Utveckling av en Rapportgenerator

Henrik Tinnilä

Examensarbete

Informations- och medieteknik

2016

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	
Författare:	Henrik Tinnilä
Arbetets namn:	Utveckling av en Rapportgenerator
Handledare (Arcada):	Jonny Karlsson
Uppdragsgivare:	Arkkitehtuuritoimisto Visio

Sammandrag:
Det här examensarbetets syfte är att beskriva utvecklingsprocessen för en rapportgenerator för konditionsgranskningar av byggnader. Konditionsgranskningar behövs när man skall sälja eller köpa byggnader för att känna till eventuella risker. Att skriva rapporter om granskningarna är långsamt och därför vill man ha ett program som försnabbar processen. Detta program kommer att användas av Arkkitehtuuritoimisto Visio när de gör konditionsgranskningar på fältet. Syftet med programmet är att främst spara tid i skrivprocessen.

Rapportgeneratorn består av två delar. Första delen är en editor var man skapar rubriker och texter, samt frågor och svarsalternativ som sparas i en databas. Sedan skapas olika regler för vad som händer när man väljer ett svarsalternativ. I andra delen av programmet svarar man på frågorna. Enligt regler kommer det nya frågor på skärmen samt rubriker och texter sätts till en rapport. När man svarat färdigt har man en preliminär rapport man sedan kan fortsätta på.

Nyckelord:	Arkkitehtuuritoimisto Visio, Databaser, PHP, JavaScript, Programuveckling, webbsidor, HTML
Sidantal:	38
Språk:	Svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information and Media Technology
Identification number:	
Author:	Henrik Tinnilä
Title:	Development of a report generator
Supervisor (Arcada):	Jonny Karlsson
Commissioned by:	Arkkitehtuuritoimisto Visio

Abstract:
This thesis describes how a report generator was developed for Arkkitehtuuritoimisto Visio. Condition inspections are important to evaluate a buildings state when trying to buy or sell it. Writing reports about the inspection is slow, so Visio wants a program that will make it faster. This program will be used by Visio when they do condition inspections on the field. The main purpose of this program is to save time during the process of writing a report.

The report generator has two parts. The first part is an editor where the user creates titles and texts, and also questions and multiple answers which are all saved in a database. Then the user creates rules for what will happen when you choose a certain answer. In the second part of the program the user chooses answers for these questions. According to the rules new questions will appear on screen and titles and texts will be added to a report. When the user has answered all questions they will get a report that they can then continue to write on.

Keywords:	Arkkitehtuuritoimisto Visio, Databases, PHP, JavaScript, Program development, webbsites, HTML
Number of pages:	38
Language:	Swedish
Date of acceptance:	

INNEHÅLL / CONTENTS

1 Inledning 7

1.1 Bakgrund 7
1.2 Syfte och mål 8
1.3 Avgränsning 8

2 Planering 9

2.1 Krav 9
2.2 Design 9

3 Teknisk uppbyggnad 12

3.1 JavaScript 12
3.2 PHP 12
3.3 SQL/MYSQL 13
3.4 AJAX 14
3.5 JQUERY 14

4 Utveckling av programmet 15

4.1 Databasen 15
4.2 Editorn 18
4.2.1 Editorns fuktioner 18
4.2.2 Editorns Vyer 21
4.3 Raportgenerator 23
4.3.1 Raportgeneratorns funktioner 23
4.3.2 Raportgeneratorns vyer 25

5 Test 27

5.1 Testmetoder 27
5.1.1 Blackbox 27
5.1.2 Whitebox 27
5.2 Testandet av Rapportgeneratorn 28

6 Slutsatser 29

Källor 30

Bilagor 33

Figurer

Figur 1 Databasens uppbyggnad 15
Figur 2 Kod för göra en anslutning med databasen 18
Figur 3 Kod för att spara en ny rapport till databasen 19
Figur 4 Kod för att göra en rullgardinmeny av alla rubriker 20
Figur 5 Kod för att 2 frågor inte skall ha samma ordningsnummer 20
Figur 6 Editorns huvudvy 21
Figur 7 Vyn Saannot 22
Figur 8 Demo-vy var man svarar på frågor 25
Figur 9 Demo-vy var man ser den färdiga rapporten och kan sedan ladda ner den 26

Förtkortningar och terminologi

HTML = HyperText Markup Language

PHP = PHP: Hypertext Preprocessor

AJAX = Asynchronous JavaScript and XML

CSS = Cascading Style Sheets

SQL = Structured Query Language

MySQL = En databashanterare som använder sig av SQL

JavaScript = Ett skriptspråk som körs på klienten

JQuery = Ett JavaScript bibliotek som gör HTML och CSS behandling enklare

1 INLEDNING

Arkkitehtuuritoimisto Visio är ett arkitektföretag som är grundat år 1993. Företaget erbjuder tjänster i huvudstadsregionen med planering, byggtjänst och konditionsgranskningar.

När man gör fastighetstransaktioner behöver man veta i hurudant skick huset är. För att få veta det anställer man en neutral person för att göra en konditionsgranskning. Från konditionsgranskningen får säljaren och köparen veta om husets behov av reparation och säkerhetsrisker samt möjliga skade- och hälsorisker. Konditionsgranskningar görs av en expert inom byggnadsteknik. Man evaluerar byggnaden enligt observationer och info man får av ägaren. Man går igenom alla strukturer och lokaler enligt den grad som guiden säger. Man gör alltid en skriftlig rapport av resultaten. (visio)

En konditionsgranskning kostar ofta över 1000€. Den är dock värd att göra före försäljning av ett hus så man får reda på rätt försäljningspris och kan informera om eventuella fel som måste fixas. Säljaren ansvarar för dolda fel även om man gjort en konditionsgranskning. Konditionsgranskning har varit en verksamhet i endast tio år och det finns därför inga officiella kompetenskrav för att bli en granskare. En felgjord granskning kan ge en fel bild av husets skick och därför är det viktigt att hitta en kompetent granskare med erfarenhet. (kkv)

1.1 Bakgrund

Ett stort problem med att göra konditionsgranskningar är att rapporterna tar länge att skriva. Detta ledde till att chefen på Visio ville ha ett program som skulle försnabba processen. I programmet skulle granskaren svara på flervalsfrågor om byggnadens kondition. Beroende på vad användaren svarat skulle det komma nya frågor och text skulle sättas till en textfil. När man svarat klart skulle man ha en halvfärdig rapport som en word-fil med en hel del text som granskaren sedan skulle manuellt fortsätta på. Efter lite diskussion kom man också fram till att Visio vill själv kunna skapa nya frågor, svarsalternativ och

välja vad för texter tilläggs till rapporten. Man bestämde att man också måste skapa en editor för att göra det.

1.2 Syfte och mål

Syftet med arbetet är att skapa ett 2-delat program till uppdragsgivaren. Första delen är en editor var användaren skapar frågor med flera svarsalternativ, text till en rapport, och regler till frågorna. Andra delen av programmet skall användas när man gör konditions-granskningar. I den delen skall man svara på frågor om husets kondition. Enligt svaret man valt och reglerna man gjort i första delen skall programmet skapa en halvfärdig rapport till användaren. Detta sparar tid för personalen på Visio vilket är också det främsta syftet med programmet. Målet med detta examensarbete är att skapa en första fungerande version av programmet som är användbar och lättanvänd.

1.3 Avgränsning

Första versionen av programmet som gjorts inom ramen för examensarbetet har enkel grafik eftersom den inte är en kommersiell produkt och den bara skall vara så lättanvänd som möjligt. Det finns planer för mer avancerad funktionalitet som inte kommer att finnas i första versionen av programmet. Det skulle vara bra att kunna ta en bild med tabletten vid konditionsgranskningen och få bilden direkt med i rapporten. Programmet kunde ha en mängd färdiga frågor så man lättare kan bygga upp en grupp av dem för olika slags hus, t.ex. tegelhus, trähus och höghus. Mer avancerad grafik och möjligheten att ändra på programmets utseende skulle vara nyttigt. Det vore bra att ha en bild av byggnadens grundplan som man skulle kunna märka felpunkter på. Slutliga versionen borde också ha mera editeringsmöjligheter för gamla texter och rubriker.
Resten av examensarbetet är strukturerat enligt följande: I kapitel 2 beskrivs planeringen av programmet och vad för design principer som följs. I kapitel 3 beskrivs vad för tekniker som använts för att skapa programmet. Kapitel 4 berättar hur databasen är uppbyggd och beskriver också de viktigaste delarna av koden. Kapitel 5 redogör för hur programmet testats. Kapitel 6 innehåller slutsatser.

2 PLANERING

Den ursprungliga planen för projektet var att skapa en nätsida med frågor om en byggnads kondition. Efter att man svarat på frågorna skulle man få en halvfärdig word-fil man sedan skulle kunna fortsätta skriva på. Planen vidareutvecklades senare med egenskapen att själv kunna skapa frågorna/svarsalternativen och ändra på vad för text man sedan får. Behovet av någon typ av editor och en databas att spara data i togs även upp till diskussion. I den slutliga planen ingår tre delar: en editor var man skapar frågor, svarsalternativ, texter, och regler. Ett program var man svarar på frågorna, och en databas som länkar ihop regler till svarsalternativ och svarsalternativ till frågor etc.

2.1 Krav

Programmet skapas för personer som inte har tidigare kunskaper om data behandling, därför är det väldigt viktigt att programmet är enkelt att använda. Det skall finnas text som berättar vad man skall göra i varje skede av programmet och man skall lätt hitta nästa knapp man behöver trycka på. Programmets idé är att spara tid så det skall vara snabbt att använda det.

Med editor delen av programmet skall man kunna skapa rubriker med tillhörande text som hör till en rapport. Sedan skall man också kunna skapa frågor, svarsalternativ och regler av vad som händer när man väljer ett visst svarsalternativ. Frågor skall också höra till grupper så man får dem på skärmen en grupp per gång. Med rapportgeneratorn bör man kunna lätt välja svarsalternativ till frågor. När man väljer ett svarsalternativ skall det komma eventuella nya frågor och onödiga frågor skall tas bort. Efter att man svarat på allt skall programmet producera texter i ett word dokument som användaren själv kan fortsätta skriva på.

2.2 Design

I det här programmets design följs Galitz design principer (Galitz 2007). Han har listat 14 steg för att göra ett användarvänligt program. Härnäst diskuteras hur dessa principer tillämpas i detta projekt. Alla principer är inte lika viktiga i detta sammanhang, därför tas de viktigaste upp en i gången.

steg 1: Känn din användare. Personerna som skall använda programmet är inte så bra med datorer så designen skall vara lätt och rakt på sak. Varje vy skall gärna ha knapparna på ungefär samma platser så man inte behöver söka efter dem. Det skall också finnas text som berättar precis vad som skall göras steg för steg.

steg 2: Förstå programmets uppgift i företaget. Detta innehåller också definiering av vilka tutorials en användare behöver. Med programmet skapar man rubriker och texter som hör till dem. Sedan skapar man frågor med flera svarsalternativ. Till svarsalternativen sätter man regler. När man sedan väljer ett svarsalternativ sätter programmet enligt regeln text till en rapport. För att leda användaren genom processen finns det på sidan guide-texter som berättar steg för steg vad man skall göra.

steg 3: Förstå principerna av bra gränssnittsdesign. En bra vy skall vara skapad enligt vad användaren behöver och klarar av. Själva rapportgeneratorn skall användas på tablett så saker skall för läsbarhetens skull vara byggda vertikalt så man inte behöver scrolla åt sidorna. Med horisontal vy får man flera frågor på skärmen samtidigt. I designen skall man beakta att tabletten används som en anteckningsbok när man gör konditionsgranskningar i olika hus och lägenheter. Slutliga rapporten kompletteras i kontoret.

steg 4: Skapa menyer och navigeringsplaner. Eftersom den skall vara enkel finns nästa del alltid under den förra. Knapparna för att gå framåt och bakåt skall finnas på samma plats så man hittar dem snabbt.

steg 5: Välj passliga typer av vyer. Typiskt har ett program vyer som följer varandra. Det skall vara i logisk ordning så att användaren har lätt att använda programmet. I rapportgeneratorns editor kan man skapa de olika delarna i valfri ordning men i menyn är knappar till de olika vyerna i en fixad ordning. De är ordnade så att man inte hamnar i ett dödläge var användaren t.ex. försöker skapa svarsalternativ utan att ha skapat frågor först.

steg 6: Välj passande interaktions apparat. Ena delen av programmet används med mus och tangentbord medan andra delen har pekskärm. Eftersom programmet kommer

att vara simpelt kommer det inte att ha tangentbordsgenvägar. Målet är att uppleva båda programmen på samma sätt så känns det som en helhet.

steg 7: Välj passande skärmkontroller. När man använder pekskärmen kan inte saker vara för nära varandra. När pekskärmen används utanför kontoret måste kontrollerna vara lätt användbara och klara.

steg 8: Skriv klar text och meddelanden. Programmet som skapas kommer att ha instruktioner på sidan så användaren vet vad man skall göra. Instruktionerna bör vara tydliga utan tekniska förklaringar.

Steg 9 till 11 är mer för större, internationella projekt, och gäller inte det här arbetet.

steg 12: Välja passliga färger. Stor del av programmet kommer att ha lite färger. Vita delar med grå bakgrund så det är lättare för ögonen. Instruktionerna kommer att ha annan färg så man genast hittar dem när man kommer till vyn.

steg 13: Organisera sidans layout. Programmet är designat så att de kontroller som hör ihop är satta i samma CSS-box, som separerar de olika funktionerna från varandra.

steg 14: Testa, testa och testa igen. Version 1.0 kommer att testas av programmeraren och av en slutlig användare från Visio.

3 TEKNISK UPPBYGGNAD

I det här kapitlet beskrivs olika tekniker och programmeringsspråk som används i programmeringen av arbetet.

3.1 JavaScript

JavaScript är ett programmeringsspråk som körs på klientsidan. JavaScript används i webbutvecklande för att göra skript som interagerar med användaren, och ändrar på vad som syns på webbsidan utan att ladda om den. (Flanagan 2006)

JavaScript skapades i maj 1995 på 10 dagar av Brendan Eich. Först kallade man det för Mocha, sedan för LiveScript, efter det bytte man det till JavaScript eftersom Java var väldigt populär då. JavaScript är en viktig del av Ajax som låter nätsidor ladda data i bakgrunden från t.ex. SQL databaser och sedan visa ny data på skärmen utan att behöva ladda om sidan. Efter Ajax skapade man andra open source bibliotek såsom JQuery, Dojo och Mootools. Mera nyligen har man skapat Node.js som låtit oss använda JavaScript på serversidan. (W3 Schools)
JavaScript i sig användes inte mycket i det här projektet. Det användes endast för att ta fram mera hårdkodade textboxar så man kan få flera ”svarsalternativ” eller ”rubriker”. Det som användes mer var JQuery som är ett JavaScript bibliotek och AJAX som är en teknik som använder sig av JavaScript.
En fördel med JavaScript är att den körs på klientsidan vilket gör det snabbt. Ett problem med JavaScript är att det genast utförs när användaren kommer till en sida. Pga. detta kan man potentiellt utföra illvillig kod. (Jscripters JavaScript)

3.2 PHP

För att kommunicera med SQL databasen användes PHP i detta projekt.
PHP skapades 1994 av Rasmus Lerdorf, men det som man använder nuförtiden är väldigt annorlunda än första versionen. I början användes PHP till funktionalitet som lösenord till nätsidor och till att lätt skapa blanketter. PHP kunde inte i början användas för att skripta saker utan den kallade på C-kod när nätsidan kom till vissa taggar i HTML-koden.

Version 2.0 av PHP kom ut 1996, då talade man om det som en skript-språk på serversidan och berättade att den kunde användas för avancerade funktioner såsom webbkakor. Efter version 3.0 började PHP bli mer och mer populär och vid 2012 använde 77.8% av de 1 miljon mest besökta nätsidor PHP.
PHP används för att skapa dynamiskt webbinnehåll. PHP har inbyggda stöd för att skapa PDF, GIF JPEG och PNG filer. En av PHPs viktigaste funktioner är dens stöd för databaser. Med PHP kan man lätt hämta saker från, eller lätt spara data till databaser. PHP stöder databaser som MySQL, Oracle, Sybase och flera andra, också NoSQL-databaser som SQLite och MongoDB. (Tatroe 2014)

Basidén med programmet är att spara och hämta data från en databas. Till detta användes PHP. På Visio har man använt PHP tillsammans med MySQL i tidigare projekt och man bestämde att det skulle användas här också. Med detta behöver en person inte kunna allt för många programmeringsspråk för att ändra på deras program.

Fördelar med PHP är att den är gratis och den fungerar på Windows, Linux och Mac. PHP fungerar enkelt med MySQL, och det är en populär kombination så det finns mycket dokumentation om det. (webdesign)

3.3 SQL/MYSQL

Det är väldigt vanligt att nätsidor sparar och hämtar data från databaser på webbservrar. MySQL är en databas skapad av MySQL AB. MySQL är inte en del av PHP, men det är troligen den databasen som används mest med PHP. MySQL använder SQL för sitt datamanipulationsspråk och är väldigt populär för att den är open source och gratis. (Kofler 2005)
MySQL databasen användes i detta projekt för att först spara data som ”frågor” och ”svarsalternativ”, och sedan för att hämta dem. Man valde MySQL för att det är gratis och Visio använt det i andra projekt så det är lättare om man har samma typs databas.

Fördelar med MySQL är att den är säker och är lätt att använda med grundläggande SQL kunskaper. MySQL är också gratis och snabb. Snabbheten kommer med nackdelen att

den inte har alla funktioner som andra databaser men i en stor del av fallen har den allt man behöver. (Novell)

3.4 AJAX

AJAX låter webbutvecklare skapa sidor som smidigt hämtar data från servrar utan att ladda om sidan. Amazon.com var en av de första sidorna att använda Ajax. De använde Ajax i deras sökmotor A9.com. Med hjälp av Ajax kunde man visa extra information om websidor när man satte pekaren på länken till dem. Google har använt Ajax i skapandet av Google Maps, och Gmail. (Ford 2009)
AJAX används i projektet flera gånger till samma sak. När man väljer ett värde från rullgardinsmeny A skall programmet sedan fylla rullgardinsmeny B med relevanta värden.

En stor fördel med AJAX är att man inte behöver ladda om sidan för att få nytt innehåll. Ett problem med det är att ”back” och ”refresh” knapparna på browsern inte fungerar bra med det. Eftersom AJAX inte ändrar på URL:en så kan ”back” knappen föra användaren till en oväntad sida och ”refresh” tar en tillbaks till första vyn av sidan. (Jscripters AJAX)

3.5 JQUERY

JQuery är ett JavaScript bibliotek som skapades av John Resig år 2005. Han skapade funktioner som gjorde det lätt att hitta element på nätsidor och sedan tilldela beteenden åt dem. Namnet jQuery kommer från att hitta, eller ”querya” delar av websidor och sedan påverka dem med JavaScript. (Chaffer 2009)
JQuery används i projektet för att den är bättre än bara JavaScript på att ändra på dynamiskt skapade element. Den används till att dölja och ta fram skapade tilläggsfrågor och till att spara i local storage vilka svarsalternativ du valt.

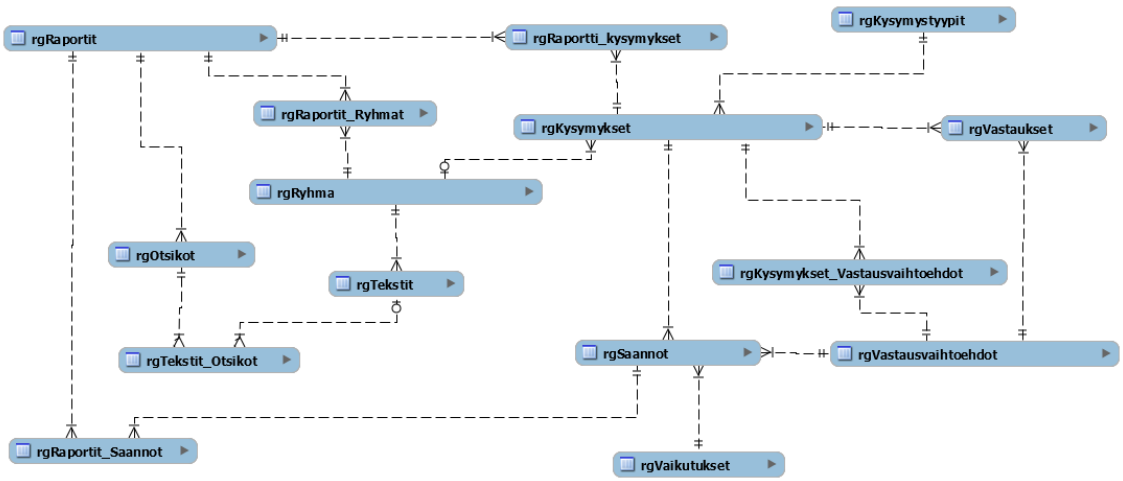
Fördelar med JQuery är att den har ett stort bibliotek av funktioner. Den har en stark gemenskap som har skapat flera JQuery insticksprogram. Med insticksprogrammen kan man lätt få specifika saker gjort utan att själv skapa för mycket ny kod. JQuery har inga stora nackdelar förutom att man måste ha en JQuery JavaScript fil för att köra JQuery kod. (Jscripters, Jquery)

4 UTVECKLING AV PROGRAMMET

Det här kapitlet beskriver hur programmet tekniskt fungerar. Väsentliga kodsnutter och lösningar kommenteras. Programmet består av tre delar: en databas där allt data sparas, en editor var man skapar frågor och text till databasen samt en frågedel var man hämtar frågor från databasen och svarar på dem.

4.1 Databasen

I det här projektet används en MySQL databas. I den sparas allt som skapas i editor-delen av programmet. Databasen och programmet använder sig av PHP för att kommunicera. Figur 1 visar databasens struktur medan bilaga 1 ger en mer detaljerad bild av databasen.



Figur 1 Databasens uppbyggnad

Som följande en beskrivning på databasens tabellers innehåll och deras relationer till varandra.

rgRaportit

Denna tabell är basen vart allt länkas. En rapport har ett eget namn och skaparens namn så man lättare kan hitta sin egen rapport. En färdig rapport innehåller en stor mängd rubriker och textsnuttar som kan kopplas ihop. Det skall också innehålla flera frågor med svarsalternativ som skall ha regler kopplade till dem. Enligt reglerna skall man sedan få flera frågor att svara på och bygga upp en texthelhet.

rgOtsikot

Denna tabell har rubriker. Rubrikerna har ett ordningsnummer som berättar i vilken ordning de sätts till den kopplade rapporten. De rubrikerna som är underrubriker har vid ’Otsikot_ylaotsikkoID’ fältet ID:n av överrubriken (bilaga 1).

rgTekstit

Den här tabellen har i sig alla texterna. Den innehåller själva texten, en beskrivning eller namn, och till vilken grupp den hör. Gruppen används för att lättare hitta texten när man gör regler. För att länka ihop texter till rubriker används följande tabell.

rgTekstit_Otsikot

I denna tabell länkar man ihop en text till en rubrik och ger till texten ett ordningsnummer. Idén med att man har en extern tabell för länkande är att på det här sättet kan en Text höra till flera Rubriker.

rgKysymykset

Den här tabellen har alla frågorna. De har ett ordningstal så man får dem i rätt ordning. Sedan har de själva frågan, namnet av frågan och vilken grupp den hör till. Frågans namn används för att lättare hitta den när man gör regler. Man svarar på frågor en grupp per gång. Man skapar frågor till grupper som ”källaren” och med att välja den gruppen kan man svara på alla frågor som hör dit.

rgKysymykset har också en rad om vad för slags fråga det är, radioknapp, rullgardinsmeny eller checkbox. De väljs från rgKysymystyypit var ID 1 representerar rullgardinsmeny, 2 radioknapp och 3 en Checkbox.

rgVastausvaihtoehdot

I den här tabellen finns endast ett svarsalternativens text. Det länkas ihop till frågor via **rgKysymykset_Vastausvaihtoehdot** så att samma svarsalternativ lättare kan länkas till flera frågor.

rgRyhma

I tabellen finns text- och frågegrupper. Text-grupperna används för att lättare hitta de texterna man vill lägga till eller ta bort när man skapar regler. Frågegrupper används för att koppla ihop sammanhängande frågor.

rgRaportit_Ryhmat

Frågor hör till grupper så att man hittar dem lättare. I programmet skapar man grupper såsom ”källaren” och ”övre våningen”. Vid skapandet länkar man också grupperna till en rapport. Sedan när man väljer den rapporten kan användaren byta mellan de relevanta grupperna och svara på frågor.

rgVastaukset

Det är en tabell som används när man håller på och väljer svarsalternativ till frågor i raportgeneratorn. Vid val av ett svarsalternativ sparas valet och frågan i tabellen. När man byter svarsalternativ updateras raden. Tabellen används sedan för att hämta alla frågor och svarsalternativ man svarat på och sedan bygger den upp en rapport enligt regler.

rgSaannot

Det är en av de viktigaste tabellerna. Den innehåller alla regler som användaren skapat. Användaren väljer vilken fråga och vilket svarsalternativ som orsakar regeln. Sedan väljer man vad som händer från rgVaikutukset. Lägger regeln till eller tar bort en fråga, eller lägger den till text eller en rubrik till rapporten. Efter det väljer man vilken fråga, rubrik eller text den påverkar. Till slut kan man också välja om någon annan regel måste ha hänt före den nuvarande regeln kan aktiveras.

rgRaportti_kysymykset

Tabellen kopplar ihop frågor till Rapporter. Det används för att lättare hitta de frågor man vill ha när man skapar regler.

4.2 Editorn

I editorn skapar användaren allt som behövs till en färdig rapport. Till en rapport hör rubriker, texter, frågor, svarsalternativ och regler. Editorn är till stor del HTML-former som skapas med PHP. I formerna fyller användaren i textboxar och väljer saker från rullgardinsmenyer. Rullgardinsmenyerna innehåller oftast värden från databasen och skapas därför med hjälp av PHP. Efter att man fyllt i allt och valt värden från listor sparas en ny t.ex. text i databasen. Detta händer också via PHP. All PHP-kod finns i filen functions.php och man kallar på den från de olika html-filerna.

4.2.1 Editorns fuktioner

Funktionen *connect* (figur 2) är den funktionen som används mest i programmet. I den tar programmet kontakt med databasen så att man sedan kan hämta och spara data. En stor del av funktionerna börjar med att kalla på *connect* och sedan göra något med databasen.

```
function connect(){
    $servername = "**********";
    $username = "**********";
    $password = "**********";
    $dbname = "**********";

    try {
        $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
        $conn->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }
    catch(PDOException $e)
        {
        echo $sql . "<br>" . $e->getMessage();
        }
        return $conn;
}
```

Figur 2 Kod för göra en anslutning med databasen

Funktionen *AddRapport* (figur 3) är en av de flera funktioner som sparar data i databasen. Den börjar med att kalla på funktionen *connect*, sedan använder den prepared statements för att förbereda databasen att ta emot nya värden. Sedan utförs SQL koden. Prepared statements används i det här projektet eftersom de skyddar emot SQL-injektioner.
Funktionerna för att spara texter, rubriker etc. använder sig av väldigt liknande funktioner.

```
function AddRapport($Name,$Writer){

    try{
    $conn = connect();

    $stmt = $conn->prepare("INSERT INTO rgRaportit (Raportit_nimi, Raportit_kirjoittaja)
        VALUES (:Name,:Writer)");
    $stmt->bindParam(':Name', $Name);
    $stmt->bindParam(':Writer', $Writer);
    $stmt->execute();

    return("New record created successfully");
    }
    catch(PDOException $e)
        {
        return("Connection failed: " . $e->getMessage());
        }
}
```

Figur 3 Kod för att spara en ny rapport till databasen

En stor del av tabellerna i databasen hör ihop. För att få dem länkade till varandra valde jag att göra rullgardinsmenyer med värden från tabellen något skall länkas till. Funktionen *TitleList* (figur 4) skapar en av dessa rullgardinsmenyer. Den börjar med att kalla på funktionen *connect*, sedan gör den ett SQL-anrop var den ber efter all info från rgOtsikot som innehåller alla rubrikerna. Efter det skapar funktionen början av en rullgardinsmeny och sätter till först en rad med texten ”Valitse Otsikko”. Sedan tar den varje rad som man fick som svar från SQL-anropet och gör det till en rad i rullgardinsmenyn. Varje rad har värdet av en rubriks ID, men i listan syns namnen. Efter att den är klar returneras listan och den sätts på skärmen.
I några delar av programmet väljer man först ett värde från en rullgardinsmeny vilket sedan skall fylla en annan rullgardinsmeny med relevanta värden. För att göra det har första listan en onClick som startar en JavaScript-funktion och skickar med värdet från första listan. JavaScriptet använder sig av AJAX för att starta en PHP-funktion som fyller andra rullgardinsmenyn enligt värdet som JavaScriptet tog emot.

```
function TitleList(){
    $conn = connect();

    $sql = "SELECT * FROM rgOtsikot";
    $stmt = $conn->query($sql);

    $dropdown = "<select name='OtsikkoID'>";
    $dropdown .= "\r\n<option value='0'>Valitse Otsikko</option>";
    foreach ($stmt as $row) {
        $dropdown .= "\r\n<option value='{$row['Otsikot_id']}'>{$row['Otsikot_nimi']}</option>";
        }
    $dropdown .= "\r\n</select>";
    return $dropdown;
}
```

Figur 4 Kod för att göra en rullgardinmeny av alla rubriker

Frågor, texter och rubriker har ordningsnummer i databasen. Rader som hör till samma rapport eller grupp skall ha unika ordningsnummer. För att göra det har programmet funktioner som skuffar bakåt raden som skulle ha samma nummer och alla rader efter den. Efter att man lagt till en ny fråga i databasen kallar man på funktionen *KysymyksetJarjestysNoPusher* (figur 5). Funktionen tar först kontakt med databasen. Sedan hämtar den alla rader som hör till samma grupp och har ett ordningsnummer som är samma eller större än den just tillagda raden. Efter det loopar den igenom alla raderna och skuffar dem 1 plats bakåt så att den nya ryms dit och på så vis har alla rader unika ordningsnummer.

```
function KysymyksetJarjestysNoPusher($RyhmaID,$Kysymys_Jno ){
    $conn = connect();
    $sql = "select * from rgKysymykset
    WHERE Kysymykset_ryhmaID=".$RyhmaID."
    && Kysymykset_jarjestysno>=".$Kysymys_Jno."";
    $stmt = $conn->query($sql);

    foreach ($stmt as $row) {
        $sql2 = "UPDATE rgKysymykset
                SET Kysymykset_jarjestysno = Kysymykset_jarjestysno + 1
                WHERE Kysymykset_id=".$row['Kysymykset_id']."";
                $stmt2 = $conn->query($sql2); }
}
```

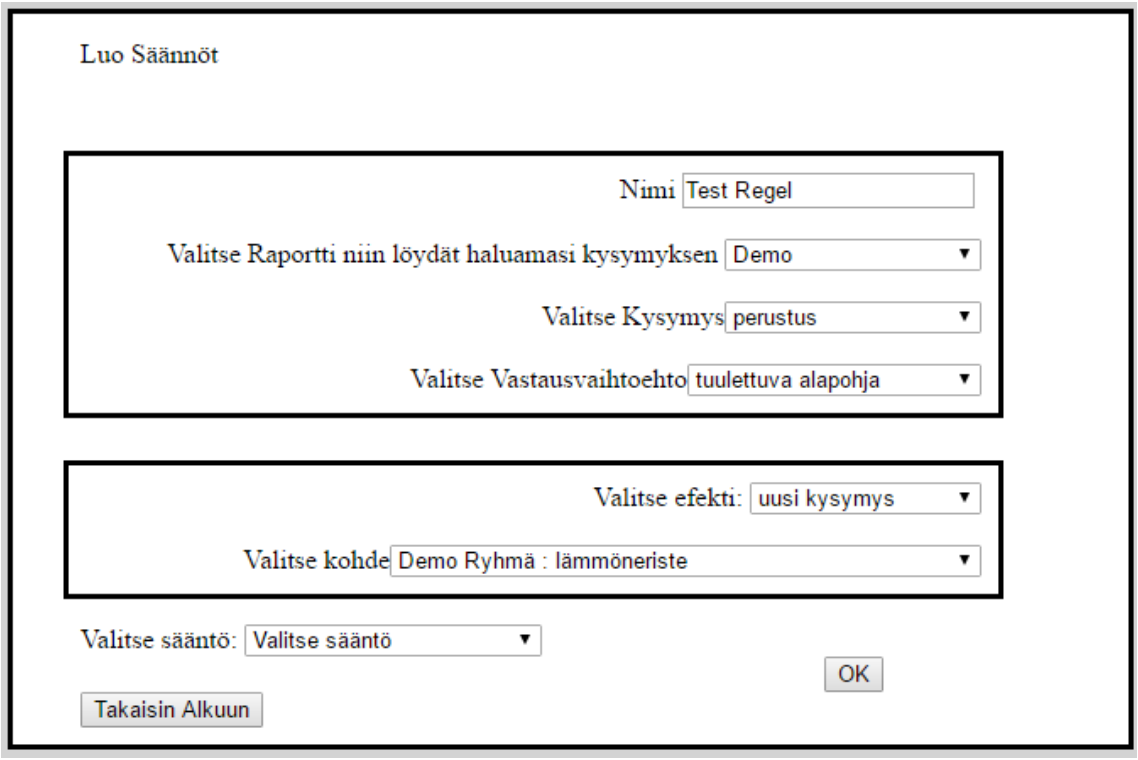
Figur 5 Kod för att 2 frågor inte skall ha samma ordningsnummer

4.2.2 Editorns Vyer

Figur 6 är start menyn för editorn. För att få gjort en färdig rapport går användaren igenom alla knapparna från ”Uusi Raportti” till ”Saannot”. Det finns också instruktioner till höger om rutan som inte visas i Figur 6. Snabbaste sätter att skapa en färdig rapport med alla delar är att först gå till ”Uusi Raportti”. Dit skapar användaren en Rapport vart man länkar allt. Sedan skapar man rubriker i ”Otsikko”, och texter i ”Teksti”. Efter det skapar man frågor i ”Kysymys” och svarsalternativ i ”Vastausvaihtoehto”. Till slut går man till ”Saannot” och skapar alla reglerna man vill ha.

Figur 6 Editorns huvudvy

I figur 7 syns vyn ”Saannot”. Jag valde att inte använda ÅÄÖ i namnen för att undvika möjliga problem. Den är den mest komplexa vyn i editorn. För att göra det lättare finns det instruktioner till höger, dessa syns inte i figur 7. Man börjar med att skapa ett namn till Regeln. Efter det väljer man Rapporten så man hittar den frågan man vill göra en regel för. Efter att man valt frågan väljer man svarsalternativet som orsakar regeln. Efter det väljer man regelns effekt: tar regeln fram eller gömmer en fråga, eller lägger den till en text eller rubrik till rapporten. När man väljer effekten fylls en rullgardinsmeny med passande objekt. Sedan är regeln färdig.



Figur 7 Vyn Saannot

Efter ”Saannot” finns ännu några knappar (Figur 6). I ”Esikatselu” kan man se på sina rubriker och texter och hur de byggs upp till en rapport. Sedan kan man också länka en fråga till en rapport och en text till en rubrik. Man kan också Ta bort en Rapport i ”Poista Raportti”.

4.3 Raportgenerator

I raportgeneratorn svarar man på frågorna man skapat. Användaren börjar med att välja rapporten man vill svara på. Sedan kommer man till en vy var man kan skriva i en textarea och välja mellan olika frågegrupper. När man väljer en grupp kommer alla frågor som hör till den gruppen på skärmen. Sedan väljer man svarsalternativ. Man kan byta mellan grupper utan att programmet glömmer dina svar. Efter att man svarat klart trycker man på ”jatka” och sedan ser man hur rapporten med rubriker och texter ser ut. Sedan kan man ladda ner den som en word-fil.

4.3.1 Raportgeneratorns funktioner

Funktionen *QuestionsOnScreen* (bilaga 2) tar in ID för den valda gruppen och sätter alla relevanta frågorna på skärmen. Först använder den sig av funktionen *connect* för att få kontakt med databasen. Sedan väljer den alla frågorna som hör till den valda gruppen. De frågor som tas fram med regler från andra svarsalternativ sätts på skärmen med gömd html div-tag. För varje fråga hämtar den alla svarsalternativen. Sedan kollar den vad för slags fråga det är, rullgardinsmeny, checkbox eller radioknapp. Den lägger till svarsalternativen enligt frågetypen och sätter en onClick-funktion till dem. onClick startar JavaScript funktionen *LoadRuleCheck* som kollar reglerna på det valda svarsalternativet.

LoadRuleCheck(bilaga 3) är en Jquery funktion som fungerar tillsammans med PHP funktionen *RuleCheck*(bilaga 4). När man väljer ett svarsalternativ startas *LoadRuleCheck* som sedan använder sig av AJAX för att starta PHP funktionen och skicka med ID:n av frågan och svarsalternativet man valt. *RuleCheck* börjar med att lägga till frågan och svarsalternativets ID till rgVastaukset. Om frågan redan har ett svar i rgVastaukset uppdateras endast svarsalternativet. För att lägga till saker i databasen används först *connect* funktionen, och sedan prepared statements. Efter att man lagt saker i databasen söker man efter regler som passar till kombinationen av frågans och svarsalternativets ID. Beroende på vad regeln vill göra sätter man data till en Array. Arrayn innehåller data om att lägga till eller ta bort en fråga och målfrågans ID. Sedan skickas arrayn som JSON enkodad tillbaks till Jquery funktionen *LoadRuleCheck*. *LoadRuleCheck* tar emot det och om

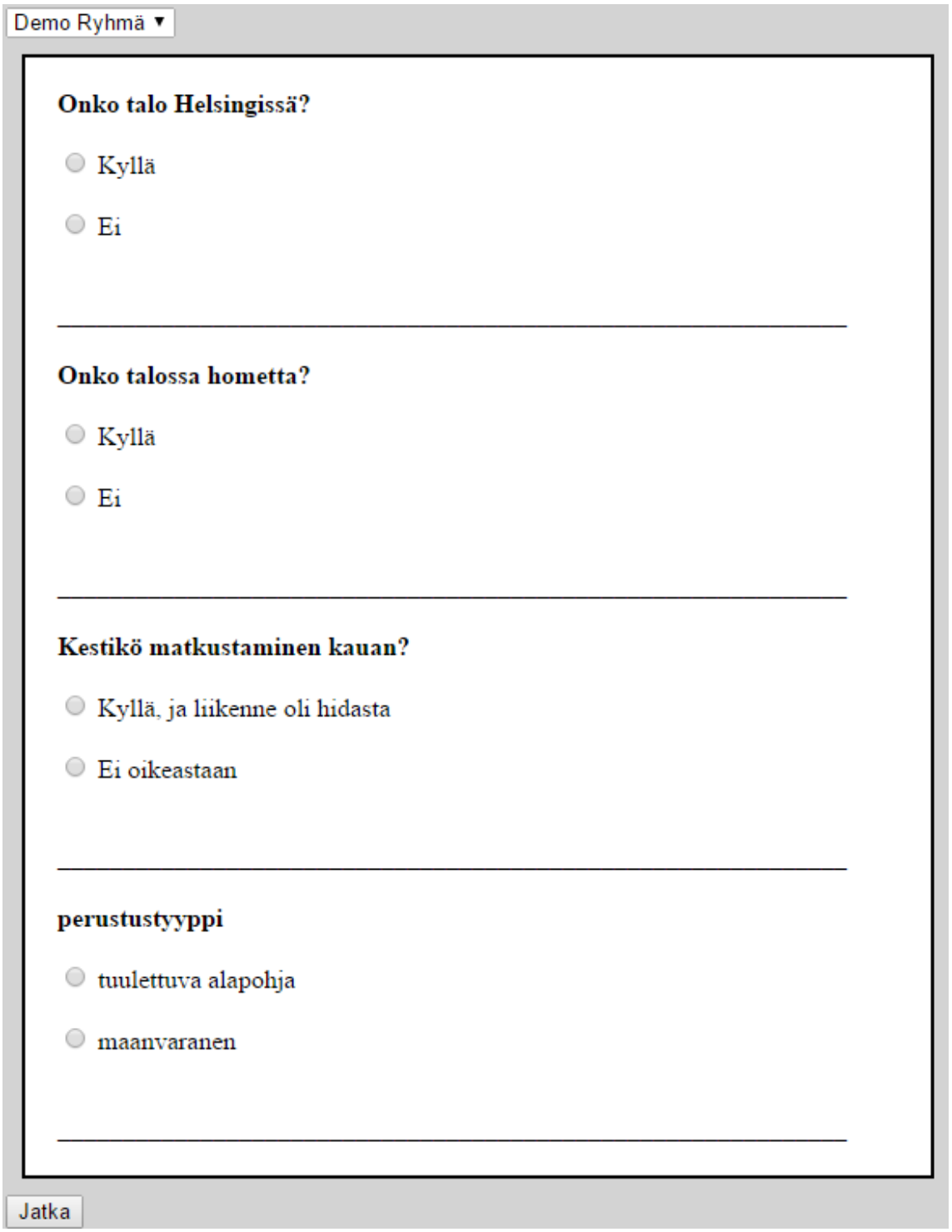
[’action’] är ”add question” sätter den målfrågans display till inline enligt ID:n. Om [’action’] är ”delete_question” sätts display till none.

Funktionen *Raportti*(bilaga 5) startas när användaren svarat på alla frågorna. Funktionen hämtar alla frågorna du svarat på från rgVastaukset och ser vilket svarsalternativ du valt senast. För varje svarsalternativ söker den efter regler från rgSaannot. Om en regel skall lägga till en rubrik sätts rubriken till en Array med rubriker, om den skall lägga till en text sätts texten till en Array med texter. Efter att den gått igenom alla frågorna du svarat på börjar den bygga upp rapporten från rubrikerna och texterna. Funktionen hämtar alla rubrikerna som inte har överrubriker och ordnar dem enligt ordningssnumret. För varje rubrik kollar den om den rubriken finns i Arrayn med rubriker, om den finns dit fortsätter loopen. Funktionen skriver ut rubriken på skärmen och börjar söka efter texter som hör till den från databasen. För varje text från databasen söker den igenom Arrayn med texter för samma ID. Om en text med samma ID finns där sätts den på skärmen. Sedan startar man funktionen *RaporttiAlaotsikot* för att söka igenom alla underrubriker. *RaporttiAlaotsikot* fungerar i stort sätt lika som *Raportti* med att den söker efter rubriker och texter. Om *RaporttiAlaotsikot* hittar en underrubrik kallar den på sig själv igen och söker efter underrubrikens underrubiker. Den fortsätter tills den inte mera hittar något. Sedan fortsätter *Raportti* att gå igenom rubrikerna i databasen och Arrayn tills den är klar.

För att hålla svarsalternativen valda i html Formen sparas läget till local storage med JQuery funktionen *LocalStorageRetrieve*.(bilaga 6) Vid val av ny grupp startas funktionen och gamla lägen hämtas. När man sedan väljer ett svarsalternativ från rullgardinsmenyer eller checkboxar sparas valet. Radioknappar sparas före man byter till nästa grupp.
För att få laddat ner en word fil användes en JQuery plugin. (Jqueryscripts)

4.3.2 Raportgeneratorns vyer

I första vyn väljer man endast vilken Rapport man arbetar på och kan deletera allt från local storage om man vill starta en ny session. Efter det kommer man till själva frågorna. Från rullgardinsmenyn byter man mellan grupperna och man svarar på frågorna.(Figur 8)



Figur 8 Demo-vy var man svarar på frågor

Efter att man svarat på alla frågor trycker man på ”jatka” då kommer man till en vy var man ser hur rapporten kommer att se ut. Sedan kan man ladda ner den som en word fil. (Figur 9)

Figur 9 Demo-vy var man ser den färdiga rapporten och kan sedan ladda ner den

5 TESTNING

Mjukvarutestning är processen att utvärdera en del av ett program och märka skillnader mellan input och förväntat output. Syftet med testning är att försäkra att programmets kvalitet är så god som möjligt. Det är bra att kontinuerligt testa ett program så man kan förhindra buggar i förväg. Det här kapitlet beskriver de vanligaste testmetoderna samt vilka som använts i testningen av detta program. Här diskuteras också vilka testmetoder som planeras att användas i framtida utveckling av programmet.

5.1 Testmetoder

Enligt Zafar (zafar 2012) delas testmetoder till två grupper, som är blackbox testning och whitebox testning.

5.1.1 Blackbox

Blackbox testning är funktionell testning var man fokuserar på vad för output programmet ger vid matning av input. Till *funktionell testning* hör också att testa att programmet gör vad den skall enligt kraven. Till blackbox testning hör också *system testning*. Det betyder att man testar programmet på olika operativ system. *Usability testning* görs från beställarens eller slutanvändarens perspektiv. Man testar om programmet är användarvänligt och hur lätt det är att lära sig att använda programmet. Dit hör också att testa hur effektiv användaren är efter att man lärt sig hur programmet fungerar.

I *Adhoc testning* testar man på programmet utan ordentlig planering eller dokumentation. Adhoc testning görs ofta när man inte har så mycket tid. Adhoc testning är endast effektivt om testaren förstår sig på hur programmet fungerar. (tutorialspoint)

5.1.2 Whitebox

Whitebox testning är strukturell testning var man tar i beakta hur programmets kod fungerar. Unit testning är att man testar att delar av programmets kod fungerar som det skall. Oftast testas det med att man skickar in till en funktion en input och man testar om outputten är det man väntar sig. (Zafar 2012)

5.2 Testandet av Rapportgeneratorn

I det här projektet har man mest använt sig av Adhoc testning. Eftersom programmet testas av samma person som gjort det har testaren en bra uppfattning om vad som kan gå fel och vad som skall testas.
Projektet hann inte testas så mycket men det finns planer att senare göra systemtestning. Programmet har skapats och testats på en windows 8 dator. Eftersom Rapportgenerator delen skall användas på tablett måste det testas för eventuella fel. Gärna på alla operativ-systemen Visio har i användning.
Det är också meningen att gör Usability testning när beställaren lärt sig använda program-met. Meningen var att skapandet av nya frågor inte skall vara för svårt och inte ta allt för länge. Man skall också spara tid när man använder raportgeneratorn och får en word fil.

6 SLUTSATSER

Det här examensarbetets mål var att skapa en Rapportgenerator för konditionsgranskningar av byggnader. Konditionsgranskningar behövs när man skall sälja eller köpa byggnader så man vet om eventuella risker. Projektet gjordes som beställningsarbete av Arkkitehtuuritoimisto Visio för att det är långsamt att skriva rapporter om konditionsgranskningar och de vill spara tid.

I det här slutarbetet skapades version 1.0 av Rapportgeneratorn samt en skriftlig rapport om utvecklingsprocessen. I den skriftliga rapporten beskrivs detaljerat planeringsprocessen och programmets uppbyggnad.
Programmets viktigaste funktioner är färdiga men det finns ännu små buggar i vissa delar. Med programmet kan man skapa en helhet av rubriker och texter, samt frågor, svarsalternativ och regler som påverkar dem. Man kan också svara på frågorna och enligt regler få till eller färre frågor på skärmen. Man får också laddat ner en word-fil med alla rubriker och texter byggda ihop som man sedan kan editera vidare på.

Programmet har fyllt sina funktionella krav, men eftersom användartestningen fattas har vi inte klar uppfattning om programmets användarvänlighet. Vidareutvecklingen kommer mest att vara buggfixande, finslipning och användartestning. Uppdragsgivaren har planer att använda programmet efter att det vidareutvecklats och testats.

KÄLLOR

Arkkitehtuuritoimisto Visio
http://www.arkkitehtuuritoimistovisio.fi/?cat=35&lang=fi
Hämtad 28.3.2016

Jonathan Chaffer, Karl Swedberg, Learning jQuery 1.3 (2009)
https://books.google.fi/books?hl=fi&lr=&id=R9JwBcAUHaoC&oi=fnd&pg=PT6&dq=Jquery+&ots=rbDFJMzAIP&sig=A6jhxBlxRfNSSXZGhFw5wDmuW5A&redir_esc=y#v=onepage&q&f=false Hämtad 12.12.2015

David Flanagan, Javascript The definitive guide (2006)
https://books.google.fi/books?hl=fi&lr=&id=k0CbAgAAQBAJ&oi=fnd&pg=PT6&dq=javascript&ots=O2pAjlgBpS&sig=ulSkPi68dM_b9umu-HmKbht-SEas&redir_esc=y#v=onepage&q&f=false Hämtad 12.12.2015

Jerry Lee Ford JR, Ajax programming for the absolute beginner (2009)
https://books.google.fi/books?hl=fi&lr=&id=sVgLAAAAQBAJ&oi=fnd&pg=PT7&dq=ajax+program-ming&ots=9FBxYOseQZ&sig=NZVODJTHXoxjPc0Ye_n_J_hGkBE&redir_esc=y#v=onepage&q&f=false Hämtad 12.12.2015

Wilbert O.Galitz, The essential guide to user interface design (2007)
https://books.google.fi/books?hl=fi&lr=&id=Q3Xp_Awu49sC&oi=fnd&pg=PR5&dq=gui+guide&ots=I_7ZBXbl27&sig=zVQbF5iRYAVCC9kprtr7IIvmqdo&redir_esc=y#v=onepage&q&f=false Hämtad 12.12.2015

Gosselin, Kokoska, Easterbrooks, PHP programming with mySQL (2010)
https://books.google.fi/books?hl=fi&lr=&id=Ifk7AAAAQBAJ&oi=fnd&pg=PT5&dq=php+programming&ots=IglLE8X_zd&sig=NTCtRm5wDDJdUo2H9Z69fDwN-txM&redir_esc=y#v=onepage&q&f=false Hämtad 12.12.2015

Jscripters
http://www.jscripters.com/jquery-disadvantages-and-advantages/
hämtad 23.4.2016

Jscripters
http://www.jscripters.com/ajax-disadvantages-and-advantages/
hämtad 23.4.2016

Jscripters
http://www.jscripters.com/javascript-advantages-and-disadvantages/
hämtad 23.4.2016

Jqueryscript
http://www.jqueryscript.net/other/Export-Html-To-Word-Document-With-Images-Using-jQuery-Word-Export-Plugin.html
hämtad 10.5.2016

KKV- Kilpailu- ja kuluttajavirasto
http://www.kkv.fi/Tietoa-ja-ohjeita/Ostaminen-myyminen-ja-sopimukset/asuntokauppa/kuntotarkastus/
Hämtad 28.3.2016

Michael Kofler, The definitive guide to MySQL 5 (2005)
https://books.google.fi/books?hl=fi&lr=&id=s_87mv-Eo4AC&oi=fnd&pg=PP1&dq=mysql+guide&ots=tdWyx6UFre&sig=svHTE7N6Tis6-rEvPKgq20etwoY&redir_esc=y#v=onepage&q&f=false Hämtad 12.12.2015

Novell
http://www.novell.com/documentation/nw65/web_mysql_nw/data/aj5bj52.html
hämtad 22.4.2016

Kevin Tatroe, Peter MacIntyre, Rasmus Lerdorf, Programming PHP (2014)
https://books.google.fi/books?hl=fi&lr=&id=w69fcU5dHgAC&oi=fnd&pg=PR5&dq=rasmus+lerdorf+programming+php&ots=5oSXtZLpbz&sig=jif-3tOf-KcwOUAV6jwnx8qnrqd4&redir_esc=y#v=onepage&q&f=false chapter 1

Tutorialspoint
http://www.tutorialspoint.com/software_testing_dictionary/adhoc_testing.htm
hämtad 15.5.2016

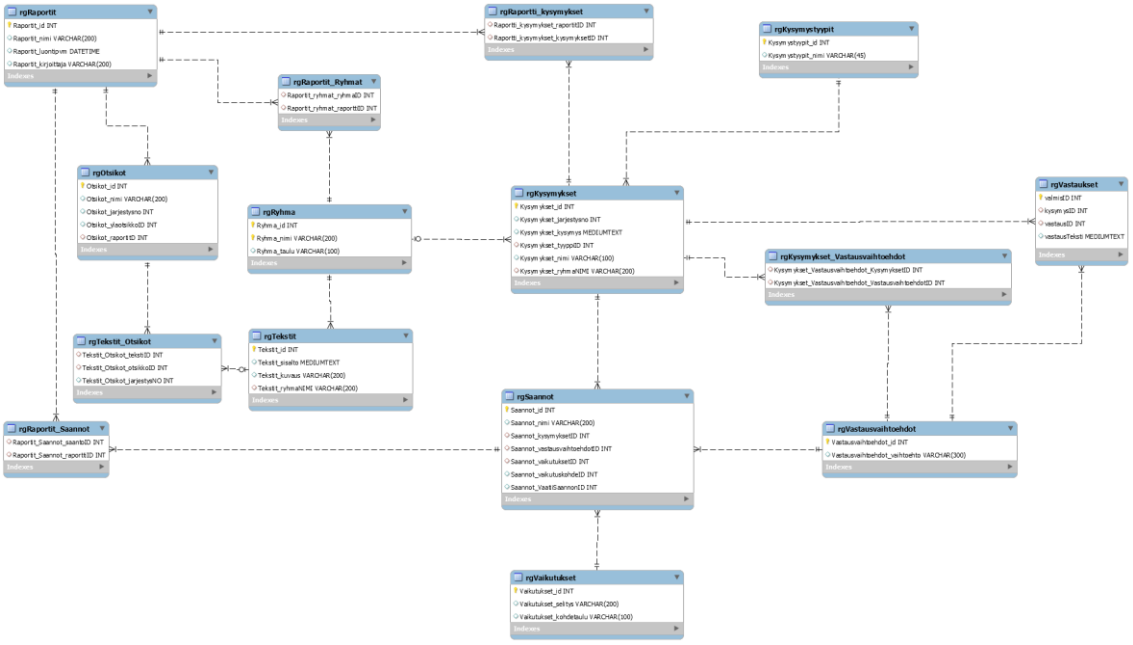
W3Schools
https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript
Hämtad 14.4.2016

Webdesign
http://www.webdesign.org/web-programming/php/advantages-of-php-programming.21905.html
hämtad 23.4.2016

Rehman Zafar (2012)
http://www.codeproject.com/Tips/351122/What-is-software-testing-What-are-the-different-ty , hämtad 15.5.2016

BILAGOR

Bilaga 1 SQL databasen, med detaljer



Bilaga 2 QuestionsOnScreen

```
function QuestionsOnScreen($ID1){
    session_start();
    $conn = connect();

    $sql2 = "SELECT * FROM rgKysymykset
    WHERE Kysymykset_ryhmaID=".$ID1."";
    $stmt2 = $conn->query($sql2);

    foreach ($stmt2 as $row2) {
        if($row2['Kysymykset_Hidden'] == 1){
            echo "<div id='".$row2['Kysymykset_id']."' style='display: none'
            name='".$row2['Kysymykset_id']."'>";
        }
        else{
            echo "<div id='".$row2['Kysymykset_id']."'
            name=".$row2['Kysymykset_id']."'>";
        }

        echo "<strong>".$row2['Kysymykset_kysymys']."</strong>   ";
        echo "\n<br />\n<br />";
        $sql3 = "SELECT * from rgKysymykset_Vastausvaihtoehdot
        WHERE Kysymykset_Vastausvaihtoehdot_KysymyksetID=".$row2['Kysymykset_id']."";
        $stmt3 = $conn->query($sql3);

        $KysymysID_Current = $row2['Kysymykset_id'];
        $dropdown = "<select class='Dropdown' id='".$row2['Kysymykset_id']."'
        name='".$row2['Kysymykset_id']."'
        onchange=LoadRuleCheck(this.value,".$KysymysID_Current.")>";
        $dropdown .= '<option value="0">"Valitse"</option>';

        foreach ($stmt3 as $row3){
            $sql4 = "SELECT * from rgVastausvaihtoehdot
            WHERE Vastausvaihtoehdot_id=".$row3['Kysymykset_Vastausvaihtoehdot_VastausvaihtoehdotID']."";
            $stmt4 = $conn->query($sql4);

            foreach ($stmt4 as $row4){
                if($row2['Kysymykset_tyyppiID'] == 1){
                    $dropdown .= '<option value="'.$row4['Vastausvaihtoehdot_id'].'">
                    '.$row4['Vastausvaihtoehdot_vaihtoehto'].'</option>';
                }

                else if($row2['Kysymykset_tyyppiID'] == 2){
                    echo '<input type="radio" name="'.$row2['Kysymykset_id'].'" value="'.$row4['Vastausvaihtoehdot_id'].'"
                    id="'.$row4['Vastausvaihtoehdot_id'].'"   onClick="LoadRuleCheck(this.value,'.$KysymysID_Current.')" >
                    '.$row4['Vastausvaihtoehdot_vaihtoehto'].'<br>';
                    echo "\n<br />";
                }

                else if($row2['Kysymykset_tyyppiID'] == 3){
                    echo '<input type="checkbox"  name="'.$row2['Kysymykset_id'].'"
                    value="'.$row4['Vastausvaihtoehdot_id'].'"
                    onClick="LoadRuleCheck(this.value,'.$KysymysID_Current.')"      >
                    '.$row4['Vastausvaihtoehdot_vaihtoehto'].'';
                    echo "\n<br />";
                }
            }
        }

    if($row2['Kysymykset_tyyppiID'] == 1){
        $dropdown .= "</select>";
        echo $dropdown;
    }

    echo "\n<br />";
    echo "_____________________________________________________________";
    echo "</div>";
    }

}
```

Bilaga 3 LoadRuleCheck

```
function LoadRuleCheck(id,id2){
    if (id.length == 0) {
            document.getElementById('Lista2').innerHTML = '';
    return;
    }
    if (window.XMLHttpRequest) {
         xmlhttp = new XMLHttpRequest();}
    else {
         xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");}
        xmlhttp.onreadystatechange = function() {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                document.getElementById("Lista2").innerHTML = xmlhttp.responseText;}};
        xmlhttp.open("POST","functions.php?ID="+id+"&ID2="+id2,true);
        xmlhttp.send();
        $.ajax({
          url: 'functions.php',
          type: 'post',
          data: {'id1': id, 'id2': id2},
          dataType: 'json',
          success: function(data) {

            if(data['action'] == "add_question") {

              $('#' + data['Kysymykset_id']).css('display', 'inline');
            }

            else if(data['action'] == "delete_question") {
              $('#' + data['Kysymykset_id']).css('display', 'none');
            } else {
              console.debug('jquery ajax response action either missing or invalid');
            }
          },
          error: function(xhr, desc, err) {
            console.log(xhr);
            console.log("Details: " + desc + "\nError:" + err);
          }
        });

}
```

Bilaga 4 RuleCheck (nästa sida)

```
    function RuleCheck($ID,$ID2,$jqueryajaxcall = false){
        session_start();
        $jqueryajaxdata = array();
        $conn = connect();  /


        $VALMISID = $_SESSION['VastauksetID'];
        $raporttiTEMP = $_SESSION['RaportinID_Generaattori'];
        $ID;
        $ID2;
        $asdasd = true;
        $TEMPTEXT = $_SESSION['Text'];
    try{
    $sql = "SELECT * FROM rgVastaukset
            WHERE valmisID=".$VALMISID." &&
            kysymysID=".$ID2."
            ";
        $stmt = $conn->query($sql);

        foreach($stmt as $row) {

            $sql = "UPDATE rgVastaukset
            SET vastausID=".$ID."
            WHERE valmisID=".$VALMISID." &&
            kysymysID=".$ID2."
            ";
            $stmt = $conn->query($sql);
            $asdasd = false;
        }
    }
    catch(Exception $e) {
      echo 'Message: ' .$e->getMessage();
    }


    if($asdasd == true){
    try{
$stmt = $conn->prepare("INSERT INTO rgVastaukset (valmisID, kysymysID, vastausID, vastausTeksti)
    VALUES (:valmis,:Kysymys,:Vastaus,:Teksti)");
$stmt->bindParam(':valmis', $VALMISID);
$stmt->bindParam(':Kysymys', $ID2);
$stmt->bindParam(':Vastaus', $ID);
$stmt->bindParam(':Teksti', $TEMPTEXT);
$stmt->execute();



    }
catch(PDOException $e)
    {

    }
}

    $sql = "SELECT * FROM rgRaportit_Saannot
        WHERE Raportit_Saannot_RaporttiID=".$_SESSION['RaportinID_Generaattori']."
        ";
    $stmt = $conn->query($sql);


    foreach($stmt as $row) {


        $sql2 = "SELECT * FROM rgSaannot
            WHERE Saannot_id=".$row['Raportit_Saannot_SaantoID']." && Saannot_kysymyksetID=".$ID2."
            && Saannot_vastausvaihtoehdotID=".$ID."
            ";
        $stmt2 = $conn->query($sql2);

        foreach($stmt2 as $row2) {
            if($row2['Saannot_vaikutuksetID'] == 1){

                $xmlhttpdata .= "\n<br />";
                $xmlhttpdata .= "Saanto haluaa lisätä Kysymyksen";
                $sql3 = "SELECT * from rgKysymykset
                WHERE Kysymykset_id = ".$row2['Saannot_vaikutuskohdeID']."";
                $stmt3 = $conn->query($sql3);

                foreach($stmt3 as $row3){


      $jqueryajaxdata['action'] = 'add_question';
      $jqueryajaxdata['Kysymykset_id'] = $row3['Kysymykset_id'];
      $jqueryajaxdata['Kysymykset_kysymys'] = $row3['Kysymykset_kysymys'];
      $jqueryajaxdata['Kysymykset_jarjestysno'] = $row3['Kysymykset_jarjestysno'];
                }
            }
            else if($row2['Saannot_vaikutuksetID'] == 2){


                $sql3 = "SELECT * from rgKysymykset
                WHERE Kysymykset_id = ".$row2['Saannot_vaikutuskohdeID']."";
                $stmt3 = $conn->query($sql3);

                foreach($stmt3 as $row3){

      $jqueryajaxdata['action'] = 'delete_question';
      $jqueryajaxdata['Kysymykset_id'] = $row3['Kysymykset_id'];
      $jqueryajaxdata['Kysymykset_kysymys'] = $row3['Kysymykset_kysymys'];
      $jqueryajaxdata['Kysymykset_jarjestysno'] = $row3['Kysymykset_jarjestysno'];
                }
            }

            }

        }
        if ($jqueryajaxcall === true) {
          echo json_encode($jqueryajaxdata);
          error_log('palautetaan jqueryajaxdataa');
        } else {

          echo $xmlhttpdata;
          error_log('palautetaan xmlhttpdataa');
        }
    }
```

Bilaga 5 Raportti (nästa sida)

```php
function Raportti(){

    session_start();
    $RaporttiID = $_SESSION['RaportinID_Generaattori'];
    $VastauksetSession = $_SESSION['VastauksetID'];
    $OtsikkoArray = array();
    $TekstiArray = array();
    $conn = connect();

    $sql = "select * from rgVastaukset
    WHERE valmisID=".$VastauksetSession."";
    $stmt = $conn->query($sql);

    foreach($stmt as $row){
            $Kysymys = $row['kysymysID'];
            $Vastaus = $row['vastausID'];

            $sql2 = "SELECT * from rgSaannot
            INNER JOIN rgRaportit_Saannot
            ON rgSaannot.Saannot_id = rgRaportit_Saannot.Raportit_Saannot_SaantoID
            WHERE rgSaannot.Saannot_kysymyksetID=".$Kysymys." &&
            rgSaannot.Saannot_vastausvaihtoehdotID=".$Vastaus." &&
            rgRaportit_Saannot.Raportit_Saannot_RaporttiID =".$RaporttiID."";
            $stmt2 = $conn->query($sql2);

            foreach($stmt2 as $row2){

                if($row2['Saannot_vaikutuksetID'] == 3){
                    $sql3 = "select * from rgTekstit
                    WHERE Tekstit_id = ".$row2['Saannot_vaikutuskohdeID']."";
                    $stmt3 = $conn->query($sql3);

                    foreach($stmt3 as $row3){
                        $TekstiArray[] = array($row3['Tekstit_id'],$row3['Tekstit_sisalto']);
                    }
                }
                else if($row2['Saannot_vaikutuksetID'] == 5){
                    $sql3 = "select * from rgOtsikot
                    WHERE Otsikot_id = ".$row2['Saannot_vaikutuskohdeID']."";
                    $stmt3 = $conn->query($sql3);

                    foreach($stmt3 as $row3){
                        $OtsikkoArray[] = array($row3['Otsikot_id'],$row3['Otsikot_jarjestysno'],
                        $row3['Otsikot_nimi'],$row3['Otsikot_ylaotsikkoID']);
                    }
                }
            }
    }

    sort($OtsikkoArray);

    $sql = "SELECT * FROM rgOtsikot
        WHERE Otsikot_ylaotsikkoID=0
        ORDER BY Otsikot_jarjestysno";
        $stmt = $conn->query($sql);

        $length1 = count($OtsikkoArray);
        $length2 = count($TekstiArray);

    try{
        foreach ($stmt as $row) {
                for ($i = 0; $i < $length1; $i++) {

                if($OtsikkoArray[$i][0] == $row['Otsikot_id']){
                        echo "<strong>".$row['Otsikot_nimi']. "  \n<br />\n<br />  </strong>  ";
                        $sql2 = "SELECT * from rgTekstit_Otsikot
                        WHERE Tekstit_Otsikot_otsikkoID=".$row['Otsikot_id']."
                        ORDER by Tekstit_Otsikot_jarjestysNO";
                        $stmt2 = $conn->query($sql2);

                        foreach ($stmt2 as $row2) {
                            for ($j = 0; $j < $length2; $j++) {
                                if($TekstiArray[$j][0] == $row2['Tekstit_Otsikot_tekstiID']){
                                    $sql3 = "SELECT * from rgTekstit
                                    WHERE Tekstit_id=".$row2['Tekstit_Otsikot_tekstiID']."";
                                    $stmt3 = $conn->query($sql3);

                                    foreach ($stmt3 as $row3){
                                        echo  nl2br($row3['Tekstit_sisalto']) .  "\n<br />\n<br /> ";
                                    }
                                }
                            }
                        }

                    $asd = $row['Otsikot_id'];
                    RaporttiAlaotsikot($asd, $OtsikkoArray, $TekstiArray);
                }
            }
        }
    }
    catch(Exception $e){echo 'Caught exception: ',  $e->getMessage(), "\n";}
}

function RaporttiAlaotsikot($asd, $OtsikkoArray, $TekstiArray){

    $length1 = count($OtsikkoArray);
    $length2 = count($TekstiArray);

    $conn = connect();


    $sql = "SELECT * FROM rgOtsikot
        WHERE Otsikot_ylaotsikkoID=".$asd."
        ORDER BY Otsikot_jarjestysno";
        $stmt = $conn->query($sql);
```

Fortsätter på samma sätt som Raportti

Bilaga 6 LocalStorageRetrieve

```
 $("select").click(function()
{
    $('input[type=radio]').each(function()
    {
        localStorage.setItem(
            'radio_' + this.id, JSON.stringify({checked: this.checked})
        );
    });
});

function LocalStorageRetrieve(){

$('.Dropdown').change(function () {
    var key = $(this).attr('id');
    var value = $(this).val();
    localStorage.setItem(key, value)
});

(function() {
    var boxes = document.querySelectorAll("input[type='checkbox']");
    for (var i = 0; i < boxes.length; i++) {
        var box = boxes[i];
        if (box.hasAttribute("value")) {
            setupBox(box);
        }
    }

    function setupBox(box) {
        var storageId = box.getAttribute("value");
        var oldVal    = localStorage.getItem(storageId);
        console.log(oldVal);
        box.checked = oldVal === "true" ? true : false;

        box.addEventListener("change", function() {
            localStorage.setItem(storageId, this.checked);
        });
    }
})();
    (function()

{
    $('input[type=radio]').each(function()
    {
        var state = JSON.parse( localStorage.getItem('radio_'  + this.id) );

        if (state) this.checked = state.checked;
    });
})();
```