en länk ser ut när man för muspekaren över den. Detta pseudo-element definieras med ”:hover”. (Powell, 2010, s. 454)

I kodexempel 2 används CSS för att ställa utseendet på ett HTML-dokument. I exemplet visas hur utseende ställs på direkt för element, med definierade klasser och id. För att illustrera pseudo-element har fet stil angetts för länkar, a-element, när muspekaren förs över. Resultatet ses i figur 13.

Kodexempel 2. HTML-dokument där innehållets presentation bestäms med CSS.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Min XHTML-sida med CSS</title>
   <style type="text/css">
   h1
   {
      color: red;
   }
   a:hover
   {
      font-weight: bold;
   }
   .subheader
   {
      color: blue;
   }
   #top
   {
      background-color: lightgrey;
   }
      #top .subheader
      {
            color: green;
      }
</style>
</head>
<body>
   <div id="top">
       <h1>Hello World!</h1>
      <span class="subheader">Underrubrik 1</span><br />
      <a href="http://www.novia.fi">Novia</a>
   </div>
   <div>
      <span class="subheader">Underrubrik 2</span>
   </div>
</body>
</html>
```

Figur 13. HTML-dokument innehållande CSS-stilmallar presenterat i en webbläsare för koden i kodexempel 2.

3.1.3 JavaScript

JavaScript är ett skriptspråk som används för dynamik på klientsidan i webbsidor. De flesta moderna webbsidor använder sig av JavaScript i någon form. JavaScript är så vanligt tack vare att alla moderna webbläsare, i vanliga datorer, mobiltelefoner och spelkonsoler innehåller en tolk för JavaScript. (Flanagan, 2011, s. 1)

Namnet JavaScript associeras ibland felaktigt med Java. Bortsett från liknande syntax så skiljer sig JavaScript helt från programmeringsspråket Java. JavaScript skapades av Netscape i ett tidigt skede när Internet och webben började nå framgångar. Netscape lämnade in språket för standardisering till ECMA (European Computer Manufacturer's Association). Eftersom JavaScript var ett varumärke registrerat av Sun Microsystems fick det standardiserade språket det besvärliga namnet ECMAScript. I praktiken talar dock alla om JavaScript, och alla moderna webbläsare implementerar version 5.0 av ECMAScript standarden. (Flanagan, 2011, s. 1–2)

JavaScript kan implementeras på olika sätt, om samma JavaScript-funktion ska köras på flera sidor implementeras den vanligen i en skild fil med filändelsen .js. Det går också bra att ha JavaScript-kod inom taggen script i ett HTML-dokument. I HTML-standarden finns olika sätt att anropa JavaScript-funktioner på, t.ex. när ett dokument laddat klart, när man klickar på ett visst element eller för muspekaren över det.

I kodexempel 3 visas hur JavaScript kan användas för att ta reda på dagens datum och visa detta i en webbläsares titelrad. Datumet visas direkt i titelraden när HTML-sidan tolkas av webbläsaren. En funktion som sätter en rubrik på sidan kräver dock anrop, antingen från användaren, eller som i exemplet när sidan laddats klart i body-elementets onload-attribut. Resultatet visas i figur 14.

Kodexempel 3. Skriver ut texten Hello World med hjälp av JavaScript i en HTML-sida. Dagens datum skrivs ut med JavaScript i webbläsarens titelrad.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type"
         content="text/html; charset=utf-8" />
   <title>Min XHTML-sida med JavaScript</title>
   <script type="text/javascript">
      // Skapar ett datumobjekt
      var date = new Date();

      /* Plockar ut dag, månad och år som skilda
      variabler för formatering vid utskift */
      var day = date.getDate();
      var month = date.getMonth() + 1;
      var year = date.getFullYear();

      // Sätter sidans titel till dagens datum
      document.title = year + "-" + month + "-" + day;

      function Greet()
      {
         // Referera till rubrikelementet
         var h1 = document.getElementById('greeting');
         // Sätter rubriken på sidan till "Hello World!"
         h1.innerHTML = "Hello World!";
      }
   </script>
</head>
<body onload="Greet();">
   <h1 id="greeting"></h1>
</body>
</html>
```

Figur 14. HTML-sidan med JavaScript visad i en webbläsare för koden i kodexempel 3.

3.1.4 PHP

PHP står för PHP: Hypertext Preprocessor, som är en rekursiv akronym. PHP är ett skriptspråk som används främst inom webbutveckling men kan också exekveras i en kommandotolk. I webbsidor tolkas PHP-koden av en programtolk kallad Zend Engine på webbservern. Programtolken returnerar ett resultat som vanligen presenteras för besökaren med hjälp av HTML och CSS. PHP kan också användas för att leverera en rad andra format som t.ex. XML och JSON.

Detta skriptspråk började ursprungligen utvecklas av Rasmus Lerdorf under namnet PHP/FI år 1994. I detta skede var det dock inget annat än en uppsättning binärer till protokollet CGI, som denna tid i huvudsak användes tillsammans med skriptspråket PERL. Efter hand, när behovet av funktionalitet ökade skrev Rasmus Lerdorf om hela uppsättningen av binärer i programmeringsspråket C och bytte namnet till PHP Tools. I den nya uppsättningen binärer infördes bland annat stöd för kommunikation med databaser. (History of PHP, u.d.)

I juni, år 1995 släpptes källkoden för PHP Tools publikt. Detta bidrog till att programmerare över hela världen kunde bidra med att fixa buggar och förbättra källkoden. Detta var den första release som betraktades som ett avancerat skriptinterface. Fortfarande var man dock tvungen att infoga det man ville tolka som PHP inom HTML-kommentarer i HTML-dokumenten, och PHP kunde endast användas i UNIX-miljö. År 1997 beslöt sig Rasmus Lerdorf tillsammans med två studerande, Andi Gutmans och Zeev Suraski att samarbeta och ännu en gång skriva om källkoden för PHP. Version PHP 3.0 lanserades i juni 1998 av PHP Development Team som vuxit fram efter att många frivilliga från hela världen anslutit sig. De största fördelarna med den nya versionen var att den var lätt att bygga ut med nya moduler samt att den hade stöd för objektorienterad programmering. Nu kunde PHP även användas på Windows och Macintoshservrar. (History of PHP, u.d.)

Den aktuella versionen är PHP version 5 som släpptes år 2004. Versionen har kraftigt förbättrad prestanda och utvecklat stöd för objektorienterad programmering. Det finns inga exakta siffror på hur populärt PHP är men

det sägs vara det mest använda skriptspråket på webbservrar över hela världen. (History of PHP, u.d.)

Alternativ till PHP kunde vara t.ex. ASP.NET eller Ruby On Rails. ASP.NET är utvecklat av Microsoft och är efterföljare till Active Server Pages (ASP). Den största fördelen med ASP.NET är att det är baserat på .NET-framework och att kodning kan göras med flera olika programmeringsspråk. Kodning i något av programmeringsspråken C# eller Visual Basic är dock vanligast. Felsökning av koden är också lätt att göra i programmeringsverktyget Visual Studio. ASP.NET kompileras också innan det sätts på den webbserver som ska visa webbsidorna. Ruby on Rails är ett ramverk med öppen källkod som är optimerat för programmerarens produktivitet. Ramverket använder skriptspråket Ruby som utvecklats av Yukihiro Matsumoto under nittiotalet. Ruby är objektorienterat och saknar primitiva datatyper. Ruby on Rails följer designmönstret Model-View-Controller (MVC) och tillhandahåller grundläggande funktioner för databashantering, d.v.s. för läsning, redigering och radering. (Getting started with Rails, u.d.)

Att valet blev PHP beror till stor del på att den nuvarande webbapplikationen är uppbyggd med PHP. Webbapplikationen innehåller över 200 stycken PHP-filer och ett stort klassbibliotek för att generera PDF-dokument direkt från genererad HTML-kod. I samråd med uppdragsgivaren konstaterades att det skulle ha tagit för lång tid att skriva om den nuvarande källkoden till något annat språk. Detta skulle inte heller ha gett något större mervärde, eftersom funktionaliteten skulle ha behållits i sin nuvarande form då applikationen fungerat bra. I övrigt skulle det inte ha funnits några större hinder för att ha använt några av alternativen som berördes ovan.

I kodexempel 4 visas hur PHP kan användas för att ta reda på dagens datum, formatera datumet och skriva ut en textsträng på skärmen. Datumet och textsträngen sparas undan i en variabel, som i PHP deklareras inledandes med ett dollartecken. Datumet visas i webbläsarens titelrad med hjälp av HTML-elementet title och texten skrivs ut i rubrikelementet h1. Resultatet visas i figur 15.

Kodexempel 4. Skriver ut texten Hello World med hjälp av PHP som en HTML-sida. Dagens datum visas i webbläsarens titelrad.

```
<?php
// Deklarerar två varibler inledandes med $-tecken
// Dagens datum i formatet år-månad-dag
$todaysDate = date("Y-m-d");
// Hälsningstext
$greeting = "Hello World!";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type"
         content="text/html; charset=utf-8" />
   <title><?php echo $todaysDate ?></title>
</head>
<body>
   <h1><?php echo $greeting ?></h1>
</body>
</html>
```

Figur 15. HTML-sidan med PHP visad i en webbläsare från koden i kodexempel 4.

3.2 Programmeringsspråk

I detta kapitel behandlas det renodlade programmeringsspråket Java. Javas historik och vad som skiljer Java från de flesta andra programmeringsspråk presenteras. Kapitlet avslutas med två lätta exempel på hur man kan använda Java för att bygga ett program med grafiskt gränssnitt eller ett konsolprogram.

3.2.1 Java

Det objektorienterade programmeringsspråket Java började utvecklas av Sun Microsystems år 1990 under ledning av James Gosling och Bill Joy. Java är designat för att kunna användas oberoende av arkitektur och miljö, erbjuda hög säkerhet och kunna köra program med höga krav på prestanda. (Niemeyer & Knudsen, 2005, s. 1–2)

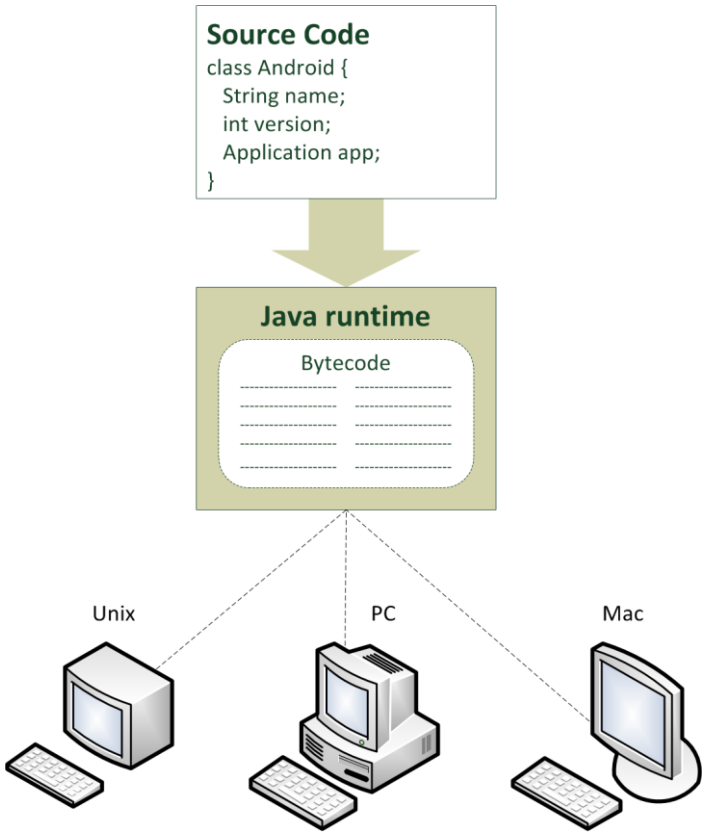
När utvecklingen påbörjades var Sun Microsystems ett företag som konkurrerade på en liten marknad för arbetsstationer. Bill Joy var fast besluten vid idén att det skulle vara möjligt att åstadkomma invecklade uppgifter med enkel programvara. Bill Joy började forska i detta och samlade ihop ett team programmerare som bland annat bestod av James Gosling, som kan sägas vara Javas skapare. Gosling gjorde sig först känd som skaparen av Gosling Emacs, en version av textredigeraren Emacs. Hans nästa projekt blev att designa Sun’s grafiska gränssnitt NeWs för Unix-miljö. NeWs förlorade dock till förmån för X Window System eftersom Sun inte valda att publicera källkoden för NeWs publikt. (Niemeyer & Knudsen, 2005, s. 1–2)

Goslings nästa projekt var att ta fram programmeringsspråket Oak som var tänkt att användas i inbyggda kretsar och hemelektronik. Den stora massan var dock inte ännu redo för handdatorer och smart TV-apparatur. Intresset för internet vid det här laget började däremot öka. Gosling och Bill Joy påbörjade arbetet med att anpassa Oak för webbruk. Oak var passande för internet eftersom det var litet, säkert, objektorienterat och inte var beroende av en viss arkitektur. År 1995 bytte Oak namn och blev Java. (Niemeyer & Knudsen, 2005, s. 3)

Till en början användes Java främst för webbapplikationer, så kallade Java-applets. Webbapplikationerna var dock begränsade och kunde inte göras speciellt avancerade, vare sig funktionellt eller utseendemässigt. Under årens lopp har Java utvecklats och har idag ett av de mest mångsidiga verktygen kallat Swing. Swing används för att bygga grafiska gränssnitt i traditionella klient-server applikationer. På senare år har Java också blivit en av de vanligaste plattformarna för webbaserade applikationer och webbtjänster. Javas framgångar har berott mycket på den fristående virtuella maskin, JVM (Java Virtual Machine) där Java-program körs. Den virtuella maskinen kan köras på olika typer av plattformar, allt från persondatorer till klockor och ringar. (Niemeyer & Knudsen, 2005, s. 2–4)

Java är både ett tolkat och kompilerat programmeringsspråk. Vid kompileringen av Java ändras koden till binära instruktioner. Till skillnad från programmeringsspråk som t.ex. C där kod kompileras läsbar för en typ av processor så kompileras Java-kod till ett universellt format som kan läsas av Javas virtuella maskin. Formatet kallas Java bytecode och de filer som skapas har filändelsen .jar, en såkallad Java runtime. Den virtuella maskinen tolkar och kör Java bytecode och hanterar programmets minnesanvändning i en säker virtuell miljö.

I figur 16 visas förfarandet med hur källkod blir Java bytecode och kan tolkas av en Java Virtual Machine oavsett plattform. (Niemeyer & Knudsen, 2005, s. 4–5)



Figur 16. Källkod i Java kompileras till ett universellt format, Java bytecode som kan tolkas av en Java Virtual Machine, oavsett plattform.

Tolkade språk har ansetts vara långsamma men Java är designat så att en (just-in-time) JIT-kompilerare kan kompilera kod läsbar av processorn under körning. Nackdelen med JIT är att uppstarten av program kan ta lång tid när kompileringen körs. Utvecklarna av Java har löst detta med en typ av anpassningsbar kompilering kallad HotSpot. HotSpot analyserar de koddelar som oftast exekveras och kompilerar dem på förhand till läsbar kod för processorn. Det sägs därför att Java kan vara snabbare än kompilerad C-kod i vissa fall. (Niemeyer & Knudsen, 2005, s. 5–6)

I kodexempel 5 visas hur ett konsolprogram skrivet i Java skriver ut texten ”Hello World” och dagens datum till skärmen. Resultatet kan ses i figur 17 där programmet har körts i Windows-miljö.

Kodexempel 5. Skriver ut texten Hello World och dagens datum med Java i ett konsolfönster.

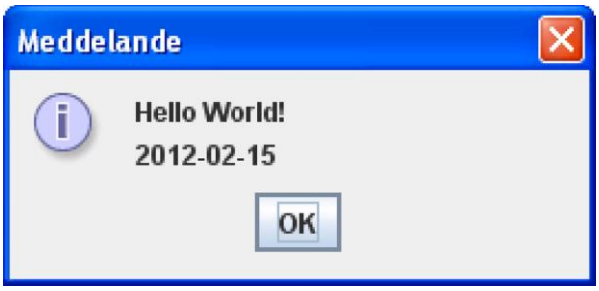
```
import java.util.Date;
import java.text.SimpleDateFormat;

public class HelloConsole {
   public static void main(String[] args) {
      Date date = new Date();
      SimpleDateFormat dateformat;
      dateformat = new SimpleDateFormat("yyyy-MM-dd");

      System.out.println("Hello World");
      System.out.println(dateformat.format(date));
   }
}
```

Figur 17. Resultatet av Java-koden i kodexempel 5 visat i ett konsolfönster.

Motsvarande program där Javas grafiska verktyg Swing används för att skriva ut dagens datum och texten Hello World i en dialogruta visas i kodexempel 6 och resultatet i Windows-miljö kan ses i figur 18.

Kodexempel 6. Skriver ut texten Hello World och dagens datum i en dialogruta i Java med Javas grafiska bibliotek, Swing.

```
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.swing.*;

public class HelloGUI {
   public static void main(String[] args) {
      Date date = new Date();
      SimpleDateFormat dateformat;
      dateformat = new SimpleDateFormat("yyyy-MM-dd");

      JOptionPane.showMessageDialog(null, "Hello World!" +
      "\n" + dateformat.format(date));
   }
}
```



Figur 18. Dialogruta med resultatet av Java-koden i kodexempel 6.

3.3 Övriga tekniker

3.3.1 SQL

SQL är en förkortning av Structured Query Language och är det språk som vanligen används för att kommunicera med databasmotorn. Alla de vanligaste databasmotorerna använder SQL, såsom Oracle, MySQL, Microsoft SQL-Server och Access.

SQL är speciellt ämnat för att användas med databaser som är byggda enligt relationsmodellen. Fördelar med att bygga en relationsdatabas är att flera användare lättare samtidigt ska kunna få fram information. I en väl planerad relationsdatabas är det dessutom lätt att utöka mängden information som ska sparas eller förändra den ursprungliga strukturen och datat. (Patrick, 2008, s. 3–7)

En relationsdatabas byggs upp av tabeller. En tabell innehåller namngivna kolumner som beskriver den data som sätts in i kolumnen. Kolumner kan ha en koppling, en s.k. relation till en kolumn i en annan tabell. När information sätts in i en tabell skapas rader i tabellen, där finns datat lagrat avskilt i kolumnerna som definierats. (Patrick, 2008, s. 10)

SQL-språket brukar delas in i tre grupper: DDL (Data Definition Language), DQL (Data Query Language) och DML (Data Manipulation Language). Med DDL så skapar, ändrar eller raderar man tabeller. I kodexempel 7 visas hur en databastabell namngiven ”cities” skapas i databasmotorn MySQL. Tabellen innehåller tre kolumner, en identifieringskolumn kallad id, en för namn och en för befolkningsmängden. Kolumnen id är definierad som primärnyckel, vilket

betyder att kolumnen används som unik identifierare för raderna som infogas. Datatyper och parametrar kan variera mellan olika databasmotorer.

Kodexempel 7. Skapar en databastabell cities, innehållande tre kolumner som representerar städer.

```
CREATE TABLE cities (
  id int(11) NOT NULL,
  name varchar(100) NOT NULL,
  population int(11) NOT NULL,
  PRIMARY KEY (id)
)
```

Med hjälp av DML-kommandon så läggs nya rader till eller så uppdaterar man eller raderar rader från tabeller. Kodexempel 8 beskriver hur fyra nya rader läggs till, hur en rad uppdateras och en tas bort.

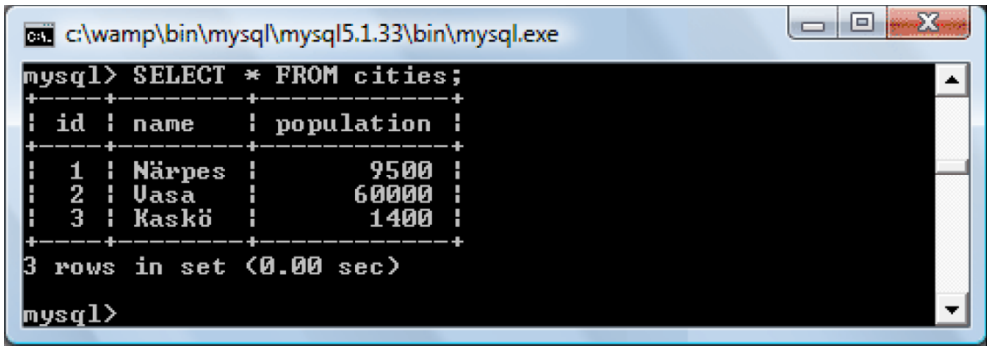
Kodexempel 8. Fyra nya rader läggs till i tabellen cities, befolkningen uppdateras för raden med id: 1 och raden med id: 4 tas bort.

```
INSERT INTO cities (id, name, population) VALUES
(1, 'Närpes', 10000),
(2, 'Vasa', 60000),
(3, 'Kaskö', 1400),
(4, 'Ankeborg', 316000);

UPDATE cities SET population = 9500 WHERE id = 1;

DELETE FROM cities WHERE id = 4;
```

DQL beskriver endast hur hämtning av information från tabeller skrivs. I kodexempel 9 hämtas alla rader ut från tabellen ”cities” och resultatet visas i figur 19.

Kodexempel 9. Alla rader hämtas ut från tabellen cities.

```
SELECT * FROM cities;
```



Figur 19. Resultatet från hämtningen av data från tabellen cities i en MySQL-terminal.

3.3.2 ODBC

ODBC (Open Database Connectivity) är ett API (Application Programming Interface) för åtkomst till databaser. ODBC utvecklades år 1992 av SQL Access group till följd av att företag började ha flera olika databashanterare i bruk. ODBC har utformats för maximal kompatibilitet att kommunicera med olika databashanteringssystem och möjliggör användning av samma källkod i databasapplikationer.

Varje databashanterare har sina egna drivrutiner som används vid anrop mot ODBC-interfacet. Drivrutinerna ansvarar för att översätta databasförfrågningar till något som varje databashanterare förstår. För databasförfrågningar används standard SQL. ODBC är språkoberoende och kan användas både i UNIX-, Macintosh- och Windows-miljö. (What is ODBC?, u.d.)

3.3.3 JSON

JSON står för JavaScript Object Notation och är ett kompakt textformat som används för serialisering av data, t.ex. objekt med flera egenskaper. Begreppet serialisering innebär att data översätts till ett annat format för att underlätta lagring. Det serialiserade datat ska gå att återskapa till den form det var i före serialiseringen. JSON är baserat på en delmängd av skriptspråket JavaScript. JSON finns specificerat i standarden för JavaScript version ECMA-262 3.0 och standarden RFC 4627. (Flanagan, 2011, s. 786)

JSON är plattformsoberoende och finns ofta färdigt tillgängligt i flera programmeringsspråk eller kan installeras med tilläggsbibliotek. JSON-data ska vara unicode kodat, som standardencoding används UTF-8. JSON består av objekt, vektorer och värden. Objekt och vektorer innehåller i sin tur också värden och kan representeras av textsträngar, tal, objekt, vektorer, sant och falskt eller null. Egenskaper i objekt och värdena i vektorer separeras med kommatecken. (Introducing JSON, u.d.)

Ett annat alternativ för utbyte av information hade varit XML. XML står för eXtensible Markup Language och är precis som HTML uppbyggt av taggar. Valet av JSON före XML berodde på att JSON oftast blir lättare till storleken vilket är en fördel för mobila enheter och överföring över trådlösa nätverk.

JSON är också lätt att komma igång med och använda både i PHP och Java. I kodexempel 10 och 11 så representeras samma information som JSON respektive XML för jämförelse.

Kodexempel 10. Representation av data med JSON.

```
{
  "Example": {
    "Greeting": "Hello World!",
    "Date": "2012-01-01",
    "ProgrammingLanguages": {
      "Language": [
        "C#",
        "C++",
        "PHP"
      ]
    }
  }
}
```

Kodexempel 11. Representation av data med XML.

```
<Example>
    <Greeting>Hello World!</Greeting>
    <Date>2012-01-01</Date>
    <ProgrammingLanguages>
       <Language>C#</Language>
       <Language>C++</Language>
       <Language>PHP</Language>
    </ProgrammingLanguages>
</Example>
```

4 Program och utvecklingsverktyg

4.1 Eclipse

Eclipse är ett utvecklingsverktyg för programvara, främst för programmeringsspråket Java. Eclipse kan även användas för ett flertal andra programmeringsspråk som t.ex. C++, Python och PHP. Eclipse är baserat på öppen källkod och utvecklas av den ideella organisationen The Eclipse Foundation. (Eclipse (programvara), 2012)

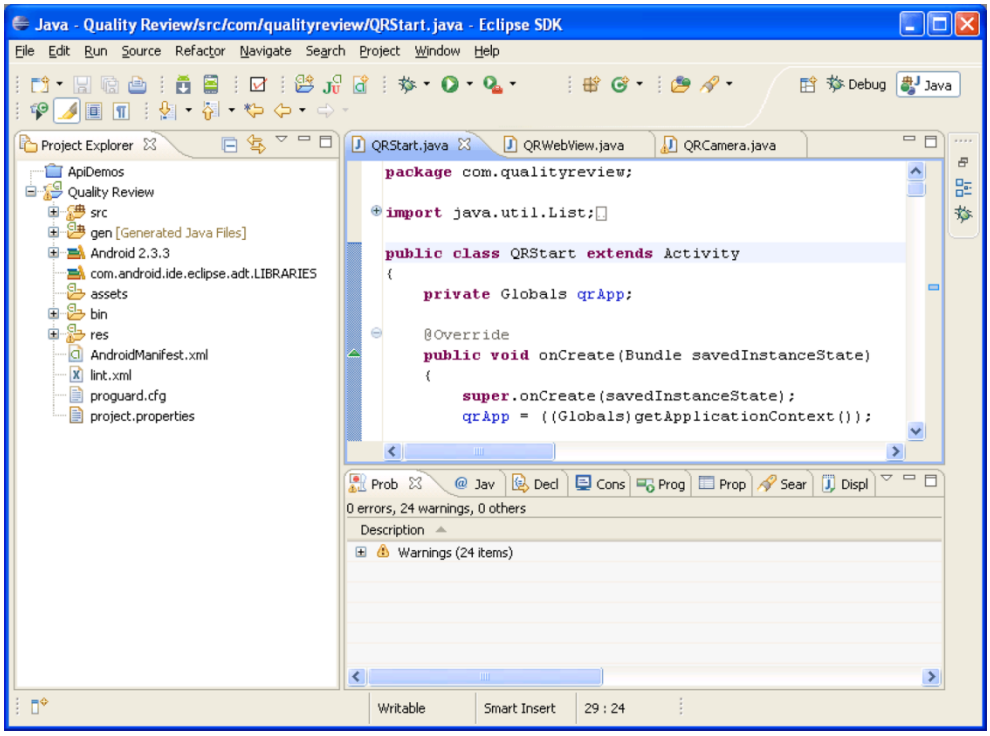
Grunden till Eclipse var The Eclipse Project som grundades av IBM i slutet av år 2001, med stöd från ett flertal utvecklare av programvara. Några av de andra grundarna var Borland, IBM, Rational Software, Red Hat och SuSE. I slutet av år 2003 hade The Eclipse Project vuxit till över 80 medlemmar. I februari år 2004 omformades projektet till den ideella organisationen The Eclipse Foundation som driver utvecklingen vidare. (About the Eclipse Foundation, u.d.)

Till Eclipse går det att få en hel del plug-ins och moduler som kompletterar verktyget. Standardinstallationen innehåller redan en hel del användbar funktionalitet, som refaktorering av kod för t.ex. byte av variabelnamn på alla ställen och automatisk genering av konstruktorer och såkallade get- och set-metoder.

Ett av alla plug-ins är Android Development Tools (ADT), som integrerar funktionalitet för att bygga Android applikationer i Eclipse. ADT gör att det går snabbare att skapa nya Android-projekt, skapa användargränssnitt, bygga nya komponenter byggda på Androids ramverk samt köra och debugga applikationerna tillsammans med Android SDK. I Android Development Tools ingår även export av Androids applikationsfiler .apk för att kunna distribuera applikationerna. (ADT Plugin for Eclipse, u.d.)

Användargränssnittet i Eclipse är uppbyggt av flera fönster. Fönstren kan anpassas och flyttas efter eget tycke och flera tilläggsfönster som är dolda som standard kan tas fram och visas. Som standard visas ett fönster där alla projekt listas och projektens innehålls visas som en trädstruktur. I mitten

finns kodfönstret där man kan ha flera kodfiler öppna samtidigt i flikar och byta mellan dem. Längst ner visas en lista på varningar eller problem som uppkommer i källkoden. En bild på hur gränssnittet ser ut visas i figur 20.



Figur 20. Standardanvändargränssnitt för Java-utveckling i Eclipse.

4.2 Android

Android är ett operativsystem för mobila enheter som är baserat på en modifierad Linux-kernel. Utvecklingen påbörjades år 2003 av företaget Android Inc. som grundades samma år. År 2005 köpte Google företaget Android Inc som en del av deras strategi att satsa på den mobila marknaden. Google övertog därmed utvecklingen och utvecklingsteamet som jobbat för Android Inc.

Google ville att Android skulle vara baserat på öppen källkod och därför lanserades också Android under den öppna källkodslicensen Apache License. Alla som vill använda Android kan därför ladda ner källkoden och bygga sina egna tillägg och anpassa Android efter sina behov. Detta passade perfekt för hårdvarutillverkare av mobiltelefoner och smarta telefoner som behövde något nytt efter att Apple’s iPhone lanserats. (Lee, 2011, s. 2)

Open Handset Alliance bildades av Google tillsammans med tillverkare som bl.a. Motorola, Samsung och Sony Ericsson. Open Handset Alliance består 2011 av 80 medlemmar vars huvudsakliga mål är att utveckla Android för smarta telefoner och surfplattor. (Komatineni, MacLean, & Hashimi, 2011, s. 4)

Den vanligaste versionen för smarta telefoner vid skrivtillfället är version 2.3 som lanserades i december år 2010. Versionen innehåller flera förbättringar mot tidigare versioner, bl.a. stöd för WiFi-hotspot, bättre bildkvalitet vid användning av kamera vid dåliga ljudförhållanden och väsentligt förbättrad prestanda. (Komatineni, MacLean, & Hashimi, 2011, s. 5)

Som alternativ till Android-plattformen finns bland annat Microsofts operativsystem Windows Phone och Apples iOS. Det finns också andra populära plattformar som Symbian. Symbian valdes främst bort p.g.a. att Nokia bestämt sig att avveckla operativsystemet i sina telefoner i framtiden. (Nokia Press Releases, 2011)

Den senaste lanserade versionen av Windows Phone är version 7. Microsoft har gjort en nysatsning på den mobila marknaden och drastiskt tänkt om gentemot deras tidigare mobila OS, Windows Mobile. Windows Phone 7 inför ett helt nytt simplifierat användargränssnitt och beter sig annorlunda mot de flesta andra liknande system. Windows Phone 7 använder sig av två populära och moderna ramverk, Silverlight och XNA för uppbyggnad av grafiska gränssnitt. Silverlight är en lättare version av Microsofts WPF för vanliga datorer, och lämpas därför bättre för mobila enheter. XNA är mera lämpat för spel och används bland annat i spelkonsolen XBOX 360. Applikationer för Windows Phone kan skrivas i programmeringsspråken C# eller Visual Basic för .NET-ramverket och som utvecklingsverktyg används Visual Studio. (Petzold, Programming Windows Phone 7, 2010, s. 2)

Apples operativsystem för mobila enheter iOS släpptes i version 5 år 2011. Operativsystemet används enbart i Apples produkter som iPhone och iPad. Utbudet av enheter är därför begränsat. Fördelen med att det inte finns flera tillverkare som använder sig av iOS är att Apple har större kontroll över

funktionalitet och kompabilitet. För att bygga och kompilera applikationer för iOS krävs en Apple-dator som kör Mac OS X. Applikationerna för iOS skrivs i programmeringsspråket Objective-C och vanligen i programmet XCode för MacOS.

Det finns också andra populära alternativ som t.ex. Titanium Appcelerator som möjliggör att applikationer kan byggas med hjälp av HTML, CSS och JavaScript. Applikationerna översätts sedan från respektive webbteknik till kod i Objective-C eller Java.

Valet av Android gentemot Windows Phone och iOS kan främst motiveras med det större utbudet av telefoner i olika storlek- och prisklasser. Ett stort hinder för utveckling av applikationen för iOS var att ingen Apple-dator fanns att tillgå. När lärdomsprovet påbörjades var urvalet av Windows Phone telefoner begränsat. En lämplig modell, med krav på stor skärm och god tålighet för damm och slitage hittades bland alla tillgängliga Android telefoner på marknaden.

4.3 Android SDK

Android SDK är ett hjälpmedel som hjälper utvecklare att bygga applikationer för Android. Förkortningen står för Software Development Kit och innehåller vanligen flera API, kompilator, emulator och dokumentation.

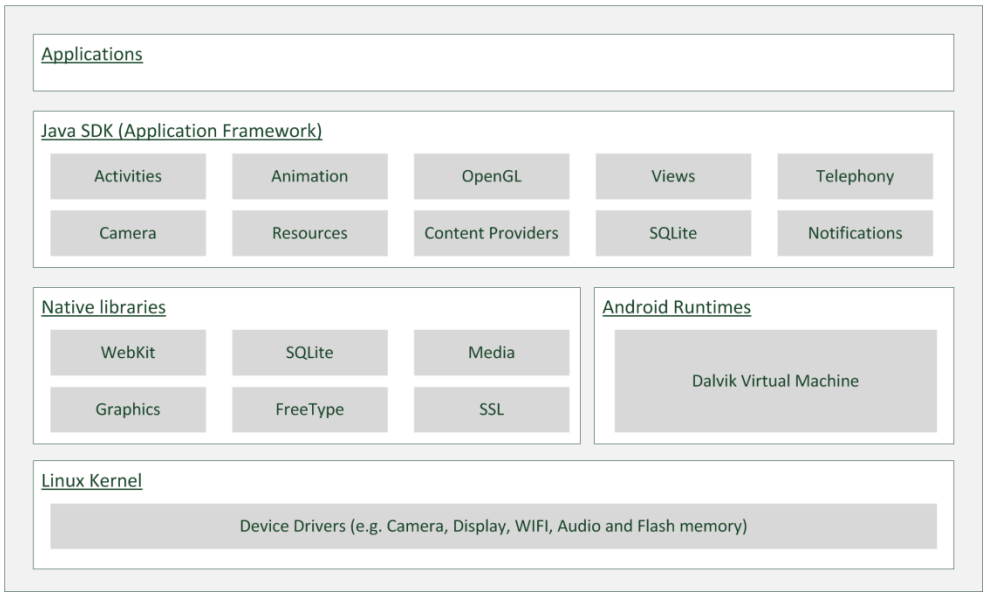
När man använder Android SDK för utveckling av applikationer så programmerar man i Java. SDK:n stöder det mesta av Javas standardplattform Java SE, undantaget verktygen för grafiska gränssnitt AWT och Swing. I stället för dessa har Android SDK ett eget modernt ramverk för grafiska gränssnitt. (Komatineni, MacLean, & Hashimi, 2011, s. 2)

Precis som andra Java-program behöver man en Java Virtual Machine för att köra program. Eftersom mobila enheter har mindre processorkapacitet och minne använder Android en optimerad variant kallad Dalvik Virtual Machine. För Dalvik VM kompileras källkoden till Dalvik Executable-filer (.dex-filer) där kod från flera klasser återanvänds, vilket minskar de

kompilerade filernas storlek till hälften jämfört med vanliga Java archive-filer. (Komatineni, MacLean, & Hashimi, 2011, s. 6)

Androids arkitektur delas upp i fem olika sektioner i fyra lager. Sektionerna är applications, Java SDK, native libraries, Android runtimes och Linux kernel. Det understa lagret, kärnan som Android baserar sig på är en Linux kernel. Kerneln innehåller drivrutiner för enheters hårdvarukomponenter, såsom skärm, ljud och nätverk. Nästa lager native libraries innehåller bibliotek för huvudfunktioner i Android, t.ex. WebKit för att visa webbsidor och det inbyggda biblioteket för relationsdatabaser SQLite.

På samma lager som native libraries återfinns den optimerade virtuella maskinen Dalvik VM som gör att varje Android applikation kan köras som en egen process, i en egen instans av Dalvik VM. På lagret Java SDK finns bibliotek för bl.a. aktiviteter, vyer och kamera, som man använder för att utveckla applikationer för det översta lagret. Android innehåller färdiga applikationer som distribueras med enheterna, bl.a. telefon-, webbläsar- och meddelandeapplikationer. Se figur 21 för en översikt av Androids arkitektur. (Lee, 2011, s. 3–4)



Figur 21. Androids arkitektur uppdelad i fyra lager och fem sektioner.

Activities och Views är av de viktigaste biblioteken när man bygger applikationer med Android SDK. Views specificerar de grafiska komponenter som visas i gränssnittet som t.ex. etiketter, textrutor och

knappar. Views kan även kallas widgets. Så kallade viewgroups används för att gruppera vyer på ett visst sätt. Layout i Android byggs upp med XML och kan jämföras med t.ex. Microsoft WPF och JavaFX. Activities kan beskrivas som en motsvarighet till fönster som har tillgång till de grafiska komponenterna. I en activity bestäms vad som ska hända när man utför något i programmet, t.ex. trycker på en knapp.

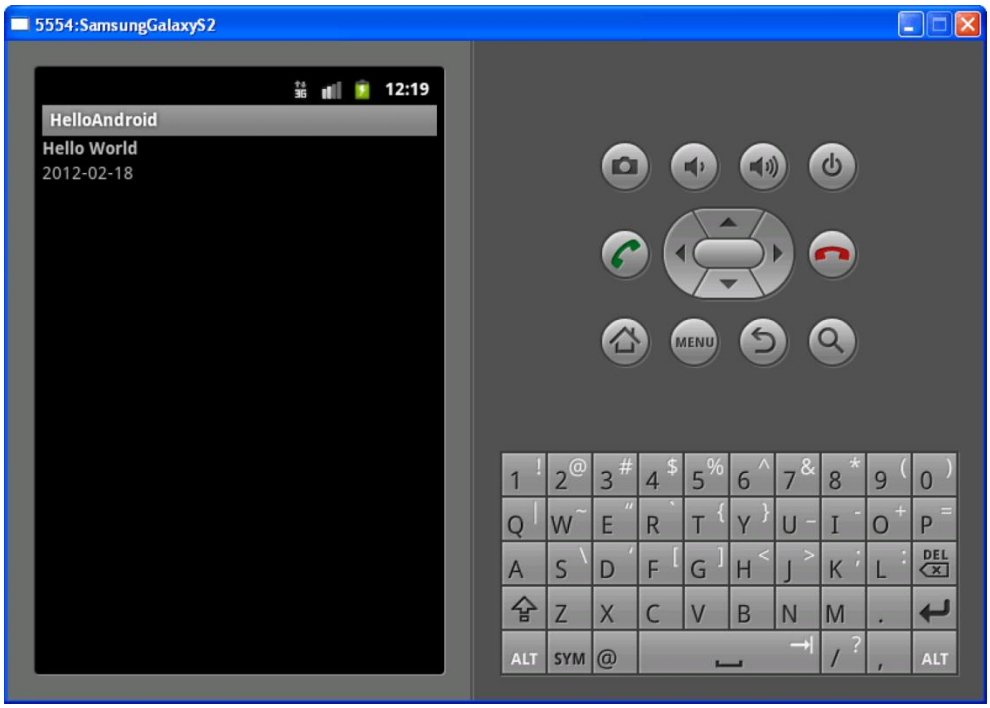
I kodexempel 12 och 13 skapas en Android Applikation som skriver ut ”Hello World” och dagens datum i varsin widget av typen TextView. Användargränssnittet defineras i en skild fil med XML enligt kodexempel 12. Vad som ska utföras när applikationen startar bestäms i activityn HelloWorld i kodexempel 13. Slutresultatet ses i figur 22 där applikationen startats i Android-emulatorn. Emulatorn kör Android version 2.3.

Kodexempel 12. Skapar ett användargränssnitt med XML som innehåller två widgets av typen TextView.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView
      android:id="@+id/tvHello"
      android:layout_width="fill_parent"
      android:layout height="wrap content"
      android:textStyle="bold" />
   <TextView
      android:id="@+id/tvDate"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" />
</LinearLayout>
```

Kodexempel 13. Android ”Hello World” activity som bestämmer vad som ska utföras när applikationen startas.

```
package my.hello.android;
import java.text.SimpleDateFormat;
import java.util.Date;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloWorld extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      TextView tvHello;
      tvHello = (TextView)findViewById(R.id.tvHello);
      tvHello.setText("Hello World");

      Date date = new Date();
      SimpleDateFormat dateformat;
      dateformat = new SimpleDateFormat("yyyy-MM-dd");

      TextView tvDate = (TextView)findViewById(R.id.tvDate);
      tvDate.setText(dateformat.format(date));
   }
}
```



Figur 22. Resultatet när programmet från koden i kodexempel 12 och 13 körs i Android-emulatorn.

5 Utförande

Detta kapitel behandlar det praktiska utförandet av examensarbetet. Utförandet delades in i två huvudsakliga delar, en Android- och en webbapplikationsdel.

5.1 Planering

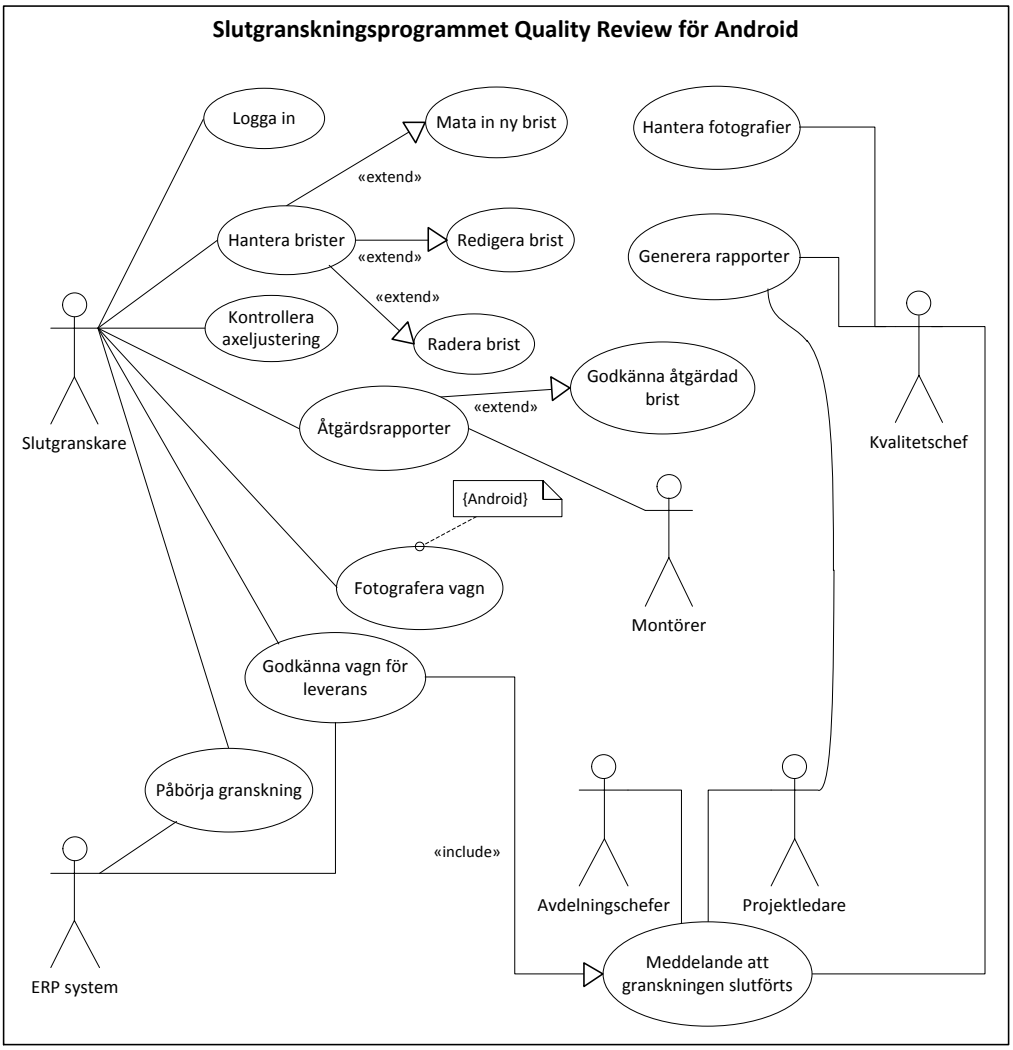
Utförandet av lärdomsprovet började med ett planeringsmöte. Tillsammans med uppdragsgivaren behandlades förslag till förbättringar i slutgranskningsprocessen. Efter planeringsmötet var utgångsläget att webbapplikationen Quality Review som fanns i företaget skulle byggas ut med hjälp av HTML5. Det visade sig dock efter lite efterforskningar att åtkomst till kameran ännu inte fanns att tillgå i HTML5. Andra alternativa lösningar fick av den orsaken sökas för implementering av förbättringarna.

För åtkomst till en mobil enhets kamera behövdes en applikation utvecklas för de mobila operativsystem som finns på marknaden. Applikationen medför också andra fördelar, bland annat finns fler möjligheter att göra funktionalitet tillgänglig i en och samma applikation för slutgranskarna. Utöver kamerafunktionaliteten är det också lätt att integrera den befintliga webbapplikationen i den mobila applikationen.

Ett val av utvecklingsplattform för den mobila applikationen, och därmed också en mobil enhet gjordes i samråd med Närko. Valet föll på Android. Motiveringarna till valet var många men de tyngsta skälen var att utvecklingsverktygen var gratis och att utbudet av Android-enheter var stort. Valet av Android-enhet blev av den senare orsaken svårt, eftersom det fanns så många modeller på marknaden vid anskaffningstillfället.

Kraven som var ställda var i alla fall att det skulle vara en smart telefon med bra kamera och en stor skärm, minst 4 tum. Valet stod till sist mellan Samsung Galaxy S2 och HTC Sensation. Valet av enhet blev enkelt när den lokala elektronikaffären i Närpes endast hade en modell i lager. En Samsung Galaxy S2 med en dubbelkärnig 1,4 GHz processor, 8 megapixel kamera och 4,2 tum stor skärm köptes.

Själva funktionaliteten planerades tillsammans med kvalitetschefen och Närkos anlitade IT-konsult. De olika stegen som skulle utföras dokumenterades på papper i punktform. I figur 23 visas ett UML Use Case diagram. Av diagrammet framgår den huvudsakliga funktionaliteten som planerats i den vidareutvecklade Quality Review applikationen, efter att examensarbetet slutförts.



Figur 23. Use Case-diagram som visar funktionaliteten i Quality Review efter slutfört examensarbete.

5.2 Android

5.2.1 Allmänt om grafiska gränssnittet

Det grafiska gränssnittet i Android-applikationen består av fyra separata vyer. Vyerna är uppbyggda i enlighet med Androids definition för XML-layouter och innehåller viewgroup- och widgetelement. Utformningen har gjorts så att den viktigaste funktionaliteten ska vara lättåtkomlig med synliga knappar eller via Android-enhetens menyknapp. Menyerna byggs också upp med XML och innehåller då de menyalternativ som ska visas.

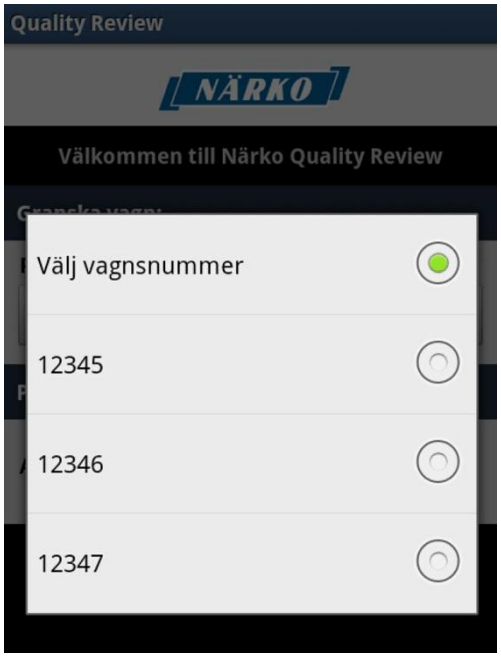
5.2.2 Startvyn

Den första vy som visas när applikationen startar är startvyn. Vyn innehåller valmöjligheter som gör det lätt att komma igång med granskningen. Som användare kan man välja att påbörja granskningen från Android-applikationen eller gå till webbapplikationen via menyknappen. I figur 24 ses användargränssnittet som visas när applikationen startas.

Figur 24. Startvyn i Android-applikationen.

Om det finns vagnar där granskningen redan påbörjats men inte färdigställts, så visas en widget med påbörjade vagnar. Widgeten kallas spinner och kan liknas vid en så kallad drop-down meny i flera andra

gränssnittsmiljöer. Trycker man på spinnern så visas en lista på vagnsnummer enligt figur 25.



Figur 25. Startvyns visning av vagnar där slutgranskning påbörjats.

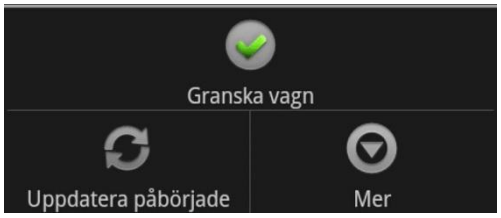
Väljer man något av vagnsnumren skickas man vidare till webbapplikationen som tar emot det valda numret för vidare processering. I de fall som det handlar om en ny vagn som ska granskas, kan numret matas in manuellt i textfältet under spinnern. Matar man in ett korrekt numeriskt värde och trycker på OK-knappen, skickas man vidare till webbapplikationen där granskningen påbörjas.

Sista valet på startvyn är Android-enhetens menyknapp. Menyknappen på startvyn visar en meny som innehåller tre huvudval:

- Granska vagn
- Uppdatera påbörjade
- Mer

Granska vagn öppnar webbapplikationsvyn. Uppdatera påbörjade uppdaterar listan med påbörjade vagnar på startvyn. Alternativet Mer visar en lista med underalternativ. Underalternativen är ”Kontrollera och överför bilder i kö” och ”Avsluta”. Det första alternativet är kopplat till funktionalitet som kontrollerar om det finns bilder som inte överförts till

filservern. Detta skulle kunna inträffa vid störningar i nätverksanslutningen. Kontrollen av bildöverföringen till filservern och dess kö, beskrivs ytterligare i kapitel 5.2.7. Alternativet avsluta stänger hela applikationen. Startvyns meny innehållande huvudvalen visas i figur 26.



Figur 26. Startvyns meny, visas när man trycker på Android-enhetens menyknapp.

5.2.3 Slutgranskningsvyn

Slutgranskningsvyn består endast av en widget, en s.k. webbvy. Webbvyn har i Androids XML namnet WebView. Webbvyn ramar in webbapplikationen som finns på den interna webbservern, d.v.s. den är länkad till webbapplikationens URL. Det som sker inom webbvyn exekveras på webbservern, resultatet visas sedan i Android-applikationen som HTML tillsammans med CSS och JavaScript.

Slutgranskningsmenyn innehåller likt startvyn en meny som visas genom att använda menyknappen på Android-enheten. Menyn i denna vy innehåller tre menyalternativ:

- Kamera
- Brister
- Till startskärm

För att de två första alternativen ska kunna väljas, krävs att man har en pågående granskning. Det aktuella vagnsnumret som granskas finns lagrat i en sessionsvariabel i webbapplikationen. Android-applikationen kontrollerar att sessions-variabeln har fått ett värde, d.v.s. att granskning pågår av en vagn. Om sessionsvariabeln saknar värde anses granskningen ej vara påbörjad och de två första menyalternativen inaktiveras.

I figur 27 har slutgranskning påbörjats för vagnsnummer 12345. När menyn expanderas som i figuren kan man därför välja alla tre menyalternativ.

Figur 27. Slutgranskningsvyn med tillhörande meny i nedre kanten. I figuren har man angett vagnsnummer 12345 för granskning.

Alternativet kamera visar kameravyn. För att få en bättre överblick av en vagns brister kan man använda menyvalet brister. Brister öppnar en separat vy, där alla brister på den granskade vagnen listas i kronologisk ordning, enligt kontrollpunkt. Listningen visar vilka brister som lämnats, åtgärdats och vilka brister som ännu ska godkännas. I figur 28 visas hur de lämnade bristerna är ljusgråa, brister som godkänts har en grön bock och bristerna som väntar på godkännande fet text och en vit bock. Användaren kan trycka på en brist i listan för att gå till bristens redigeringsformulär.

Figur 28. Lista på en vagns inmatade brister ordnade efter kontrollpunkt.

När webbvyer används i Android-applikationer så behöver man aktivera JavaScript för varje webbvy. Viss funktionalitet som finns inbyggt i JavaScript behöver också implementeras i Android-aktiviteterna som använder funktionaliteten. Exempel på sådan här funktionalitet är alert- och confirm-dialogrutorna i JavaScript. En alert-dialog skriver ut ett meddelande i en separat ruta som användaren behöver stänga med ett knapptryck. Confirm-dialogerna används för att användaren ska bekräfta eller avbryta ett val som gjorts. Ett vanligt tillfälle som confirm brukar användas på, är när användaren raderar något.

Inställningsmöjligheterna i HTML och JavaScript för dialogrutorna är starkt begränsade. I Android finns fler möjligheter att bestämma text i dialogrutan och på knapparna. I kodexempel 14 och figur 29 visas ett exempel på hur en bekräftelsedialog implementerats för slutgranskningsvyn.

Kodexempel 14. Bekräftelsedialog med knappar för att godkänna eller avbryta ett val som användaren gjort i webbapplikationen.

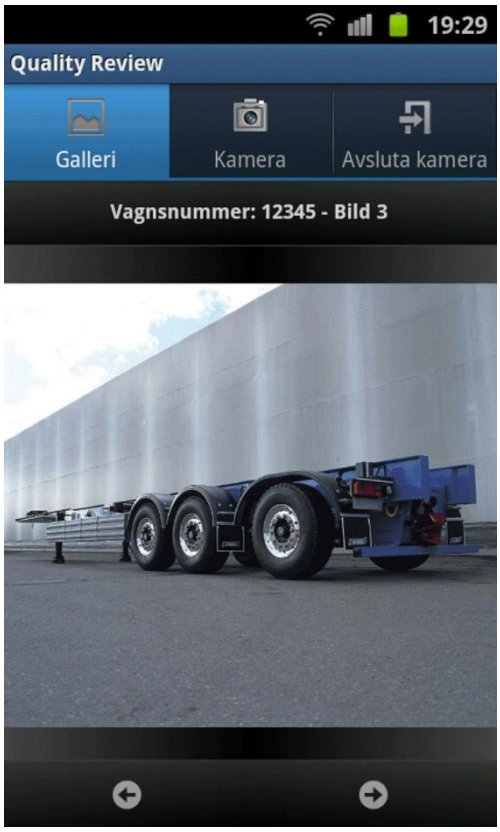
```
qrWebView.setWebChromeClient(new WebChromeClient() {
  @Override
  public boolean onJsConfirm(WebView view, String url,
  String message, final android.webkit.JsResult result) {
    new AlertDialog.Builder(myApp)
    .setTitle("Quality Review")
    .setMessage(message)
    .setIcon(R.drawable.qrquestion)
    .setPositiveButton(R.string.yes,
    new DialogInterface.OnClickListener()
    {
      public void onClick(DialogInterface dialog, int which)
      {
         result.confirm();
      }
    })
    .setNegativeButton(R.string.no,
    new DialogInterface.OnClickListener()
    {
      public void onClick(DialogInterface dialog, int which)
      {
         result.cancel();
      }
    })
    .create()
    .show();

    return true;
  }
});
```

Figur 29. Bekräftelsedialog öppnad från webbapplikationen i Android.

5.2.4 Kameravyn

Kameravyn är uppbyggd som en tabbvy med tre flikar i övre kanten. Flikarna är galleri, kamera och avsluta kamera. Den första fliken galleri är vald som standard.

Gallerifliken är uppbyggd som en webbvy. Webbvyn är länkad till ett HTML-galleri som finns i anslutning till webbapplikationen. Galleriet är byggt med hjälp av den fria JavaScript-komponenten PhotoSwipe som gör det lätt att navigera mellan bilderna. Galleriet för vagnsnummer 12345 med en tillhörande fotograferad bild visas i figur 30.



Figur 30. Galleriet för vagnsnummer 12345 där bild nummer tre är vald.

Bilderna hämtas direkt från filservern med en wildcardsökning för visning i galleriet, som vidare beskrivs i kapitel 5.3.4.

Den mittersta kamerafliken innehåller en startknapp för den inbyggda kameraapplikationen som finns i Android-enheten. På fliken finns också instruktioner för hur kameran ska användas.

Den inbyggda kameraapplikationen valdes eftersom den har många bra egenskaper som underlättar vid fotografering. Egenskaperna är bl.a. zoom och inställningar för blixten. Den inbyggda kameraapplikationen returnerar resultatet av fotograferingen till kameravyn. Kameravyn kontrollerar resultatet som antingen kan vara en avbruten fotografering eller en tagen bild. Om den inbyggda kameraapplikationen returnerar ett resultat med en fotograferad bild, påbörjas överföringen till filservern. Överföringen och applikationens överföringskö för bilder beskrivs vidare i kapitel 5.2.7.

Beroende på WiFi-anslutningens tillgänglighet och hastighet vid överföringstillfället kan bilden finnas tillgänglig i galleriet i princip omedelbart.

Väljer användaren att avsluta kameravyn med den tredje fliken ”Avsluta kamera”, avslutas kameravyn och webbapplikationssidan i slutgranskningsvyn som man tidigare befann sig på visas.

Menyn på kameravyn innehåller två alternativ. Användaren kan antingen återvända till granskningen genom att välja ”Fortsätt granska vagn” eller återvända till startskärmen med alternativet ”Till startskärm”. Figur 31 visar kamerafliken med kameravyns tillhörande meny.

Figur 31. Kameravyn där kamerafliken valts för vagnsnummer 12345.

5.2.5 Global applikationsklass

För att kunna hålla information tillgänglig mellan de olika vyerna och aktiviteterna skapades en global applikationsklass. Den globala klassen innehåller privata medlemsvariabler som kan hämtas och ges värde med get- och setmetoder. De medlemsvariabler som finns är bl.a. vagnsnummer för pågående granskad vagn och en lista på vagnar som granskning har påbörjats för.

För att komma åt den globala klassen görs en referens till den i varje aktivitet när aktiviteten skapas. De globala klassmetoderna kan sedan nås från varje aktivitets egna privata variabel för den globala klassen. Kodexempel 15 visar hur den globala klassen implementerats för att komma ihåg pågående granskad vagn i applikationen.

Kodexempel 15. Global applikationsklass för utbyte av data mellan vyerna. Innehåller en privat variabel, en metod för hämtning (get) och en för att uppdatera värdet på den privata variabeln (set).

```
import android.app.Application;

public class Globals extends Application
{
   // Private members
   private String currentTrailer;

   // Getters
   public String getCurrentTrailer()
   {
      return currentTrailer;
   }

   // Setters
   public void setCurrentTrailer(String trailerId)
   {
      currentTrailer = trailerId;
   }
}
```

För användning av den globala klassen använder man de publika metoderna för att spara undan ett vagnsnummer och hämta det. För att åskådliggöra detta visas i kodexempel 16 hur referensen till den globala klassen skapas och de publika metoderna används.

Kodexempel 16. Användning av den globala klassen.

```
private Globals qrApp = ((Globals)getApplicationContext());

// Sparar vagnsnummer 12345
qrApp.setCurrentTrailer("12345 ");

// Hämtar det påbörjade vagnsnumret till en strängvariabel
String trailer = qrApp.getCurrentTrailer();

// Skriver ut innehållet i variabeln trailer, d.v.s. 12345
System.out.println(trailer);
```

5.2.6 Hjälpklass

I Android-applikationen har en hjälpklass utvecklats där metoder som är användbara i flera vyer och aktiviteter finns. Klassen innehåller statiska metoder, vilket innebär de kan anropas utan att ha tillgång till ett objekt av klassen som metoderna finns i. Eftersom det är statiska metoder så behöver de också i allmänhet anropas med parametrar.

I hjälpklassen finns bland annat metoder för att hämta information från webbapplikationens API, med hjälp av Http requests.

Den metod som sänder fotograferade bilder till filservern från kameravyn finns också i hjälpklassen. Överföringen startar på en separat tråd, vilket gör att användaren kan fortsätta arbeta med applikationen utan fördröjningar eller avbrott. Bilden komprimeras först för att bli mindre till storleken. Bildens information konverteras därefter till Base64-kodning och skickas som Post-data till webbservern. Behandlingen av det skickade datat på webbservern beskrivs vidare i kapitel 5.3.4.

Den information som serialiserats till JSON-format konverteras i hjälpklassen med hjälp av Java till JSON-objekt från de textsträngar som hämtas med Http request. Genereringen av det JSON-serialiserade datat, som behandlas i Android-applikationen beskrivs mer i kapitel 5.3.2.

5.2.7 Bildkö med SQLite databas

I vissa fall när det gäller trådlösa anslutningar som t.ex. WiFi kan det förekomma störningar och långsamma överföringshastigheter. För att minimera risken att information går förlorad, som i detta fall är bilder, behövs en kö som håller reda på bilderna. För denna kö med bilder har en klass för Androids inbyggda databashanterare SQLite utvecklats.

Klassen för SQLite hanterar idag en databastabell, men kan enkelt byggas ut för fler om behov finns. Tabellen innehåller tre kolumner som beskriver bilderna som finns i kön och en kolumn med ett unikt löpande nummer. Kolumnerna som beskriver bilderna är vagnsnummer som fotograferats, sökväg till bilden på Android-enheten och bildens orientering.

I klassen finns tre förkompilerade SQLiteStatement-uttryck skrivna med SQL. Genom att använda SQLiteStatements kan uttrycken använda parametrar, som sänds när man anropas uttrycken, och därmed återanvändas flera gånger. Detta kommer väl till pass när nya bilder läggs till och tas bort från kön.

När användaren av applikationen fotograferat en bild så läggs den direkt i bildkön. Därefter påbörjas överföringen till filservern. Om överföringen lyckas tas bilden omedelbart bort från kön. Ifall första försöket misslyckas görs två nya försök vid behov. Om alla direkta försök att överföra bilden till filservern misslyckas lämnar bilden kvar i kön. Överföringen kan då bara startas manuellt från startvyns meny, av de bilder som lämnat kvar i kön.

5.3 Webbapplikation

5.3.1 Uppdatering av gränssnitt och funktionalitet

Gränssnittet i webbapplikationen är från år 2005 och har under utförandet moderniserats till vissa delar. Layouten är till stora delar uppbyggd med tabeller, vilket inte längre rekommenderas av World Wide Web Consortium (W3C) för layoutuppbyggnad. Det skulle dock innebära stora förändringar att bygga om alla sidor med divisions-taggar som är den rekommenderade standarden. Gränssnittsutvecklingen består därför främst av ett modernare utseende på knappar med bl.a. rundningar och färgövergångar.

En annan betydande modernisering av funktionaliteten i gränssnittet gjordes på kontrollflikarna. Eftersom Android-enheten har en pekskärm med bra stöd för multitouch infördes swipe-funktionalitet mellan de olika kontrollflikarna. Med detta menas att man genom att svepa med fingret över skärmen, till vänster eller höger, kan navigera mellan kontrollflikarna. Funktionaliteten implementerades med hjälp av ett Javascript som identifierar svepningen. Skriptet heter Touchwipe och använder sig i sin tur av Javascriptbiblioteket jQuery. Javascriptet innehåller kontroller att svepningen är tillräckligt lång för att utföra ett kommando, i detta fall för att byta kontrollflik.

I kodexempel 16 visas hur Touchwipe används för att reagera på svepningar från vänster till höger och köra en Javascript-funktion. Funktionen tar emot ett argument i form av den sida som användaren ska skickas till efter en svepning. Exemplet är implementerat på kontrollflik 200, och fungerar så att man flyttas till nästa flik om man sveper till vänster och till föregående flik om man sveper till höger.

Kodexempel 16. Registrerar touchwipe-komponenten på HTML-elementet som identifieras med #review när sidan laddats klart.

```
$(document).ready(function() {
   $("#review").touchwipe({
      wipeLeft: function() { go('kontroll_300.php') },
      wipeRight: function() { go('kontroll_100.php') },
      min_move_x: 150,
      preventDefaultEvents: false
   });
});
```

5.3.2 Kommunikation mellan webbapplikation och Android

För en fungerande applikationshelhet behöver Android-applikationen och webbapplikationen kunna utbyta information. Den viktigaste situationen när information behöver utbytas i applikationen är när man behöver ta reda på om granskning av en vagn påbörjats, och isåfall vilken vagn. Andra vanliga tillfällen när information behöver hämtas från webbapplikationen är för att hämta en lista med vagnar som granskning påbörjats för, och en vagns inmatade brister.

För utbyte av större mängder information och främst för hämtningar från Android-applikationen skapades API:n i webbapplikationen för ändamålet. För att få ett enhetligt format på det hämtade datat returnerar API:na vid lyckade förfrågningar JSON-serialiserat data. I kodexempel 17 hämtas en lista med vagnar där granskning har påbörjats från webbapplikationens databas.

Kodexempel 17. Hämtar en lista med vagnar där granskning har påbörjats.

```
<?php
$conn_id = odbc_connect("db_qualityreview", "user", "pass")

$sql = "SELECT vagnsnr FROM reklamation WHERE fardig = 1";
$result = odbc_exec($conn_id, $sql) or die("Query failed");

while(odbc_fetch_row($result))
{
   $arrTrailers[] = odbc_result($result, 1);
}
odbc_close($conn_id);

echo json_encode($arrTrailers);
?>
```

Listan byggs upp som en vektor i PHP och returneras tills sist som en serialiserad JSON-textsträng.

För mindre mängder information, t.ex. enstaka värden, används JavaScript för kommunikation mellan applikationerna. Denna metod ger även möjlighet att utbyta information i båda riktningarna. Genom anrop till webbvyn kan Android-applikationen köra JavaScript-funktioner som finns i webbsidan som visas i webbvyn.

Webbapplikationen kan kalla på ett JavaScript-interface som registreras på dess webbvy i Android-applikationen. I interfacet registreras funktioner som bestämmer vad som ska utföras i Android när de anropas. I kodexempel 18 har ett JavaScript-interface kallat qrJSInterface registrerats på den webbvy som visar HTML-sidan i exemplet. Funktionen updateReviewedTrailer i JavaScript-interfacet kallar på metoden setCurrentTrailer som visades i kapitel 5.2.5 i kodexempel 16.

Kodexempel 18. Anrop till funktionen updateReviewedTrailer i Android-applikationens Javascript-interface från en webbsida i en webbvy.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Anrop från webbvy till Android</title>
   <script type="text/javascript">
      function UpdateReviewedTrailer(trailer)
      {
         // Anrop till Javascript-interfacet i Android
         qrJSInterface.updateReviewedTrailer(trailer);
      }
   </script>
</head>
<body>
   <a href="javascript: UpdateReviewedTrailer('12345')">
      Uppdatera granskat vagnsnummer till 12345
   </a>
</body>
</html>
```

5.3.3 Visning av axeljustering

En ny sida för visning av axeljustering skapades i webbapplikationen. Kontrollen att axeljusteringen har blivit gjord, beror på om protokollet finns sparat på filservern eller inte.

Ett protokoll sparas på filservern efter att axeljusteringen gjorts på en vagn. Protokollet i form av en textfil får ett filnamn sammanfogat av vagnsnummer, ett unikt nummer och eventuellt ett projektnummer. Innehållet i textfilen består av data om axelinställningen före och efter gjord justering. Datat är separerat med tabbar och radbyten i textfilen.

Kontroll av axelprotokollet börjar med en wildcardsökning enligt två angivna mönster. Det som används för att identifiera rätt protokoll är vagnsnumret. Om ett protokoll hittas returneras ett booleskt värde, d.v.s. sant eller falskt. En av parametrarna till funktionen som gör wildcardsökningen fungerar som referensparameter. Referensparametern kan få ett värde i funktionen och returneras till den anropande PHP-sidan. Referensparametern får ett värde med filnamnet på det eventuellt funna protokollet.

När ett protokoll hittats kan processeringen av det textbaserade protokollet påbörjas med PHP. Inledningsvis läses hela protokollets innehåll in till en variabel. Datat består av fyra kolumner, separerade med tabbar. För att underlätta behandlingen av datat, överförs de tabbseparerade värdena till en flerdimensionell vektor. Med hjälp av vektorn kan datat lättare gås igenom radvis i en slinga.

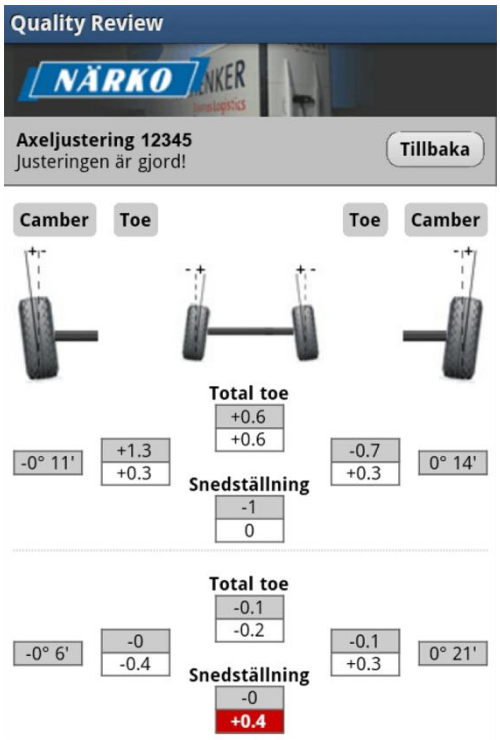
Varje rad i datat kan kommas åt med ett löpande index, börjandes från noll. Varje rad har i sin tur ett index per kolumn som innehåller ett värde. Det första indexet innehåller vilken typ av värde som finns på raden. Det andra indexet innehåller värdet före justeringen och det fjärde det justerade slutliga värdet.

Värdena från vektorn hämtas, ett objekt av en axelklass skapas, och värdena sparas undan i axelobjektets variabler. Klassen innehåller en konstruktor som tar en parameter som berättar vilken axel i ordningen det är fråga om. Axelklassen som innehåller variablerna för de värden som behövs för generering av protokollet, visas i kodexempel 19.

Kodexempel 19. Axelklass som kan innehålla värden från justeringsprotokollet.

```
<?php
class Axle
{
   // Properties
   public $sequence;

   public $leftCamber;
   public $leftToeBefore;
   public $leftToeAfter;

   public $totalToeBefore;
   public $totalToeAfter;
   public $skewBefore;
   public $skewAfter;

   public $rightToeBefore;
   public $rightToeAfter;
   public $rightCamber;

   // Constructors
   function __construct()
   {
      $a = func_get_args();
      $i = func_num_args();

      if (method_exists($this, $f='__construct'.$i))
      {
         call_user_func_array(array($this, $f), $a);
      }
   }

   function __construct1($seq)
   {
      $this->sequence = $seq;
   }
}
?>
```

Den grafiska genereringen av protokollet i webbapplikationen görs med HTML och utseendet ställs med CSS. Webbversionen av protokollet innehåller även bilder som beskriver värdena som skildras. Vid genereringen hämtas axelfabrikatet på vagnen från ett annat internt program på Närko, där streckkoder för axlarna skannats. Enligt axelfabrikatet hämtas därefter inmatade toleranser från databasen för snedställningen. Resultatet av den grafiska genereringen visas i figur 32.



Figur 32. Genererat axeljusteringsprotokoll i webbapplikationen. Snedställningen på den andra axeln är utanför toleranserna och därför markerad med röd bakgrund.

5.3.4 Kamera

I webbapplikationen har ett API utvecklats som tar emot information när bilder skickas från Android-applikationen. API:t tar emot vagnsnummer, bildorientering och bildens data när det anropas. Vid anrop skapas automatiskt, om det inte redan finns en mapp med vagnsnummer som namn, en ny mapp. Mappen skapas på Närkos filserver, dit användaren som kör webbapplikationens webbsida i IIS fått skrivrättigheter.

När mappen finns genereras ett unikt filnamn för bilden som ska sparas och det Base64-kodade bilddatat omkodas till en textsträng. Eftersom den inbyggda kameraapplikationen i den använda Android-enheten alltid levererar bilder i horisontellt läge, sänds telefonens orientering vid fotograferingen med till API:t. Med hjälp av Graphics Draw (GD) biblioteket i PHP, roteras bilderna till vertikala om telefonen varit vertikal vid fotograferingen. Om allt fungerar korrekt så sparas bilden till sist på filservern.

6 Resultat och diskussion

Resultatet av examensarbetet är en funktionerande applikationshelhet där den tidigare webbapplikationen för slutgranskning har integrerats i en Android-applikation. En kopia av den tidigare webbapplikationen har gjorts och anpassats för användning i Android. Den ursprungliga webbapplikationen finns kvar för att parallellt kunna användas vid behov, även i de äldre Windows Mobile-enheterna.

I den färdiga applikationen kan slutgranskarna kontrollera att axeljusteringen gjorts i produktion. Såvida justeringen har gjorts kan slutgranskaren hämta justeringsprotokollet och kontrollera att de justerade värdena är inom felgränserna. I slutskedet av granskningen kan slutgranskaren använda kameran för att fotografera leveransklara vagnar. Bilderna som tas vid slutgranskningen gör det lätt att få fram bilder på levererade produkter vid behov. Anställda som har tillgång till en dator ansluten till Närkos interna nätverk kan visa bilderna som sparats på en filserver i mappar namngivna efter vagnsnummer.

I skrivande stund är den nya applikationen på väg att börja användas i produktion.

6.1 Vidareutveckling

Den del av applikationen som använder kameran är nu anpassad för att endast fotografera en färdigt producerad enhet. Som vidareutveckling kunde kameran användas för många fler ändamål, t.ex. fotografering av brister när de matas in. Dessa bilder kunde sedan t.ex. användas för analys vid återkommande fel, för att förhindra att bristerna fortsättningsvis uppkommer.

Webbapplikationen innehåller en stor mängd programkod som upprepas flera gånger. En stor del av den kod som upprepas, kunde förenklas och återanvändas på flera sidor genom användning av klasser eller funktioner.

Databasen i Microsoft Access fungerar helt bra som databasmotor för en applikation av detta omfång. Databasen innehåller dock en del tabeller och

fält som inte är helt logiska. En grundlig genomgång av databasen kunde därför i något skede utföras för att få den normaliserad.

En utveckling av webbapplikationen har diskuterats som skulle innebära att slutgranskningsformuläret skulle bli mer dynamiskt. Utvecklingen skulle utföras till följd av nya lagkrav i Sverige. I vagnens planeringsskede kunde konstruktören konfigurera vissa specifika kontrollpunkter som skulle kontrolleras vid slutgranskningen. För visning av dessa nya kontrollpunkter skulle en ny flik infogas i webbapplikationen. Efter granskning kunde intyg fås över utförd kontroll för att visa att lagkravet uppfyllts.

6.2 Reflektioner

Det har varit en del utmaningar under utvecklingens gång. Till de större problemen kan räknas problem med Eclipse i kombination med pluginet för Android-utveckling, ADT. Vid några tillfällen har start av applikationen i emulatorn eller i den mobila enheten inte kunnat göras. Enligt Androids dokumentation ska de senaste uppdateringarna av Eclipse och ADT då installeras, samt en omstart av datorn göras. Detta hjälpte inte alla gånger, men det kunde fungera vid vissa omstarter. Detta gjorde att utvecklingsmiljön kändes något instabil, åtminstone i Windows-miljö.

I Androids arkitektur fanns även vissa egenheter som gjorde mig förbryllad till en början. Jag upptäckte bl.a. att varje gång enheten roteras så skapas hela aktiviteten pånytt. Vilket märktes när privata variabler i aktiviteter plötsligt var tomma när kameran användes. Dokumentation och exempel finns det dock gott om på nätet så lösningar för detta hittades, bl.a. den globala applikationsklassen. Den globala applikationsklassen är vid liv så länge programmet är igång. Denna klass var därför ändamålsenlig att spara information som man behöver komma åt i flera aktiviteter, eller som i detta fall efter rotering av enheten. Den globala applikationsklassen förklarades i kapitel 5.2.5.

I arbetet använde jag mig flera gånger av formatet JSON, som är ett ändamålsenligt sätt att överföra information. JSON-objekt innehåller endast den mest nödvändiga informationen som behövs för överföra och omvandla data. Formatet är enkelt att använda då det i de allra flesta

programmeringsspråk finns färdiga bibliotek implementerade för att serialisera data som JSON. Jag har upptäckt flera tillfällen där man kunde använda JSON. Ett av dessa tillfällen är när det finns behov av egenskaper från flera klasser samtidigt. Då kunde man skapa JSON-objekt som innehåller dessa egenskaper, i stället för att skapa nya klasser som innehåller samma egenskaper. Ett annat användningsområde där jag ser stor potential med JSON är för sparning av inställningar i en databas, i en databaskolumn per tabell. Det gäller speciellt sådana inställningar som kan användas vid programkörning och inte fordras som villkor vid hämtning från databastabellen.

6.3 Slutsatser

Detta examensarbete har varit intressant men även relativt krävande då jag varken hade erfarenhet av Java eller programmering för Android sedan tidigare. Detta har fördröjt utvecklingen till viss mån eftersom jag behövt ta reda på och lära mig mer vartefter utvecklingen framskridit. Till följd av att båda teknikerna är väldigt populära fanns mycket hjälp och exempel att fås från internet. Jag har också fått utveckla mina kunskaper inom några av de vanligaste teknikerna inom webbprogrammering.

Uppdragsgivaren är mycket nöjd med hur utvecklingen framskridit och hur det slutliga resultatet blev. Applikationen borde kunna användas utan större problem flera år framåt utan att några större åtgärder behöver göras.

7 Källförteckning

About the Eclipse Foundation. (u.d.). Hämtat från The Eclipse Foundation: http://www.eclipse.org/org/ den 16 februari 2012

ADT Plugin for Eclipse. (u.d.). Hämtat från Android Developers: http://developer.android.com/sdk/eclipse-adt.html den 18 februari 2012

Eclipse (programvara). (den 10 februari 2012). Hämtat från http://sv.wikipedia.org/wiki/Eclipse_(programvara) den 15 februari 2012

Flanagan, D. (2011). *JavaScript: The Definitive Guide* (6 uppl.). O'Reilly Media.

Getting started with Rails. (u.d.). Hämtat från Ruby On Rails: http://guides.rubyonrails.org/getting_started.html den 16 mars 2012

History of PHP. (u.d.). Hämtat från PHP: Hypertext Preprocessor: http://www.php.net/manual/en/history.php.php den 13 januari 2012

Introducing JSON. (u.d.). Hämtat från JSON.org: http://www.json.org den 13 januari 2012

Komatineni, S., MacLean, D., & Hashimi, S. Y. (2011). *Pro Android 3.* APress.

Lee, W.-M. (2011). *Beginning Android Application Development.* Wrox Press.

Niemeyer, P., & Knudsen, J. (2005). *Learning Java* (3 uppl.). O'Reilly Media.

Nokia Press Releases. (den 11 februari 2011). Hämtat från Nokia: http://press.nokia.com/2011/02/11/nokia-outlines-new-strategy-introduces-new-leadership-operational-structure den 21 mars 2012

NOMS - Närko Operational Management System. (u.d.). Hämtat från Närko Intranet: http://narko.intranet den 9 december 2011

Patrick, J. J. (2008). *SQL Fundamentals* (3 uppl.). Prentice Hall.

Petzold, C. (2010). *Programming Windows Phone 7.* MICROSOFT PRESS.

Powell, T. A. (2010). *HTML & CSS: The Complete Reference.* Osborne/McGraw-Hill.

Refsnes, H., Refsnes, S., & Refsnes, K. J. (2010). *Learn HTML and CSS with w3schools.* JOHN WILEY & SONS.

What is ODBC? (u.d.). Hämtat från Windows Dev Center: http://msdn.microsoft.com/en-us/library/windows/desktop/ms714591.aspx den 3 februari 2012