Migration av avtalshanteringssystem

Tim Strengell

Examensarbete för ingenjörsexamen (YH)

Utbildningsprogrammet för informationsteknik

Vasa 2016

EXAMENSARBETE

Författare: Tim Strengell

Utbildningsprogram och ort: Informationsteknik, Vasa

Handledare: Susanne Österholm

Titel: *Migration av avtalshanteringssystem*

Datum: 14.4.2016 Sidantal: 34

Abstrakt

Detta examensarbete gjordes på uppdrag av Wärtsilä Finland Oy. Examensarbetet innefattade utvecklingen av ett webbaserat avtalshanteringssystem som ersätter ett äldre system. Systemet används för att visa och hantera avtal inom Wärtsilä och innehåller även ett interaktivt rapporteringsverktyg. Det nya systemet behåller funktionaliteten från det äldre systemet och innehåller även några förbättringar.

Målet med examensarbetet var att skapa ett uppdaterat system med mer konsistent användargränssnitt och bättre strukturerad kod. Examensarbetet förverkligades med hjälp av moderna webbteknologier som HTML5, CSS, JavaScript med jQuery, AJAX, PHP och SQL. Resultatet blev ett uppdaterat avtalshanteringssystem som är lättare att underhålla.

Språk: svenska Nyckelord: webbapplikation, PHP, AJAX, jQuery

OPINNÄYTETYÖ

Tekijä: Tim Strengell

Koulutusohjelma ja paikkakunta: Tietotekniikka, Vaasa

Ohjaaja: Susanne Österholm

Nimike: *Sopimushallintajärjestelmän siirto*

Päivämäärä: 14.4.2016 Sivumäärä: 34

Tiivistelmä

Tämä opinnäytetyö tehtiin Wärtsilä Finland Oy:n toimeksiannosta. Opinnäytetyö sisältää Internet-pohjaisen sopimushallintajärjestelmän kehittämisen PHP:n avulla, joka korvaa vanhemman järjestelmän. Järjestelmää käytetään sopimuksien näyttämiseen ja hallitsemiseen Wärtsilässä ja se sisältää myös interaktiivisen tiedotustyökalun. Uusi järjestelmä säilyttää vanhan järjestelmän toiminnot ja sisältää myös muutamia parannuksia.

Opinnäytetyön tavoite oli luoda uudistunut järjestelmä, jossa on johdonmukaisempi käyttöliittymä ja paremmin jäsennelty koodi. Opinnäytetyö toteutettiin nykyaikaisten verkkoteknologioiden avulla (HTML5, CSS, JavaScript, jQuery, PHP ja SQL). Tulos on uudistunut sopimushallintajärjestelmä, joka on helpompi huoltaa.

Kieli: ruotsi Avainsanat: web-sovellus, PHP, AJAX, jQuery

BACHELOR'S THESIS

Author:	Tim Strengell
Degree Programme:	Information Technology, Vasa
Supervisor:	Susanne Österholm

Title: *Migration of Contract Management System*

Date: 14.4.2016 Number of pages: 34

Abstract

This Bachelor's thesis was made on behalf of Wärtsilä Finland Oy. The thesis consisted of the development of a contract management system, which replaces an older system. The contract management system is used to display and manage contracts within Wärtsilä and additionally contains an interactive report tool. The new system retains the functionality of the old system and contains a few improvements.

The goal of the thesis was to create an updated system with a more consistent user interface and better structured code. The thesis was realized with the help of modern web technologies such as HTML5, CSS, JavaScript with jQuery, AJAX, PHP and SQL. The result became an updated contract management system, which is easier to maintain.

Language: Swedish Key words: web application, PHP, AJAX, jQuery

---

Innehållsförteckning

1 Inledning

Detta examensarbete gjordes på uppdrag av Wärtsilä Finland Oy, Services Contract Management. I detta examensarbete skulle ett webbaserat avtalshanteringssystem utvecklas i PHP för att ersätta ett gammalt avtalshanteringssystem skrivet i ASP.

1.1 Uppdragsgivare

Wärtsilä Finland Oy är ett finländskt företag vilket verkar globalt inom energi- och marinbranscherna. Wärtsilä består idag av tre huvudsakliga affärsområden: Marine Solutions, Energy Solutions och Services. Marine Solutions tillverkar och erbjuder lösningar för motorer inom marinbranschen. Energy Solutions erbjuder lösningar för kraftverk. Services erbjuder bland annat lösningar för underhåll av motorer och support. Underhållsarbeten kan till exempel vara reparation, felsökning eller uppgradering av motorer eller enskilda delar. Services kan även erbjuda förslag på motoroptimering med hjälp av insamlad data från motorer. [1]

1.2 Uppgift

Inom Wärtsilä används internt ett avtalshanteringssystem kallat CMT. CMT är ett webbaserat verktyg vilket används av ett begränsat antal personer världen över för att åskådliggöra och hantera data om avtal Wärtsilä ingått. Förutom att hantera data innehåller systemet även ett rapporteringsverktyg, vilket tillåter användare att skapa anpassningsbara rapporter, vilka normalt visas i tabellform. Vissa rapporter visar även statistik i diagramform.

Uppgiften gick ut på att migrera detta avtalshanteringssystem från ASP till PHP. Det ursprungliga avtalshanteringssystemet var skrivet med hjälp av Microsofts Active Server Pages (ASP) i skriptspråket VBScript. ASP är idag en föråldrad teknik med flera tekniska begränsningar. Eftersom tekniken inte längre utvecklas av Microsoft, är det endast en tidsfråga innan stödet upphör i deras webbserver IIS. Det existerande avtalshanteringssystemet skulle därmed ersättas med ett nytt system skrivet i skriptspråket PHP, vilket är ett mer framtidssäkert val som lättare kan underhållas.

Det fanns flera mål med det nya avtalshanteringssystemet. Eftersom det gamla systemet hade svårhanterlig kod på grund av de begränsningar som kommer med ASP, var det främsta målet att migrera systemet till ett modernare programmeringsspråk, i det här fallet PHP. I avtalshanteringssystemet fanns även stora mängder överflödig kod som inte längre var i användning. Ett annat mål var därför att eliminera denna överflödiga kod och skapa en ny, ren kodbas.

Funktionaliteten hos det nya avtalshanteringssystemet skulle förbli i princip identisk med det ursprungliga. Några förbättringar som uteblivit från det gamla systemet på grund av tidsbrist eller som helt enkelt inte var möjliga att implementera i det tidigare avtalshanteringssystemet skulle implementeras i det nya systemet. Ett av kraven för avtalshanteringssystemet var att det skulle fungera med Internet Explorer 9, vilken vid utvecklingstillfället var den äldsta versionen av Internet Explorer som Wärtsilä stödde.

2 Teori

I detta examensarbete användes moderna webbteknologier som HTML5, CSS, JavaScript med jQuery, PHP och SQL. För gränssnittet användes även komponenter som ingår i jQuery UI samt några insticksmoduler till jQuery. AJAX används för att ladda in innehåll och skicka data till webbservern.

2.1 HTML

HTML är ett märkspråk vilket beskriver uppbyggnaden av en hemsida. HTML står för HyperText Markup Language. Den senaste stabila versionen av HTML är i skrivande stund HTML5. [2]

Ett HTML-dokument består av en samling element och textinnehåll. Elementen är organiserade i en trädstruktur. Element kan innehålla andra element, inom begränsningarna som finns angivna i HTML-standarden. [2] Ett element kan till exempel representera en rubrik, en paragraf, en punktlista eller en bild. Element kan även innehålla attribut som beskriver elementets egenskaper. [2]

Figur 1 visar ett exempel på hur ett HTML-dokument kan se ut. Dokumentet använder sig av lämpliga element för att definiera rubriker och paragrafer. För att skapa menyn används en punktlista med länkar inom listelement. Olika sektioner av dokumentet delas upp med hjälp av div-element. Dessa element ges id-attribut för att kunna väljas ut i skript och stilmallar.

```
<!DOCTYPE html>

<html>
    <head>
        <meta charset="UTF-8" />
        <title>Exempelsida</title>
    </head>
    <body>
        <h1>Exempelsida</h1>
        <div id="container">
            <div id="menu">
                <h2>Meny</h2>
                <ul>
                    <li><a href="index.html">Startsida</a></li>
                    <li><a href="about.html">Om sidan</a></li>
                </ul>
            </div>
            <div id="content">
                <h2>Startsida</h2>
                <p>
                    Detta är startsidan.
                </p>
            </div>
        </div>
    </body>
</html>
```


Figur 1: Exempel på ett HTML-dokument och resultatet i webbläsaren.

Eftersom HTML är ett märkspråk vilket endast beskriver innehållet i ett dokument, används det oftast ihop med andra språk. Genom att använda stilmallar kan utseendet på dokumentet definieras. För att utöka interaktivt beteende på klientsidan används ett skriptspråk. I HTML5-standarden antas JavaScript användas som skriptspråk om inte annat anges [2].

2.2 CSS

CSS är ett språk för stilmallar, vilket tillåter utseendet för HTML- och XML-dokument att anpassas. CSS står för Cascading Style Sheets. Med CSS kan egenskaper definieras hos element valda med hjälp av CSS-selektorer för att bestämma utseendet på de valda elementen. En selektor används för att välja ut element i dokumentet, till exempel enligt typ av element, ett unikt id eller tillhörighet till en klass. [3] CSS3 är den version av CSS som i skrivande stund är under utveckling. CSS3 är inte en enda standard, utan består av flera moduler, vilka standardiseras oberoende av varandra. [4]

Figur 2 visar en stilmall i CSS tillämpad på HTML-dokumentet i figur 1. Texten i sidinnehållet specificeras att texten skall vara 16 pixlar stor och använda sig av den sans-serif font som specificeras i webbläsarens inställningar. Sidmarginalerna tas även bort och de olika div-elementen ges fyllnad (padding) för utseendets skull.

Figur 2: En stilmall i CSS tillämpad på ett HTML-dokument.

Det finns flera sätt att skapa en vänsterjusterad meny, vilken används i detta examensarbete. I detta exempel används en "behållare" med id-attributet "container", vilket visas i HTML-koden i figur 1. I den stilmall som definieras i figur 2 tillämpas fyllnad på behållaren för att förskjuta innehållet åt höger lika långt som menyns bredd. För att flytta menyn till vänster om sidinnehållet används negativa marginaler. Denna metod kan användas för att skapa en layout som innehåller en meny med fast bredd och sidinnehåll vilket fyller ut resten av sidan.

2.3 JavaScript

JavaScript är ett skriptspråk som kan användas för att programmera webbapplikationer. Även om JavaScript främst används på klientsidan, kan JavaScript även användas för att programmera serverapplikationer. [5]

Den JavaScript-kod som visas som exempel i figur 3 definierar en funktion med vilken textstorleken på ett element kan ändras genom att ange dess id som argument. HTML-koden visar ett nytt menyval som används i HTML-dokumentet från figur 1. I HTML-koden används attributet "onclick" för att ange att funktionen ska anropas då användaren trycker på länken. JavaScript tillåter även att händelser kopplas till funktioner i kod.

JavaScript:

```
//Ändra textstorlek på ett element.
function toggleFontSize(elementId) {
    var element = document.getElementById(elementId);
    var fontSize = element.style.fontSize;

    if (fontSize == "1.3em") {
        //Återställ textstorlek till 100% om den är förstorad
        element.style.fontSize = "1em";
    } else {
        //Förstora textstorleken till 130% av den vanliga
        element.style.fontSize = "1.3em";
    }
    //Förhindra att webbläsaren följer länken
    return false;
}
```

HTML:

```
<li>
    <a href="#" onclick="toggleFontSize('container');">
        Ändra textstorlek
    </a>
</li>
```

Figur 3: Exempel på JavaScript-kod.

2.4 JSON

JSON är ett textbaserat format för utbyte av data. JSON står för JavaScript Object Notation och baserar sig på den syntax JavaScript använder för datatyper. Formatet används inte bara av JavaScript, utan kan även användas i andra programmeringsspråk. JSON tillåter data att sparas i form av till exempel objekt och listor på samma vis som i JavaScript. I de allra flesta fall kan JSON-data tolkas direkt av JavaScript. Eftersom JSON-formatet använder sig av mer strikt syntax, är variabeldefinitioner i JavaScript dock inte alltid giltig JSON-data. [6]

Figur 4 visar ett exempel på data sparat i JSON. I exemplet definieras ett objekt vilket innehåller en lista kallad "lådor". Listan innehåller objekt som definierar olika lådor. Strängar i JSON är alltid inneslutna i citationstecken [6].

```
{
    "lådor" : [
        {
            "namn": "liten låda",
            "längd": 0.2,
            "bredd": 0.2,
            "höjd": 0.15
        },
        {
            "namn": "stor låda",
            "längd": 0.5,
            "bredd": 0.5,
            "höjd": 0.5
        }
    ]
}
```

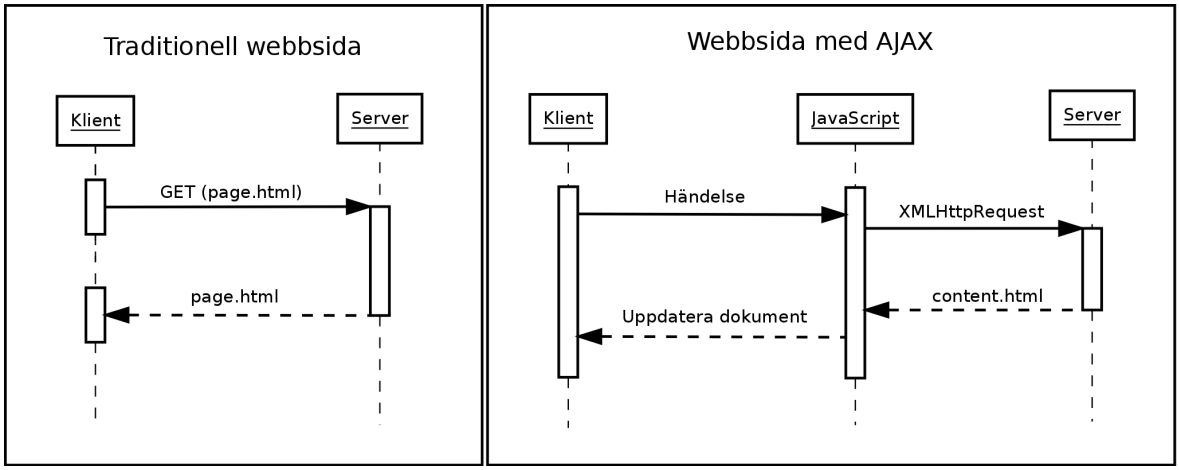
Figur 4: Exempel på data sparat i JSON.

I detta examensarbete används JSON främst för två olika syften. JSON används för att returnera data från webbservern för användning i jQuery UI:s gränssnittskomponent för autokomplettering, vilket beskrivs i kapitel 3.4.6. Det används även för den data som Flot bygger diagram utifrån, vilket beskrivs i kapitel 3.7.5.

2.5 AJAX

AJAX står för Asynchronous JavaScript And XML. AJAX är en samling tekniker vilka tillåter webbsideinnehåll att laddas från en server utan att ladda in en helt ny webbsida. Kärnan i AJAX är objektet XMLHttpRequest, vilket används för att kommunicera med webbservern och behandla hämtad data i JavaScript. [7] Även om vanlig JavaScript-kod kan användas för AJAX, innehåller till exempel JavaScript-biblioteket jQuery funktionalitet för att förenkla användningen av AJAX. [8]

Figur 5 visar sekvensdiagram vilka illustrerar hur webbsidor vilka använder AJAX skiljer sig från traditionellt uppbyggda sidor. I en traditionell, sekventiellt uppbyggd webbapplikation laddas en helt ny webbsida då länkar följs eller formulär skickas. Webbläsaren måste då vänta på svar innan data kan visas. Med AJAX kan data hämtas eller skickas till webbservern med hjälp av JavaScript-kod, och inget sidbyte krävs.



Figur 5: Sekvensdiagram över användning av AJAX.

Det finns många fördelar med AJAX. En fördel är att mindre data behöver hämtas eftersom sidan kan laddas i delar. Ytterligare en fördel är att användaren hålls kvar på den aktuella sidan. Användaren kan då fortfarande använda sig av resten av webbsidan medan ny data hämtas, vilket förbättrar användarupplevelsen.

Ett exempel på hur AJAX kan användas för att förbättra användarupplevelsen är ett bildspel, vilket är inbäddat i en webbsida. Då användaren begär nästa bild i bildspelet, kan ett skript skrivet i JavaScript använda sig av XMLHttpRequest för att begära nästa bild. Webbservern svarar med ett bildelement, vilket skriptet sätter in i HTML-dokumentet. Tack vare detta behöver endast bildspelet uppdateras och hela webbsidan behöver inte laddas om för att byta bild.

2.6 jQuery

jQuery är ett bibliotek skrivet i JavaScript ämnat för att underlätta utveckling av webbapplikationer. Bland annat erbjuder jQuery möjligheten att välja ut element i HTML-dokument med hjälp av de elementväljare som används i CSS, samt metoder för att lätt binda funktioner till olika händelser. jQuery inkluderar även animerade effekter, vilka kan användas för att förbättra användarupplevelsen, samt metoder för att förenkla användningen av AJAX. Funktionaliteten hos jQuery kan även utökas med hjälp av insticksmoduler. [8]

Figur 6 visar hur jQuery kan användas för att skapa knappar med vilka användaren kan visa och dölja ett element innehållande inställningar. Ett jQuery-objekt skapas där de knappar vilka ska kunna visa och dölja inställningarna väljs ut. Då användaren trycker på dessa knappar anropas en funktion vilken visar eller döljer ett element med hjälp av en animation. Genom att placera koden i en funktion kopplad till HTML-dokumentets "ready"-händelse körs inte koden innan dokumentet har laddat klart.

```
//Koden inuti körs först då HTML-dokumentet har laddat klart
$(document).ready(function() {
    //Välj ut alla knappar med klassen "optionsToggler"
    $buttons = $("button.optionsToggler");

    /* Koppla en funktion till klickhändelsen för
     * knapparna vilken visar/döljer elementet med
     * id "options" med en 500 ms lång animation
     */
    $buttons.click(function(event) {
        $("#options").slideToggle(500);
    });
}
```

Figur 6: Exempel på jQuery-kod.

2.6.1 jQuery och AJAX

jQuery inkluderar funktionalitet för att förenkla användningen av AJAX. Bland funktionaliteten finns en metod kallad "$.ajax", samt ytterligare metoder vilka förenklar användningen i mer specifika fall. [8]

De två vanligaste sätten att begära data från en webbserver är "GET" och "POST". En GET-begäran används vanligtvis då data hämtas från servern, medan POST-begäran används då data skall ändras på servern. Även om metoden "$.ajax" kan användas för att skicka dessa begäran, erbjuder jQuery även hjälpmetoderna "$.get" och "$.post" för att ytterligare förenkla användningen av AJAX. Ifall data endast behöver hämtas utan att skicka några variabler kan metoden "load" användas för att ladda innehåll direkt till ett HTML-element [8] Hur dessa metoder kan användas i JavaScript-kod demonstreras i figur 7.

.load()

```
/* Ladda innehållet i about.php till
 * elementet med id-attributet "content"
 */
$("#content").load("about.php");
```

$.ajax()

```
/* Använd $.ajax för att ladda in about.php
 */
$.ajax({
    url:    "about.php",
    method: "GET"
}).done(function(data) {
    $("#content").html(data);
});
```

$.get()

```
/* Skicka en GET-begäran till car.php
 * med variabeln "id"
 */
$.get("car.php", {
    data: {
        id: 5
    }
}).done(function(data) {
    $("#content").html(data);
})
```

$.post()

```
/* Skicka en POST-begäran till car.php
 * med variablerna "id" och "color"
 */
$.post("car.php", {
    data: {
        id:     3,
        color:  "Red"
    }
})
```

Figur 7: Olika sätt att använda AJAX med jQuery.

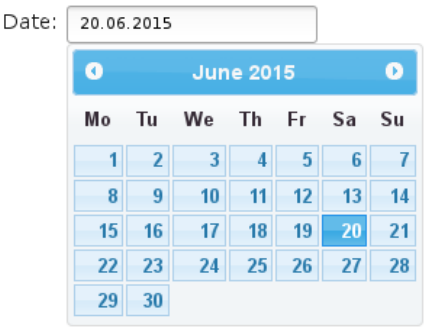
2.6.2 jQuery UI

jQuery UI är en samling med tillägg till jQuery, vilka underlättar byggandet av gränssnitt i webbapplikationer. De gränssnittskomponenter som jQuery UI innehåller har ett enhetligt utseende som kan anpassas med hjälp av teman. Teman kan till exempel skapas med hjälp av webbapplikationen ThemeRoller, vilket är en temaskapare på den officiella jQuery UI webbsidan. [9]

De komponenter som jQuery UI i stort sett kan delas in i är: interaktivitet, gränssnittskomponenter och effekter. Med hjälp av den funktionalitet för interaktivitet som finns i jQuery UI kan webbutvecklare lägga till dynamiskt beteende till HTML-element. Till exempel kan ett HTML-element vilket användare själv kan ändra storlek på skapas. Gränssnittskomponenter kan användas för att enkelt skapa till exempel knappar, flikar, datumväljare och många andra komponenter. Effekter utökar den redan existerande funktionaliteten hos jQuery och låter utvecklare anpassa utseendet på element och animationer i ännu större grad än basfunktionaliteten i jQuery tillåter. [9]

Som exempel på en ofta använd gränssnittskomponent kan nämnas en datumväljare, vilken är användbar i formulär med datumfält. Ett exempel på en datumväljare visas i figur 8. I exemplet används jQuery UI med ett anpassat tema och datumväljaren med anpassade inställningar för att visa finskt datumformat.

```
/* Använd datumväljaren på alla input-element
 * med klassen "datepicker"
 */

$("input.datepicker").datepicker({
    //Sätt datum till formatet "01.01.2015"
    dateFormat: "dd.mm.yy",
    //Visa måndag som första dag i veckan
    firstDay: 1
});
```



Figur 8: jQuery UI:s datumväljare med anpassat tidsformat.

2.6.3 jQuery Validation Engine

jQuery Validation Engine är en insticksmodul till jQuery, vilken tillåter utvecklare att enkelt implementera validering på formulär i HTML-dokumentet. Valideringsreglerna definieras med hjälp av klasser för input-elementen i formuläret. Med hjälp av de data-attribut som finns i HTML5 kan även felmeddelanden anpassas. [10]

Figur 9 visar ett i stort sett vanligt formulär uppbyggt med HTML. Till skillnad från vanliga formulär definieras här valideringsregler i klassattributet. Det första fältet använder valideringsregeln "required", vilken anger att fältet måste vara ifyllt. Det andra använder en "custom" regel. Trots namnet ingår flera sådana regler färdigt i jQuery Validation Engine. I det här fallet tillämpas regeln "integer" för att se till att endast heltal kan matas in.

```
<form id="exampleForm">
    <label>
        Required: <input type="text" name="reqfield" class="validate[required]]"></input>
    </label>
    <label>
        An integer: <input type="text" name="intfield" class="validate[custom[integer]]"></input>
    </label>
    <input type="submit" value="Submit"></input>
</form>
```

Figur 9: Exempel på ett HTML-formulär vilket definierar valideringsregler.

För att tillämpa jQuery Validation Engine på ett formulär väljs formuläret ut i ett jQuery-objekt och metoden "validationEngine" anropas. Validation Engine kommer då att förhindra att formuläret skickas innan valideringsreglerna är uppfyllda, och ge användaren feedback. I figur 10 anges även en valfri inställning för att berätta åt insticksmodulen att placera valideringsnotifieringarna till höger om fältet.

```
$("#exampleForm").validationEngine({
    promptPosition: "centerRight"
});
```

Figur 10: jQuery-kod där jQuery Validation Engine tillämpas på ett formulär.

I figur 11 kan resultatet av koden i figur 9 och 10 ses. Användaren får feedback på de felaktiga fälten, och formuläret kan inte skickas innan kraven uppfylls. Ifall formulär skickas med hjälp av JavaScript-kod istället för den vanliga händelsen för att skicka in, kan metoden "validate" användas för att manuellt validera formuläret. Detta är användbart ifall formulär skickas in med hjälp av AJAX, vilket görs i detta examensarbete.

Figur 11: Exempel på användarfeedback från jQuery Validation Engine.

2.6.4 Flot

Flot är en insticksmodul till jQuery vilken tillåter utvecklare att lätt skapa diagram utifrån JSON-formaterad data. Grafer skapade med Flot kan innehålla flera dataserier, vilka kan anpassas enligt behov. Flot stöder linje-, punkt- och stapeldiagram med standardfunktionaliteten. Varje serie kan använda en eller flera olika typer av diagram. [11] Det är därmed lätt att skapa till exempel ett diagram vilken innehåller både punkter och linjer. Användningen av Flot beskrivs i mer detalj i kapitel 3.7.5.

2.7 PHP

PHP är ett skriptspråk för webbservrar. PHP kallades ursprungligen för Personal Home Page Tools, men står idag för PHP Hypertext Preprocessor. Den ursprungliga versionen av PHP utvecklades av Rasmus Lerdorf som i Juni 1995 släppte PHP till allmänheten. I senare versioner av PHP implementerades objektorientering. [12]

Figur 12 visar ett enkelt exempel på objektorienterad PHP-kod. PHP-kod placeras inom start- och sluttaggar och kan blandas med HTML-kod vid behov. I exemplet skapas en vyklass, vilken kan användas för att rita upp ett meddelande. Klassen definierar en konstruktor, det vill säga den metod som körs då en instans av klassen skapas, vilken kallas för __construct i moderna versioner av PHP. Konstruktorn tar emot ett meddelande, vilken sparas i medlemsvariabeln message. För att skriva ut meddelandet i HTML-dokumentet definierar klassen metoden view. Resultatet av PHP-koden visas till höger i figur 12.

```php
<?php
class SampleView {
    private $message;

    public function __construct($message) {
        $this->message = $message;
    }

    public function view() {
        echo $this->message;
    }
}

$sampleView = new SampleView("Sample Text");
$sampleView->view();
?>
```

Sample Text

Figur 12: Exempel på PHP-kod.

PHP inkluderar gränssnittet PDO för att kommunicera med databaser. PDO står för PHP Data Objects och tillåter utvecklare att kommunicera med olika databashanterare på ett enhetligt sätt. [12] Detta tillåter utvecklare att kommunicera med olika databashanterare på samma vis och kunde vara till fördel ifall databashanteraren någon gång skulle bytas ut.

2.8 SQL

SQL står för Structured Query Language och är ett språk med vilket man kan utföra satser mot relationsdatabaser. Grunden för relationsdatabaser definierades ursprungligen år 1970 i Codd E.F:s publikation "A Relational Model of Data for Large Shared Data Banks" SQL utvecklades av IBM och kallades ursprungligen för SEQUEL, eller Structured English Query Language. SQL används oftast för att hantera data uppdelad i tabeller. [13]

Det finns flera varianter på SQL. Den variant som används i detta examensarbete är Oracles variant av SQL. De SQL-kommandon som finns i Oracle kan huvudsakligen delas in i två grupper: DDL (Data Definition Language) och DML (Data Manipulation Language). DDL innehåller bland annat kommandon för att skapa, ändra och ta bort de tabeller i vilka data sparas. DML används vanligtvis för att hämta, sätta in och uppdatera data i databasen. Det finns även övriga typer av kommandon för till exempel hantering av transaktioner och sessioner. [13]

I detta examensarbete existerade databasen från förut. De SQL-kommandon som användes hörde oftast till DML. Figur 13 visar användningen av DDL-kommandot "CREATE", vilket i det här fallet används för att skapa en tabell. Kommandot "SELECT" används för att hämta data ur databasen. Kommandot "INSERT" lägger till en ny rad i tabellen, medan "UPDATE" används för att ändra existerande data.

```sql
-- Skapa en tabell innehållande bilar
CREATE TABLE CARS (
    CAR_ID              NUMBER(4),
    CAR_MANUFACTURER    VARCHAR(40),
    CAR_COLOR           VARCHAR(20),

    CONSTRAINT PK_CARS PRIMARY KEY (CAR_ID)
);

-- Välj ut alla bilar som är röda
SELECT CAR_ID, CAR_MANUFACTURER FROM CARS WHERE CAR_COLOR = 'Red';

-- Sätt in en ny bil i tabellen
INSERT INTO CARS (CAR_ID, CAR_MANUFACTURER, CAR_COLOR)
    VALUES (5, 'The manufacturer', 'Blue');

-- Ändra alla röda bilar till gröna
UPDATE CARS SET CAR_COLOR = 'Green' WHERE CAR_COLOR = 'Red';
```

Figur 13: Exempel på SQL kod.

2.8.1 Prepared Statements

Prepared statements är ett sätt att skilja på tolkningen av SQL-satsen och de variabler som används i satsen. Med prepared statements skrivs själva SQL-satsen med så kallade "platshållarvariabler" istället för själva variablerna. Ett exempel på hur dessa skrivs i Oracle illustreras i figur 14, där platshållarvariabler börjar med ett kolon. Prepared statements används i detta examensarbete där variabler i en SQL-sats blivit skickade från användare eller inmatade via kod.

```sql
SELECT COLUMN1, COLUMN2 FROM SCHEMA1.TABLE1 WHERE COLUMN3 LIKE :var;
```

Figur 14: Exempel på hur en SQL-sats skrivs med prepared statements.

Användning av prepared statements har många fördelar. Eftersom SQL-satsen tolkas separat från variablerna behöver inte tecken som är speciella för SQL hanteras på något speciellt vis. Ifall variabler i en SQL-sats inte behandlas på förhand då prepared statements inte används, kan satsen bli feltolkad av databasen och systemet inte bete sig som tänkt. Användare med ont uppsåt kunde även utnyttja detta till att bilda helt nya SQL-satser, vilket kallas för SQL-injektion. Användning av prepared statements förhindrar detta. Eftersom SQL-satsen förblir den samma med endast variablerna ändrade, kan databashanterare använda cacheminne för att förbättra prestanda vid exekvering av SQL-satser. Förberedda satser är därför mer robusta, säkrare och i vissa fall snabbare än traditionellt uppbyggda SQL-satser.

3 Utförande

Arbetet började med analys av strukturen och koden hos det gamla avtalshanteringssystemet. För att visa vilka filer som faktiskt användes i det gamla avtalshanteringssystemet och deras beroenden inom systemet skapades grafiska kartor.

3.1 Tillvägagångssätt

Innan arbetet med det nya avtalshanteringssystemet påbörjades, kartlades det gamla systemet. Det nya systemet implementerades stegvist genom att dela upp funktionaliteten i delar och implementera dem en åt gången. Eftersom ett äldre system redan existerade och ett av kraven på det nya systemet var att det nya systemet skulle bibehålla funktionaliteten hos det gamla, var funktionaliteten på förhand i stort sätt redan bestämd. Regelbundna möten hölls för att gå igenom de delar av systemet vilka skulle implementeras.

Efter att en viss del implementerats, granskades delen av en handledare på Wärtsilä. Avtalshanteringssystemet uppdaterades enligt den feedback som gavs av handledaren, varefter delen granskades på nytt. Feedbacken kunde variera från felrapporter till kommentarer på utseendet hos avtalshanteringssystemet.

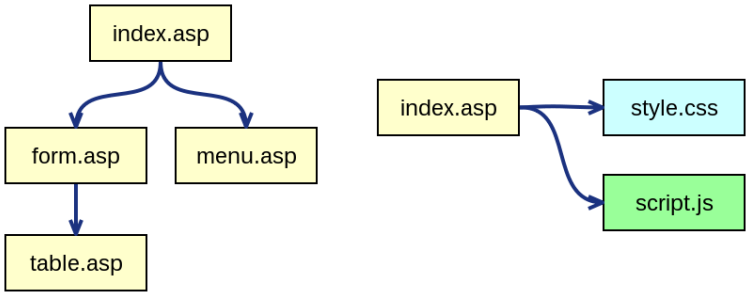
Då den funktionalitet som fanns i det gamla avtalshanteringssystemet implementerats i det nya, började förbättringsförslag på avtalshanteringssystemet från handledaren implementeras. Dessa förslag inkluderade både förslag som hade funnits från tidigare till det gamla avtalshanteringssystemet, samt nya förslag. Till förbättringsförslagen hörde bland annat en temaväljare och möjlighet för användare att anpassa sökfunktionen mera i detalj.

3.2 Kartläggning av det gamla systemet

Arbetet påbörjades genom att kartlägga det gamla avtalshanteringssystemet. Detta gjordes dels för att kunna bestämma exakt vad som ingick i det gamla avtalshanteringssystemet och dels för att kunna använda kartorna för att snabbt kunna hitta filer att använda som referens till den nya implementationen.

Det gamla avtalshanteringssystemet hade en inkonsistent kodbas. Försök att uppgradera och utöka funktionaliteten hos systemet hade redan genomförts många gånger. Det system som var i användning var en sammanslagning av dessa försök och bestod därför av många generationer av kod. Olika delar av koden kunde därför bete sig på många olika sätt. Eftersom gammal kod inte hade borttagits fanns även stora mängder överflödig kod i kodbasen, vilken inte längre var i användning.

För att få en överblick över de filer som faktiskt användes i systemet, påbörjades arbetet med att kartlägga det tidigare avtalshanteringssystemet. Kartläggningen gjordes med hjälp av Pencil. Pencil är ett gratis verktyg med öppen källkod vilket tillåter användare att skapa prototyper av grafiska gränssnitt. [14] Det finns även möjlighet att rita upp enklare figurer och koppla figurer samman med pilar, vilket var den funktionalitet som användes i kartläggningen. I detta fall användes Pencil för att grafiskt visa kopplingar mellan de filer som användes i avtalshanteringssystemet.



Figur 15: Kartor skapade med Pencil.

En stor karta skapades, vilken visade de filer som använde ASP i det existerande avtalshanteringssystemet och deras beroenden. Ett exempel på denna typ av karta finns på vänstra sidan av figur 15. Den egentliga kartan blev mycket mer komplicerad. Speciellt det gamla rapporteringsverktyget använde sig av en stor mängd filer. För sidor med ytterligare kopplingar som skriptfiler och stilmallar skapades mindre kartor som visade dessa, vilket illustreras på högra sidan i figur 15.

För att skapa dessa kartor ritades rektanglar i Pencil med hjälp av de vanliga figurer som Pencil erbjuder. Varje rektangel motsvarar en fil. Rektanglarnas bakgrund färgades så att filtypen hos den motsvarande filen lätt kunde åskådliggöras och filnamnen skrevs in i rektanglarna. Beroenden mellan filer visades genom att dra pilar mellan de utplacerade rektanglarna. Pilarna fastnar automatiskt vid de utplacerade rektanglarna och följer dessa om de flyttas.

Resultatet av kartläggningen blev flera kartor som visade de filer som ingick i det existerande avtalshanteringssystemet och deras kopplingar. Tack vare dessa kunde de filer som faktiskt var i användning lätt hittas, och funktionalitet som användes kunde lätt spåras tack vare att beroendena klart framgick.

3.3 Val av teknik

Det främsta målet med migrationen av avtalshanteringssystemet var att skapa en ny kodbas i PHP. Valet av PHP som serverskriptspråk var ett val vilket var fastställt från början på begäran av uppdragsgivaren. Även om PHP är det vanligaste alternativet

som serverskriptspråk hos webbservrar, finns även många andra språk vilka kunnat användas som alternativ, t.ex. Python, C# och Java [15]. En stor fördel med PHP är dock att det är vanligt, och att stöd för PHP därför finns hos många olika webbservrar.

På grund av att användargränssnittet till största delen skulle bete sig som det gamla användargränssnittet var mycket av den teknik som användes från förut ett självklart val. jQuery och jQuery UI användes redan i de nyare delarna av det gamla avtalshanteringssystemet. Eftersom dessa webbteknologier fortfarande är moderna användes de även i det nya avtalshanteringssystemet, och tillämpas även på delar vilka inte ursprungligen använde sig av dessa teknologier.

Den ursprungliga tekniken för att rita upp diagram i rapporteringsverktyget fungerade inte alltid som tänkt, vilket delvis berodde på brist på data i utvecklingsdatabasen. Eftersom jQuery-insticksmodulen Flot verkade uppfylla de behov som fanns för diagrammen och innehålla bra dokumentation, beslöts det att den skulle användas.

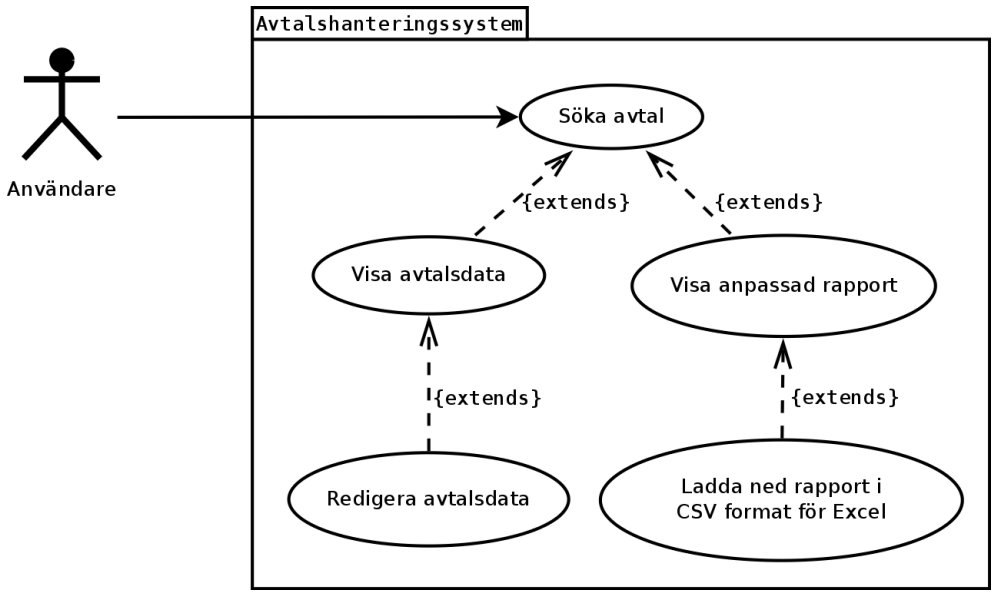
3.4 Användargränssnitt

Användargränssnittet utvecklades huvudsakligen med hjälp av HTML, CSS, jQuery och jQuery UI. Även många olika insticksmoduler till jQuery användes. Avtalshanteringssystemet fungerar som en webbapplikation, vilken aldrig byter webbsida utan istället laddar in innehåll och skickar formulär med hjälp av AJAX. AJAX används dels genom manuella AJAX-anrop från webbapplikationens kod, och dels automatiskt av komponenter vilka finns i jQuery UI.

3.4.1 Designprincip

Avtalshanteringssystemet använder engelska som språk i både gränssnitt och kod, eftersom systemet används världen över. Gränssnittet följer en traditionell design för webbsidor för vanliga persondatorer. Responsiv webbdesign är ett relativt nytt begrepp, i vilket en webbsida designas så att den automatiskt skalar beroende på storleken på enheten. Eftersom systemet inte är planerat att användas på mobila enheter i detta skede, beslöts det dock att inte tillämpa denna designprincip.

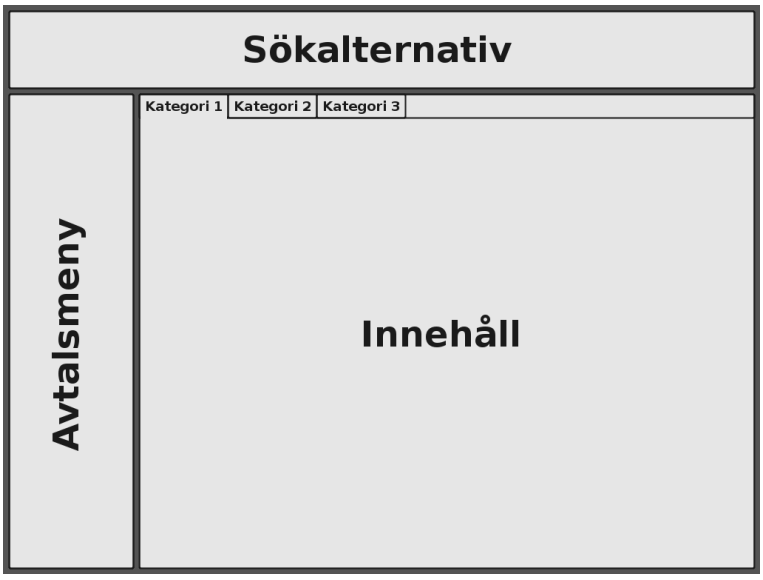
Avtalshanteringssystemet kan delas upp i ett antal användningsområden. Dessa illustreras i ett use case-diagram i figur 16. En användare kan söka på avtal i systemet. Från sökresultaten kan användaren även välja att visa data om avtal, och vid behov ändra viss data. Alternativt kan användaren välja att skapa rapporter för en kategori ur sökresultaten, och vid behov ladda ner en rapport i CSV-format, vilken kan öppnas i Excel.



Figur 16: Use case-diagram som visar hur systemet kan användas.

3.4.2 Gränssnittets struktur

Gränssnittet kan delas upp i tre huvudsakliga delar: Sökalternativ, avtalsmeny och innehåll. Dessa illustreras i figur 17. Längst uppe på sidan finns sökalternativen. Dessa består av en sökruta, ett antal flervalsknappar och några andra knappar. Under sökalternativen finns till vänster en avtalsmeny. Avtalsmenyn är i princip en trädvy vars innehåll beror på de sökalternativ användaren har valt. Avtalsmenyn uppdateras automatiskt beroende på de sökalternativ användaren väljer. Resten av sidan består av själva innehållet, vilket beror på de val användaren gör i avtalsmenyn. Innehållet består normalt av en uppsättning flikar vilka delar upp innehållet i kategorier.



Figur 17: Skiss över gränssnittets uppbyggnad.

3.4.3 Avtalsmeny

Gränssnittet innehåller en meny med avtal, vilken alltid är synlig på vänstra sidan i gränssnittet. Avtalen presenteras i form av en trädvy med kategorier. De kategorier som innehållet grupperas enligt och innehållet i menyn beror på de sökalternativ användaren valt.

I HTML-kod skapas trädvyn som en samling nästade oordnade listor, även kallade punktlistor. En oordnad lista får endast innehålla listelement (HTML-elementet <li>). Ett listelement kan däremot innehålla nästan vad som helst. För kategorier består listelementet av ett span-element, vilket fungerar som kategorirubrik, följt av en oordnad lista. För de egentliga alternativen innehåller listelementen länkar. Figur 18 visar ett exempel på denna struktur. I den egentliga koden skapas HTML-delen av trädvyn utifrån data i databasen med hjälp av en menyklass i PHP.

```
<div id="mainMenu">
    <ul>
        <li>
            <span>Category 1</span>
            <ul>
                <li>
                    <span>Subcategory 1</span>
                    <ul>
                        <li><a href="page1.php">Option 1</a></li>
                        <li><a href="page2.php">Option 2</a></li>
                    </ul>
                </li>
            </ul>
        </li>
    </ul>
</div>
```

Figur 18: Exempel på hur en enkel trädvy kan definieras i HTML-kod.

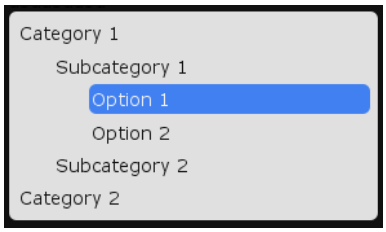
Avtalsmenyn i det ursprungliga avtalshanteringssystemet använde sig av en insticksmodul till jQuery för att skapa trädvyn. Eftersom det fanns vissa förbättringsförslag för att förbättra användbarheten och för att en grundläggande trädvy inte är alltför svår att skapa med hjälp av jQuery skapades en helt ny trädvy.

Figur 19 visar exempel på kod vilken kan användas för att skapa en interaktiv trädvy med jQuery. Då användare klickar på kategorirubrik används en animation för att visa eller dölja listelementet som följs av kategorirubriken. Ifall ett av alternativen väljs, förhindras den vanliga funktionaliteten hos länken, så att inte den sida som länken hänvisas till laddas in i webbläsaren. Istället laddas den senare in med hjälp av jQuerys AJAX-funktionalitet i ett element på sidan. Den valda raden tilldelas även klassen “active”, för att kunna färgläggas med hjälp av en stilmall.

```
$("#mainMenu").on("click", "span", function(event) {
    //Hantera kategorirubriker

    //Visa/dölj det följande elementet med en animation.
    $(this).next().slideToggle(200);
}).on("click", "a", function(event) {
    //Hantera menyalternativ

    //Förhindra att länken följs på vanligt vis
    event.preventDefault();

    //Se till att inga tidigare valda menyalternativ är markerade
    $("#mainMenu .active").removeClass("active");

    //Lägg till klassen "active" för anpassat utseende
    $(this).addClass("active");

    //Ladda det länken pekar på inuti elementet med sidinnehållet
    $("#content").load($(this).attr('href'));
});
```

Figur 19: jQuery-kod för att skapa en enkel trädvy.

Utseendet på aktiva rader kan enkelt definieras i CSS-kod, eftersom de tilldelas en klass. I detta avtalshanteringssystem ges aktiva rader en annan bakgrundsfärg och textfärg, så det valda alternativet tydligt framgår. Figur 20 visar hur en meny skapad på detta sätt kan se ut med en lämplig stilmall.



Figur 20: En trädvy skapad med hjälp av jQuery med CSS för stilformatering.

3.4.4 Sökalternativ

Sökning på avtal sker genom en textbox. Flera knappar i olika form finns även för att anpassa sökningen. De knappar som används för sökalternativen är implementerade med hjälp av jQuery UI. En fördel med att implementera knappar med hjälp av jQuery UI är att utseendet på knapparna, liksom andra jQuery UI komponenter, lätt kan anpassas genom att byta det tema som jQuery UI använder. Med hjälp av metoden buttonset, vilken ingår i jQuery UI går är det även lätt att skapa flervalsknappar.

I figur 21 visas den HTML-uppställning som krävs för att metoden buttonset, vilken ingår i jQuery UI, skall kunna skapa flervalsknappar.

```
<div id="categories">
    <input type="radio" name="category" id="cat1" value="cat1"></input>
    <label for="cat1">Category 1</label>
    <input type="radio" name="category" id="cat2" value="cat2"></input>
    <label for="cat2">Category 2</label>
    <input type="radio" name="category" id="cat3" value="cat3"></input>
    <label for="cat3">Category 3</label>
</div>
```

Figur 21: HTML-uppställning för att skapa flervalsknappar.

För att skapa flervalsknapparna väljs menyn ut i ett jQuery-objekt och metoden buttonset anropas. Resultatet blir ett antal knappar vilka är grupperade samman, vilket visas i figur 22.

Figur 22: Resultatet av metoden buttonset i jQuery UI.

En av de nya funktionerna i avtalshanteringssystemet var möjligheten för användare att anpassa sökalternativen i mer detalj än tidigare. Genom att trycka på en knapp kan användare fälla ut utökade alternativ. Bland dessa alternativ kan användaren med hjälp av kryssrutor välja de kolumner vilka tas i beaktande under sökningen. Som standard söker systemet i alla möjliga kolumner som i det tidigare systemet. Dessa utökade alternativ inkluderar även ett flervalsalternativ med vilket användaren kan välja tema. Teman beskrivs i mer detalj i kapitel 3.4.8.

Avtalsmenyn uppdateras automatiskt baserat på de sökalternativ användaren har valt. Då användaren väljer något av knappalternativen eller skriver något i sökrutan skickas en begäran om en ny meny med hjälp av jQuerys AJAX-funktionalitet. En begäran med de valda sökalternativen skickas till en PHP-sida vilken returnerar en lämplig meny.

3.4.5 Flikar

Det är opraktiskt att visa all data som kan hanteras för ett avtal på en enda sida. Dessutom är vissa typer av data otillgänglig för vissa avtal av varierande orsaker. För att göra data mer lätthanterlig delas den data som kan hanteras upp i olika kategorier. Dessa kategorier visas i form av flikar. Flikarna är implementerade med hjälp av en flikkomponent (Tabs) vilken ingår i jQuery UI.

För att skapa flikar med hjälp av jQuery UI behövs ett HTML-element vilken innehåller en lista. Listan innehåller listelement, vilka i sin tur innehåller länkar till det innehåll man vill visa i fliken. Figur 23 visar ett exempel på denna struktur.

```
<div id="tabs">
    <ul>
        <li><a href="category1.php">Category 1</a>
        <li><a href="category2.php">Category 2</a>
        <li><a href="category3.php">Category 3</a>
    </ul>
</div>
```

Figur 23: HTML-kod vilken används för att skapa flikar med jQuery UI.

I det här fallet pekar länkarna till externa webbsidor, men de kan även peka på element i HTML-dokumentet, vilka i det fallet placeras inuti flikelementet efter listan. För att använda flikkomponenten med jQuery UI i JavaScript-koden anropas metoden tabs på ett jQuery-objekt med det element där flikarna skall skapas, vilket visas i figur 24.

```
$("#tabs").tabs({
    beforeLoad: function(event, ui) {
        //Don't display old data
        ui.panel.empty();
    }
});
```

Figur 24: Skapande av flikar med jQuery UI i JavaScript-kod.

Koden i figur 24 anger även en funktion vilken körs före en flik laddas in. Denna funktion är inte nödvändig för att skapa flikar, men används för att anpassa beteendet då flikar laddade via AJAX används. Eftersom flikkomponenten använder sig av AJAX, laddas data in då en flik aktiveras. Ifall en flik som blivit vald tidigare laddas in på nytt, visas gammal data i fliken innan ny data har hämtats. För att förhindra att gammal data visas, töms det element där data ska laddas före ny data sätts in.

Resultatet av koden blir en uppsättning flikar vilka lätt låter användaren bläddra mellan kategorier, samt vars utseende lätt kan anpassas genom att byta ut teman i jQuery UI. Resultatet av koden i figur 23 och 24 visas i figur 25.

Figur 25: Flikar skapade med jQuery UI.

3.4.6 Autokomplettering av fält

För många av flervalsfälten i formulären kunde de möjliga alternativen lätt skapas med hjälp av PHP-kod direkt i HTML-koden. Denna metod fungerar dock endast väl så länge antalet alternativ är få. Problemet med alltför många alternativ är att det finns risk för problem med prestandan ifall klienten behöver hämta alla alternativ,

samt att användarvänligheten försämras. För fält där det finns många möjliga alternativ behövdes ett sätt för att låta användaren söka bland de möjliga. Lösningen på problemet blev jQuery UI:s gränssnittskomponent för autokomplettering.

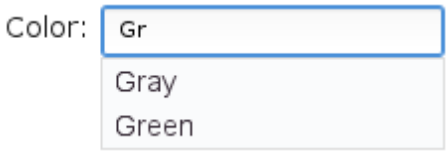
I jQuery UI ingår en gränssnittskomponent för autokomplettering, vilken tillåter användare att lätt söka bland en stor mängd alternativ. Sökalternativen kan lätt bläddras igenom med både tangentbord och mus. I denna gränssnittskomponent ingår även AJAX-funktionalitet, med vilken en begäran kan köras mot en extern sida, vilken returnerar resultat i form av JSON-data.

I figur 26 visas hur autokomplettering med hjälp av AJAX används med jQuery UI. För att tillämpa den gränssnittskomponent som används för autokomplettering i jQuery UI används metoden autocomplete. För att använda sig av AJAX behöver en sida anges som källa (source) i inställningarna hos gränssnittskomponenten.

```
$("#color").autocomplete({
    source: "results.php"
});
```

Figur 26: Kod för autokomplettering med jQuery UI.

Figur 27 visar en textbox vilken använder sig av jQuery UI:s gränssnittskomponent för autokomplettering.



Figur 27: Autokomplettering med jQuery UI.

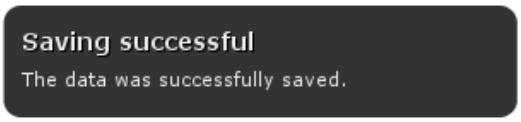
Då användaren skriver i textfältet, skickas söktermen till servern med hjälp av en GET-begäran. Söktermen placeras i variabeln "term", vilken avläses på servern. På servern körs en förfrågan mot databasen med den skickade söktermen. Efter att en lista med sökresultat skapats skrivs den ut av PHP med funktionen json_encode, vilken skapar en sträng med data i JSON-format från en variabel i PHP. Denna sträng returneras och tolkas av gränssnittskomponenten.

Även om fält med autokomplettering i avtalshanteringssystemet normalt valideras med hjälp av PHP på serversidan, kan användarupplevelsen ytterligare förbättras genom att ge feedback på klientsidan. Till exempel kan en ikon sättas bredvid textfältet, vilken indikerar vilken status textfältet har, till exempel: "OK", "felaktig data" och "hämtar resultat". Ikonen kan sedan till exempel läsas av en anpassad valideringsfunktion i jQuery Validation Engine. På detta sätt får användaren snabbare feedback på inmatad data.

3.4.7 Notifieringar

För att ge användaren feedback på försök att spara data används notifieringar. Feedback kan till exempel vara att data har sparats i databasen eller att ett fel uppstod vid försök att spara. För administratörer av systemet kan notifieringarna även visa ytterligare felinformation, något som bestäms beroende på de rättigheter en användare har i systemet.

Gritter är en insticksmodul till jQuery vilken kan användas för att enkelt kunna skapa notifieringar. [16] Notifieringarna dyker normalt upp i övre högra hörnet av webbläsarfönstret, där de syns under en kort tid för att sedan försvinna. Ett exempel på en sån notifiering finns i figur 28. Ifall flera notifieringar visas samtidigt, visas de i en rad och försvinner en och en.



Figur 28: En notifiering skapad med Gritter.

Notifieringar skapas enkelt med ett metodanrop i vilket de önskade inställningarna anges. Notifieringarna ges vanligtvis en titel och ett meddelande, men kan även innehålla bilder. För att skapa en enkel notifiering kan man till exempel använda koden i figur 29.

```
$.gritter.add({
    title: "Saving successful",
    text: "The data was successfully saved.",
    time: 5000, //5000 ms
    sticky: false
});
```

Figur 29: Skapande av en Gritter-notifikation i kod.

Koden i figur 29 definierar en notifikation med en titel och ett meddelande, vilken visas i fem sekunder på skärmen innan den försvinner. Notifieringar stannar normalt endast en kort tid på skärmen innan de försvinner, men kan även stängas genom att användaren för musen över dem och trycker på ett kryss. Tiden det tar för notifieringarna att försvinna kan dock anpassas enligt behov. Ifall man vill att notifieringen inte skall försvinna utan att användaren själv stänger den kan en så kallad “sticky” notifiering skapas. Dessa kan vara användbara för att till exempel visa information för felsökningssyften.

3.4.8 Teman

En av de nya funktionerna som skapades för det nya avtalshanteringssystemet var möjligheten för användare att välja mellan ett antal teman. Teman implementerades med hjälp av en klass i PHP. Beroende på vilket tema användaren valt väljer klassen lämpliga stilmallar, vilka sedan skrivs ut på HTML-sidan med PHP då webbapplikationen laddas. Om inget tema är valt eller det valda temat inte hittas används ett standardtema.

För att gränssnittet skulle se konsistent ut med de komponenter jQuery UI erbjuder skapades anpassade versioner av jQuery UI. Dessa versioner skapades med hjälp av ThemeRoller, vilken är en temaskapare jQuery UI erbjuder på sin officiella webbsida. ThemeRoller tillåter designers att bland annat välja färger, fonter och stil på de utsmyckningar vilka används i komponenterna. Exempel på olika teman för jQuery UI illustreras i figur 30. En lämplig version av jQuery UI laddas in beroende på det tema användaren har valt.

Figur 30: Exempel på teman för jQuery UI skapade med hjälp av ThemeRoller.

Ett problem som fanns då teman implementerades var utseendet på ikoner. De ikoner som användes hade ursprungligen skapats som bildelement i HTML-koden. För att kunna byta ut källan på bilderna beroende på tema skulle många kodändringar och tillägg krävts. Som lösning på detta byttes bilderna ut mot blockelement med klasser för ikoner. Tack vare detta kan de bilder ikonerna använder lätt definieras som en bakgrundsbild i de stilmallar som används. Detta är en smidig lösning som även möjliggör andra utseendemässiga förbättringar, till exempel att byta ut bilden då musen förs över ikonen.

3.5 Databas

Den databas som användes under utvecklingen var en databas speciellt ämnad för utveckling vilken var åtskild från den databas som användes i produktion. Det databashanterare som användes var Oracle. PHP erbjuder flera olika sätt för att ansluta till en Oracledatabas.

De två främsta alternativen som togs i beaktande som möjliga anslutningssätt var PDO med drivrutinen PDO_OCI samt extensionen Oracle OCI8. [12] Då examensarbetet utfördes var dock den PDO_OCI drivrutin som krävdes ännu i ett experimentellt stadie. Därför användes Oracles OCI8-extension i avtalshanteringssystemet. OCI8-extensionen används dock aldrig direkt utan databasförfrågningar görs alltid via en databasklass.

För vissa av de nya funktioner som hade implementerats i avtalshanteringssystemet krävdes det ändringar i den existerande databasen. Dessa ändringar bestod av att nya tabeller behövde skapas och att nya kolumner behövde sättas in i existerande tabeller. För att göra processen enkel att utföra på produktionsdatabasen skapades ett migrationsskript i PHP. Detta migrationsskript körde ett antal SQL-satser vilka senare kunde användas för att utföra de nödvändiga ändringarna på produktionsdatabasen.

3.5.1 Databasklass

För att enkelt kunna köra SQL-satser mot den databas som skulle användas av avtalshanteringssystemet skapades en databasklass. I databasklassen kan anslutningsinställningarna lätt anges i konstruktorn där de sparas som medlemsvariabler. Eftersom databasklassen används överallt där webbservern behöver åtkomst till databasen, är det lätt att ändra anslutningsinställningarna mellan utvecklingsdatabasen och produktionsdatabasen. Databasklassen öppnar anslutningen till databasen då första SQL-satsen körs eller då den manuellt öppnas med hjälp av ett metodanrop. Anslutningen stängs antingen via ett metodanrop eller i klassens destruktor.

En av metoderna i databasklassen erbjuder möjligheten att exekvera en SQL-sats med parametrar med hjälp av "prepared statements". Då prepared statements används i PHP med OCI8-extensionen tolkas först SQL-satsen med funktionen "oci_parse". Med hjälp av funktionen "oci_bind_by_name" kan platshållarvariablerna i SQL-satsen kopplas till en variabel i PHP. En parameterklass skapades för att representera denna koppling i kod. Parameterklassen är en enkel datastruktur vilken innehåller namnet på platshållarvariabeln, samt den PHP-variabel som skall bindas till platshållarvariabeln. Dessa värden initialiseras i konstruktorn hos parameterklassen.

Vanligtvis ses varenda SQL-sats som en enskild transaktion, vilket är standardbeteendet för OCI8-extensionen. För att köra flera SQL-satser i samma transaktion kan ett frivilligt argument till exekveringsmetoden, vilken skjuter upp transaktionens slutförande. För att slutföra transaktionen kan en metod i databasklassen användas eller den sista SQL-satsen i transaktionen köras utan den frivilliga parametern.

Figur 31 visar hur databasklassen kan användas för att utföra en transaktion vilken innehåller två SQL-satser. Då första SQL-satsen exekveras öppnas anslutningen till databasen, vilket även påbörjar en transaktion. Det sista argumentet i metoden execute är frivilligt, och används för att skjuta upp transaktionens slutförande. Transaktionen

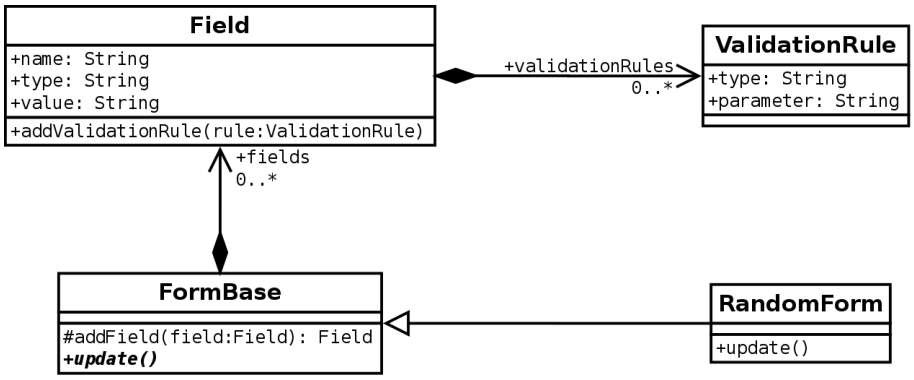
består i det här fallet av två SQL-satser. I detta fall används metoden “commit” för att slutföra transaktionen. Ifall ett fel uppstår då SQL-satserna tolkas eller exekveras kastas ett undantag. Databashanteraren ångrar då ändringarna då anslutningen stängs. Eftersom databasanslutningen i detta fall inte längre behövs används metoden “closeConnection” för att stänga anslutningen.

```
try {
    $db = new Database;
    $db->execute("INSERT INTO CAR_COLORS (CARCOLORS_COLOR) VALUES (:ccolor)", [
            new Parameter(":ccolor", "RED"),
        ], Database::NO_COMMIT);
    $db->execute("UPDATE CARS SET CAR_COLOR = :ccolor WHERE CAR_ID = :cid", [
            new Parameter(":ccolor", "RED"),
            new Parameter(":cid", 4)
        ], Database::NO_COMMIT);
    $db->commit();
    $db->closeConnection();
} catch (Exception $e) {
    echo $e->getMessage();
}
```

Figur 31: Exempel på hur databasklassen kan användas.

3.6 Hantering av avtalsdata

Den avtalsdata som kan visas och hanteras av användare är uppdelad i kategorier. Kategorierna består av flikar vilka innehåller formulär där tillhörande data visas. Flikarnas uppbyggnad beskrivs i kapitel 3.4.5. Formuläret beskrivs i kod som en klass, vilken innehåller fält. Figur 32 visar ett exempel på ett klassdiagram för ett formulär. En del data behöver presenteras i tabellform, för vilken en tabellklass skapades.



Figur 32: Klassdiagram över implementering av ett exempelformulär.

3.6.1 Formulär

Funktionaliteten för att hantera formulär har delats upp i flera delar. Ett formulär implementeras som en domändel och en vydel. Domändelen för ett formulär består oftast av en klass, vilken innehåller den data som finns på formuläret. Vydelen presenterar den data som finns i domändelen åt användaren. Vyn skapas med hjälp av en formulärgenerator. Formulärgeneratorn gör det möjligt att skapa formulär på ett konsistent

sätt och lätt implementera de olika typer av fält och de valideringsregler som används. Formulärgeneratorn beskrivs i 3.6.3.

Formulär representeras i domänen som klasser vilka ärver från en formulärbasklass. Formulärbasklassen gör det möjligt för utvecklare att skapa och hantera formulären på liknande sätt. Formulärbasklassen innehåller en lista med fält, vilka beskrivs i kapitel 3.6.2. Listan med fält används främst av en formulärkontrollerklass, vilken uppdaterar data i fälten från den data som skickats av användaren då data sparas.

En vanlig formulärklass kör i konstruktorn en eller flera SQL-förfrågningar med hjälp av en instans av databasklassen. Utifrån den data som hämtas från databasen skapas fält med hjälp av metoden “addField” i formulärbasklassen. Fält sparas dels i en lista i basklassen och dels som medlemsvariabler i formulärklassen för enkel åtkomst.

Avtalshanteringssystemet innehåller många specifika krav på den data som får lagras i databasen eller krävs för att överhuvudtaget kunna visa data. Dessa krav kontrolleras med hjälp av SQL-förfrågningar. Som felhantering kastas ett undantag ifall ett krav inte uppfylls . Felhanteringen beskrivs i kapitel 3.6.6. Viss automatisk uppbyggnad av SQL-satser används i vissa fall där data hanteras i form av en tabell. Detta sker genom en tabellklass, vilken beskrivs i kapitel 3.6.5.

3.6.2 Fält

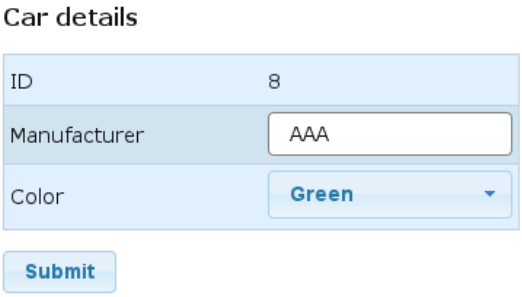
Ett formulär består av flera fält. I HTML-kod presenteras fält normalt som ett input-element med en beskrivning. Fält kan dock variera i typ och funktionalitet. Typen på ett fält kan till exempel vara text, ett nummer eller ett datum. Utöver detta kan fältet innehålla ytterligare begränsningar, vilka till exempel kan vara att de är skrivskyddade eller har ett max antal tecken som får matas in. För att enkelt kunna definiera fält i PHP-kod skapades en fältklass, vilken innehåller den nödvändiga informationen för att representera fältet i HTML-kod.

Fält innehåller ett namn, vilket är det namn som tilldelas input-elementet i HTML-koden, en typ, ett värde, samt alternativt en lista med valideringsregler. Valideringsreglerna tillämpas på klientsidan då formulärgeneratorn skapar input-elementet för fältet.

3.6.3 Formulärgenerator

Formulärgeneratorn skapades ursprungligen som en hjälpklass i PHP för att underlätta byggandet av formulär i HTML-kod. Formulär ritas upp med hjälp av en tabell med minst två kolumner och en rad per fält. För att lätt kunna skriva ut ett fält skapades en metod vilken tog emot en fältbeskrivning och ett fält som argument och skrev ut en tabellrad på sidan. Med tiden utökades funktionaliteten hos formulärgeneratorn. Den används nu allra främst för att skapa input-element med typ och valideringsregler

bestämda från de fält som finns definierade med hjälp av fältklassen. Figur 33 illustrerar hur ett formulär kan se ut.



Figur 33: Exempel på formulär.

Formulärgeneratorn består även av ett skript skrivet med JavaScript och jQuery. Detta skript har många uppgifter. Eftersom webbapplikationen inte ska byta sida då ett formulär skickas, vilket är det normala beteendet hos formulär i HTML, används jQuery för att ändra funktionaliteten hos formulärinskickningen. Till den knapp vilken sparar ändringarna kopplas en funktion vilken blockerar det normala beteendet, samlar ihop och skickar formulärdata med hjälp av jQuerys AJAX-funktionalitet. Skriptet har även som uppgift att tillämpa formulärvalidering med hjälp av jQuery Validation Engine. Valideringen beskrivs i mer detalj i kapitel 3.6.4. Detta skript initialiserar även de fält vilka använder sig av jQuery UI:s gränssnittskomponenter, till exempel tillämpar datumväljare på datumfält och autokomplettering på de fält vilka använder sig av det.

3.6.4 Validering av data

För att validera data på klientsidan används insticksmodulen jQuery Validation Engine. Validation Engine används för de flesta valideringsjobb i de fall då det är möjligt att validera data på klientsidan. Ifall användaren matar in felaktig data, markerar Validation Engine de felaktiga fälten för användaren och förhindrar att formuläret skickas. Validation Engine tillåter utvecklare att specificera anpassade funktioner för validering. Denna funktionalitet har använts då mer specifika valideringsjobb krävts.

3.6.5 Tabeller

En del av den data som kan hanteras presenteras i form av en redigerbara tabeller åt användaren. Det tidigare avtalshanteringssystemet använde sig av en tabellkomponent för ASP vilken inte kunde återanvändas. För hantering av tabeller skapades en ny klass vilken kan användas som bas för tabellklasser. Eftersom mängden data i dessa tabeller normalt inte är speciellt stor kan hela tabellen redigeras på en gång.

Tabeller är implementerade med hjälp av en basklass för tabeller. Klasser vilka ärver från denna basklass kan specificera vilka kolumner som ingår i tabellen och är redigerbara. Tabeller fungerar som ett mer specifikt fall av ett formulär och innehåller därför en lista med fält precis som andra formulär. Möjligheten att ta bort rader i en tabell kan även kontrolleras med hjälp av ett metodanrop.

Tabellklassen skapar redigerbara fält utifrån den data som finns i databasen samt en ny rad med tomma fält vilken användare kan använda för att lägga till fler rader. Vanligtvis skapar tabellklassen ett vanligt textfält för cellerna i en kolumn. I många fall krävs dock mer anpassningsbarhet för fälten. I det fallet finns möjlighet att ange ett fält som mall för kolumnfälten. Tack vare det kan formulärgeneratorn användas för att skapa fält som i vanliga formulär med till exempel rätt typ, autokomplettering och valideringsregler.

I gränssnittet representeras tabellerna av en "Tabellvy" klass. Denna klass skapar en HTML-tabell med redigerbara fält och kolumnbeskrivningar. Ifall tabellen tillåter borttagning av rader sätts en extra kolumn med kryssrutor till i tabellen, med vilka användare kan markera rader för borttagning. Tabeller fungerar i princip som ett formulär med många fält. Formulärgeneratorn som används för formulär kan då användas för att skapa input-elementen. Figur 34 visar ett exempel på hur en redigerbar tabell kan se ut.

Figur 34: Exempel på en redigerbar tabell.

3.6.6 Felhantering

För felhantering används undantag. Undantag (Exceptions) är en felhanteringsmetod vilken finns i många olika programmeringsspråk. Då ett fel uppstår, kastas ett undantag, som med hjälp av ett try-catch block kan fångas upp och hanteras. PHP innehåller en klass för undantag med vilken man kan skapa undantag med valfria felmeddelanden. Dessa felmeddelanden visas åt användaren då fel uppstår. Fel som kan uppstå är till exempel brist på rättigheter eller felaktiga SQL-kommandon.

3.7 Rapporteringsverktyg

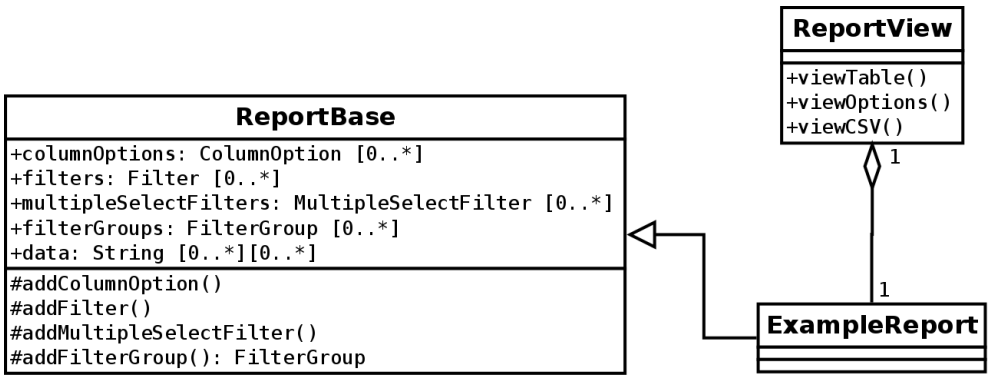
Avtalshanteringssystemet innehåller ett rapporteringsverktyg vilket tillåter användare att skapa anpassningsbara rapporter i tabellform. Gränssnittet för rapporteringsverktyget består av en samling alternativ för att välja kolumner och filtrera data samt själva rapporten i tabellform. Figur 35 visar ett exempel på en rapport i tabellform.

Cars report

ID	Manufacturer	Color
1	AAA	Green
2	AAA	Red
3	AAB	Gold

Figur 35: Exempel på en rapport i tabellform.

Figur 36 visar ett förenklat klassdiagram över hur en rapport kan skapas. Rapporter i tabellform skapades genom att skapa klasser vilka ärver från en basklass för rapporter. Rapportbasklassen tillåter till exempel filter och kolumnval att lätt läggas till i rapporten. Klassdiagrammet presenteras i förenklad form, för att göra funktionaliteten mer överskådlig. Rapportbasklassen bygger vidare på funktionaliteten som hade skapats för tabeller, men innehåller inga fält, eftersom data inte ändras.



Figur 36: Förenklat klassdiagram för en rapport.

3.7.1 Gränssnitt

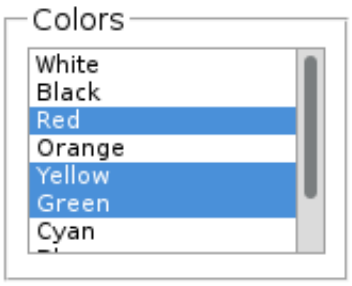
Användare kan anpassa de kolumner som skall framgå i rapporten genom att välja ut önskade kolumner från en lista med tillgängliga kolumner. Dessa listor presenteras med hjälp av jQuery UI som tillåter utvecklare att skapa listor mellan vilka användare kan dra och släppa listinnehållet. För varje rapport som skapas kan vanliga kolumnval markeras som ett standardalternativ.

Rapporteringsverktyget innehåller även filtreringsalternativ som till exempel datumbegränsningar. Filter som hör ihop med varandra kan visuellt grupperas tillsammans med hjälp av HTML-elementet "fieldset". Grupperingen påverkar inte funktionaliteten hos filtren utan endast hur filtren presenteras åt användaren. Rapporten skapas då användaren trycker på en knapp för att skapa rapporten.

3.7.2 Filter

Rapportbasklassen innehåller funktionalitet för att lägga till olika typer av filter. Ett enkelt filter visas som en textbox åt användaren. För att skapa ett enkelt filter anropas en metod i basklassen i vilket den SQL-kod som ska användas i where-uttrycket kan specificeras. Om en användare inte har använt sig av ett filter används det inte.

En annan möjlighet är att lägga in ett flervalsfilter. För att skapa flervalsfilter i HTML används select-elementet med attributet "multiple". Flervalsfilter tillåter användare att markera inga, ett eller flera alternativ ur en lista. Alternativen kan specificeras med hjälp av en lista i kod eller skapas automatiskt utifrån innehållet i databasen. De automatiskt skapade alternativen skapas genom att bygga ett SQL-sats vilket innehåller en "select distinct"-förfrågan med en vald kolumn.



Figur 37: Exempel på hur ett flervalsfilter kan se ut.

3.7.3 Rapportvy

De rapporter som skapas kan visas med hjälp av en rapportvy-klass. Rapportvy-klassen innehåller metoder för att rita upp det gränssnitt som används av rapporteringsverktyget. Det innehåller en metod för att rita upp huvudgränssnittet med valmöjligheter. Den innehåller även en metod för att rita upp den tabell som rapporten anger. Eftersom data sparas på ett enhetligt sätt i rapportbasklassen en enda rapportvyklass användas för att rita upp flera olika typer av rapporter.

3.7.4 Export av rapporter till Microsoft Excel

Rapporteringsverktyget innehåller ett alternativ för att exportera en tabell för öppning i Microsoft Excel. Eftersom PHP inte inkluderar några möjligheter för att skapa filer med Microsoft Excels egna filformat som standard, används CSV som filformat istället. CSV står för Comma-Separated Values och är ett enkelt filformat vilket kan användas för att skapa tabeller som kan öppnas i Microsoft Excel. CSV är ett textbaserat filformat i vilket kolumner i en tabell är separerade med ett separator-tecken och rader med en ny rad. Trots namnet är separatortecknet inte alltid ett kommatecken.

Det finns flera begränsningar med filformatet CSV. För det första beror separatorn på den lokalisering som används på den dator där filen öppnas. Vanligtvis tolkar Microsoft Excel ett semikolon eller ett kommatecken som separator för kolumnerna beroende på operativsystemets lokaliseringsinställningar. Eftersom avtalshanteringssystemet används globalt och inte bara i Finland skapades det en möjlighet att ändra separatorn. En annan begränsning med CSV är att tabellerna är oformaterade. Det betyder att användaren själv måste formatera tabellerna om så önskas.

3.7.5 Diagram

Vissa av rapporterna presenterar data i form av ett diagram. För att rita dessa diagram används jQuery-insticksmodulen Flot, vilken kan rita upp diagram med hjälp av canvas-element vilka ingår i HTML5. Flot valdes eftersom det är en insticksmodul som verkade relativt enkel att använda, var tillräckligt väl dokumenterad och innehåller den funktionalitet som behövdes för att rita upp diagrammen.

För att skapa ett diagram med Flot behöver ett HTML-element med fast bredd och höjd anges, den data som ska visas i diagrammet och alternativt ytterligare inställningar. Den data som Flot använder sig av är uppdelad i serier, vilka innehåller inställningar samt den data i serien, vilken ska ritas. Den data en serie innehåller kan representeras av en lista i JSON-format. Seriedata består av en lista av punkter. Punkter i Flot är en lista med två element, vilka fungerar som koordinater. Flot använder alltid punkter med numerisk data för både x- och y-axeln.

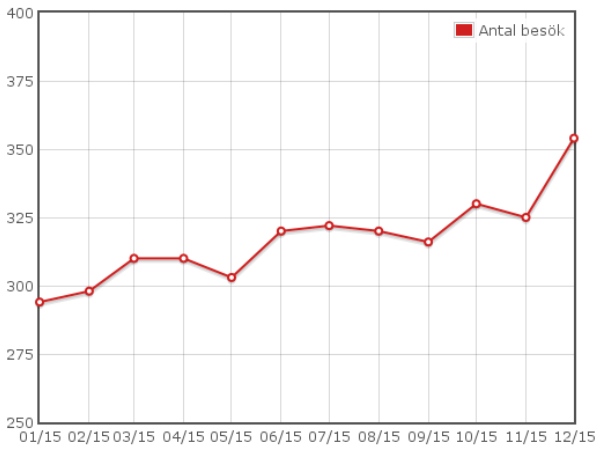
PHP inkluderar funktionen json_encode, vilken tar emot variabler och returnerar en sträng med data i JSON-format. Strängen skrivs i det här fallet ut direkt av PHP inuti ett inbäddat skript i HTML-dokumentet. I detta fall behövdes även möjlighet att visa diagram där en axel representerar tid. För detta användes insticksmodulen “time”, vilken medföljer Flot. Insticksmodulen tillåter en axel att tolkas som tid. Eftersom Flot enbart använder sig av numerisk data anges tiden som en tidsstämpel i UTC (Coordinated Universal Time) format, och konverteras för visning enligt ett valfri tidsformat angivet i inställningarna för axeln.

Figur 38 visar ett exempel i vilket ett diagram vilket visar antal besök per månad under ett år skapas. I detta exempel är #flotGraph ett div-element som har fast bredd och höjd satt med hjälp av CSS. Notera att i den verkliga koden skapas seriedata med hjälp av PHP.

```
$.plot("#flotGraph", [{
    //Inställningar för dataserie
    points: { show: true },
    lines: { show: true },
    color: "#d02020",
    label: "Antal besök",
    data: [
        [(new Date("2015-01-01")).getTime(), 294],
        [(new Date("2015-02-01")).getTime(), 298],
        [(new Date("2015-03-01")).getTime(), 310],
        [(new Date("2015-04-01")).getTime(), 310],
        [(new Date("2015-05-01")).getTime(), 303],
        [(new Date("2015-06-01")).getTime(), 320],
        [(new Date("2015-07-01")).getTime(), 322],
        [(new Date("2015-08-01")).getTime(), 320],
        [(new Date("2015-09-01")).getTime(), 316],
        [(new Date("2015-10-01")).getTime(), 330],
        [(new Date("2015-11-01")).getTime(), 325],
        [(new Date("2015-12-01")).getTime(), 354]
    ]
}], {
    //Diagraminställningar
    xaxis: {
        mode: "time",
        tickSize: [1, "month"],
        timeformat: "%m/%y"
    },
    yaxis: {
        min: 250,
        max: 400
    }
});
```

Figur 38: Exempel på kod för att skapa ett diagram med Flot.

Inställningarna anger en enda serie för vilken både punkter och linjer skall visas. För att få en tidsstämpel i JavaScript-kod, används här metoden getTime på Date-objekt. I diagraminställningarna anges tidsformatet för x-axeln och det område av y-axeln som skall visas. Resultatet av koden i figur 38 blir det diagram som visas i figur 39.



Figur 39: Ett diagram skapad med hjälp av jQuery-insticksmodulen Flot.

4 Resultat och diskussion

Resultatet blev ett webbaserat avtalshanteringssystem som använder sig av moderna webbteknologier. Avtalshanteringssystemet har i stort sett samma funktionalitet som det tidigare systemet, men innehåller några förbättringar. Användningen av PHP gör systemet lättare att underhålla än det gamla och erbjuder bättre möjligheter till fortsatt utveckling. Det uppdaterade användargränssnittet är mer konsistent än i det tidigare systemet. Användare kan dra nytta av mer avancerade sökalternativ och kan även anpassa utseendet genom teman. Efter att systemet hade testats togs avtalshanteringssystemet i bruk på Wärtsilä.

4.1 Diskussion

Jag är i stort sett nöjd med resultatet. Avtalshanteringssystemet fungerade som tänkt och är en klar förbättring till det tidigare systemet. Utvecklingen var även mycket lärorik och jag fick bekanta mig med många av de krav som ställs på ett sådant system. Det finns dock flera förbättringar som skulle kunna implementeras och kod som kunde ha strukturerats på ett annorlunda sätt.

Mycket av den jQuery-kod som skrevs är endast uppdelad i funktioner. För att förbättra strukturen på koden kunde instickningsmoduler till jQuery skapas för mycket av funktionaliteten. Koden som används för att implementera en trädvy för avtalsmenyn kunde till exempel göras om till en återanvändbar instickningsmodul till jQuery.

Formulärgeneratorn utvecklades långt utöver de ursprungliga planerna och innehåller mer funktionalitet än tänkt. Den började som en uppsättning med metoder för att enkelt kunna skapa tabellrader i formulär, men används nu i princip överallt där fält ritas upp. Även om koden fungerar som tänkt, kunde koden delas upp i flera klasser för att lättare kunna hanteras.

Även om diagramfunktionaliteten fungerar som ursprungligen tänkt, finns det möjligheter för förbättring. Flot erbjuder till exempel möjligheter för att skapa interaktiva diagram, vilka kunde göra det enklare för användare att åskådliggöra önskad data. Utöver detta skrivs seriedata i dagsläget ut direkt i ett inbäddat skript. Detta kunde möjligtvis förbättras genom att utöka användningen av AJAX till att hämta seriedata istället för en helt ny sida.

Rapporteringsverktyget kunde till exempel byggas vidare på genom att erbjuda mer exportmöjligheter. Som exempel kunde nämnas formaterade Microsoft Excel-filer eller PDF-filer som lätt kan skrivas ut.

Källförteckning

[1] Wärstilä, 2016. *Wärtsilä*. [Online]
http://www.wartsila.com [hämtat: 20. 03. 2016].

[2] W3C, 2014. *HTML5*. [Online]
https://www.w3.org/TR/2014/REC-html5-20141028/ [hämtat: 07. 03. 2016].

[3] W3C, 2011. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. [Online]
https://www.w3.org/TR/2011/REC-CSS2-20110607/ [hämtat: 29. 03. 2016].

[4] Mozilla Developer Network, 2016. *CSS3*. [Online]
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3 [hämtat: 29. 03. 2016].

[5] Mozilla Developer Network, 2016. *JavaScript Guide*. [Online]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide [hämtat: 07. 03. 2016].

[6] Mozilla Developer Network, 2016. *JSON*. [Online]
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON [hämtat: 24. 03. 2016].

[7] Garrett, J. J., 2005. *Ajax: A New Approach to Web Applications*. [Online]
http://adaptivepath.org/ideas/ajax-new-approach-web-applications/ [hämtat: 07. 03. 2016].

[8] Murphey, R., 2012. *jQuery Fundamentals*. [Online]
http://jqfundamentals.com/legacy/ [hämtat: 07. 03. 2016].

[9] The jQuery Foundation, 2016. *About jQuery UI*. [Online]
https://jqueryui.com/about/ [hämtat: 16. 03. 2016].

[10] Dugas, C. och Refalo, O., 2015. *jQuery.validationEngine v2.6.2*. [Online]
https://github.com/posabsolute/jQuery-Validation-Engine [hämtat: 17. 03. 2016].

[11] Flot, 2014. *Flot: Attractive JavaScript Plotting for jQuery*. [Online]
http://www.flotcharts.org/ [hämtat: 17. 03. 2016].

[12] The PHP Documentation Group, 2005. *PHP Manual*. [Online]
https://secure.php.net/manual/en/index.php [hämtat: 07. 03. 2016].

[13] Oracle, 2016. *Database SQL Language Reference*. [Online]
https://docs.oracle.com/cd/E11882_01/server.112/e41084/toc.htm [hämtat: 31. 03. 2016].

[14] Evolus, 2012. *Pencil Project*. [Online]
http://pencil.evolus.vn [hämtat: 23. 03. 2016].

[15] Wodehouse, C., 2015. *Server-Side Scripting: Back-End Web Development Technology*. [Online]
https://www.upwork.com/hiring/development/server-side-scripting-back-end-web-development-technology/ [hämtat: 15. 03. 2016].

[16] Boesch, J., 2009. *Gritter for jQuery (Growl)*. [Online]
https://web.archive.org/web/20150623202146/http://boedesign.com/blog/2009/07/11/growl-for-jquery-gritter/ [hämtat: 23. 03. 2016].