Webbaserat planeringsverktyg för resurserna i Arcada

Utvecklandet av en prototyp för examensansvariga

John Rothberg

Examensarbete

Informationsteknik

2015

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	5331
Författare:	John Rothberg
Arbetets namn:	Webbaserat planeringsverktyg för resurserna i Arcada - Utveckling av en prototyp för examensansvariga
Handledare:	Magnus Westerlund
Uppdragsgivare:	Hanne Karlsson, Arcada

Sammandrag:
Detta examensarbete handlar om utvecklingen av en klient-applikation för uppdragsgivaren Yrkeshögskolan Arcada. Målet med praktiska delen av arbetet var att programmera som konceptvalidering en prototyp, som ger examensansvariga möjlighet till att effektivt kunna planera resurseringen för personalen i olika utbildningsprogram.

Kraven var att applikationen skulle fungera på en server genom en webbläsare, så att flera användare kunde använda applikationen samtidigt. Användaren skall alltså kunna länka ihop lärare med kurser genom en specificerad mängd timmar per kurs. En lärares timmar under läsårens perioder samt totala timmar under läsår skall kunna uppföljas och uppdateras.

Examensarbetet täcker hela utvecklingsprocessen från projektets planering till den slutliga produkten. Efter introduktionen med bakgrund, målsättningar och problemställningen presenteras all teknisk teori, utvecklingsspråk och programvara som användes för att utveckla prototypen som kallas Resursplaneraren. Det följs av en inblick på datamodellen i projektet och arkitekturen. Till följande behandlas själva utvecklingsprocessen i närmare detalj, hur projektet förverkligades och vad de största utmaningarna var. Sedan presenteras resultaten och till slut diskuteras projektet och resultaten. Examensarbetets huvudsakliga fokus är i att bevisa att det med existerande data kan utvecklas en fungerande webbaserad applikation för att planera läsår för utbildningsprogram och sedan allokera resurser till läsårsplanen.

Nyckelord:	PHP, XML, JavaScript, AJAX, WAMP, Apache, klient-applikation, webbaserad
Sidantal:	37
Språk:	Svenska
Datum för godkännande:	1.6.2015

DEGREE THESIS	
Arcada	
Degree Programme:	Information Technology
Identification number:	5331
Author:	John Rothberg
Title:	Web-based planning tool for resources at Arcada - Prototype development for heads of curriculums
Supervisor:	Magnus Westerlund
Commisioned by:	Hanne Karlsson, Arcada

Abstract:
This thesis encapsulates the development of a client application commisioned by Yrkeshögskolan Arcada. The goal with the practical part of the degree thesis was to develop a planning tool, on a proof-of-concept-level, for allocating staff resources to different courses and curriculums.

The requirements for the project were that the application runs on a server and can be accessed through a web browser with session management supporting multiple users simultaneously. The user was supposed to be able to allocate hours to courses and staff, through different periods in different academic years, thus linking these three entities. Every teacher's allocated hours were then required to be examined and updated through the web application.

The thesis covers the entire development process starting from the planning stages reaching all the way to the finished product. After the introductive part of the thesis all the technical theory is presented, with all programming languages and development tools used to develop the prototype application called Resource Planner. This is followed by a brief look into the project's data model and architecture, after which the development process itself comes into focus, how the project was realized and what the biggest challenges were. Finally the results are presented and discussed. The main focus of the thesis is to prove that using existing data a web-based application for planning an academic year for a curriculum, and allocating resources to it, can be developed.

Keywords:	PHP, XML, JavaScript, AJAX, WAMP, Apache, client application, web based
Number of pages:	37
Language:	Swedish
Date of acceptance:	1.6.2015

INNEHÅLL

Figurer

Tabeller

FÖRORD

Planeringen av detta examensarbete påbörjades på hösten 2014. Ursprungligen var det meningen att utveckla en produkt som skulle tas i bruk på våren 2015 för att planera resurserna för läroplaner för läsåret 2015-2016. Projektet visade sig vara mycket svårare att göra än vad jag och beställaren från början förväntade oss, och kravspecifikationen gjordes om så att den bättre passade in i tidsramen för våren 2015. Istället för att programmera en färdig produkt som tas i bruk blev målet att utveckla en prototyp.

Projektet var väldigt utmanande och intressant att jobba med. Jag tackar alla kurslärare i Arcada för att ni har inspirerat mig och lärt mig tänka som en ingenjör. Jag lyfter fram Magnus Westerlund och Jonny Biström, som har haft den största inverkan på mitt sätt att arbeta och lösa problem inom IT-branschen.

Jag vill tacka min familj och mina närmaste vänner för att ni stött mig under vad som visade sig vara en oerhört svår studietid. Utan er skulle jag aldrig ha kommit såhär långt.

Framför allt vill jag tacka Nina Rajala, min sambo, partner och förlovade. Ditt stöd och uppmuntrande har burit mig. Du är guldvärd.

1 INLEDNING

Detta examensarbete är indelat i sex kapitel. I inledningen presenteras bakgrunden och problemställningen för projektet som utfördes under tidsramen november 2014 - maj 2015.

I kapitel två redogörs den tekniska teorin i fråga, d.v.s. alla programmeringsspråk och utvecklingsverktyg som användes för att bygga upp produkten. Här presenteras också arkitektur och omgivning, m.a.o. existerande, tillgänglig data, datamodellerna, servern samt databasen. Detta är viktig information i synnerhet ifall någon t.ex. väljer att vidareutveckla projektet eller installera applikationen på en webbserver.

Det tredje kapitlet, "utvecklingsprocessen", beskriver utmaningar och nyckelelement i utvecklandet av webbapplikationen. Här presenteras också applikationens användargränssnitt och utseende.

Det fjärde kapitlet behandlar resultaten. Här redogörs hur bra projektet motsvarade målen och vilka brister Resursplaneraren har.

I kapitel fem diskuteras produktens slutgiltiga form, hur den motsvarar de ursprungliga målen, vilka brister den har samt vilka möjligheter för vidareutveckling som har framkommit.

1.1 Bakgrund

Examensansvariga i Arcada har hittills producerat årsplaner och läroplaner m.h.a. Arcadas egna kursverktyg, webbportalen ASTA och en .xls kalkyltabell, utan någon sorts intelligent automatik. ASTA (Arcadas STudieAdministration) är en klient-applikation i Arcadas inre nätverk för studenter och personal. I ASTA hittas all information om kurser, läroplaner och personal. ASTA fungerar också som verktyg för anmälningar till kurser, anmälningar till tenter, vitsordsregistrering m.m. Från ASTA kan man också få ut den information som Resursplaneraren har byggts runt, läroplaner i XML-format. Att planera, bygga samt se på läroplaner genom Microsoft Excel är ineffektivt både tids- och resultatmässigt. Man kan inte effektivt filtrera vad man vill se på. (Arcada Wiki, 2015).

1.2 Problemställning

Det går inte att överföra kursinformation från ASTA till MS Excel på ett effektivt och smidigt sätt. Arcadas personal har inget modernt, effektivt verktyg m.h.a. vilken de kan planera läsåret. Diverse linjers examensansvariga behöver ett verktyg för att planera läsåret. Ett nytt verktyg skall byggas upp som svarar på frågor såsom: Vilken lärare är ansvarig för vilken kurs? Vilka kurser hör till vilket utbildningsprogram? Hur många timmar arbete har en lärare i period X? Hur många kurser har utbildningsprogram Y i period Z?

1.3 Mål och syfte

Prototypen skall erbjuda möjligheten att genom ett webbaserat användargränssnitt kunna hantera xml-data, som fås från ASTA. Genom att skapa hanterbara objekt utgående från xml-data är det meningen att en användare skall kunna importera, kolla upp och modifiera planer. Man skall kunna se på olika utbildningsprogram, deras kurser och deras personal, och samtidigt göra de ändringar som krävs inom ett läsår. Framför allt skall man kunna allokera arbetstimmar till personal, d.v.s. vilka lärare ansvarar för vilka kurser, och hur många timmar. Resursplaneraren skall vara ett fristående verktyg. Kärnan av arbetet är att utveckla en prototyp, programvara som kan utnyttja gammal kursinformation för att konstruera en ny sorts plan som inte behandlar kursernas innehåll eller tidtabell.

1.4 Avgränsning

Resursplaneraren skall endast koppla ihop personal, arbetstimmar och kurser. Prototypen behöver inte heller någon export funktion. För enkelhetens skull kom vi tillsammans med beställaren överens att vi bara behandlar en läroplan. Som personal fungerar tio lärare, och varje kurs i läroplanen skall ha 100 timmar.

1.5 Utgångspunkter

Via kurserna Introduktion till programmering, Algoritmer och datastrukturer, Databaser och SQL, Systemdesign och UML, Klientprogrammering med JavaScript och Serverprogrammering med PHP har jag lärt mig tillräckligt för att kunna ta emot ett projekt som Resursplaneraren. Jag har också jobbat på egen hand med phpMyAdmin, MySQL-front-end verktyget, som använts i detta projekt före jag började jobba på Resursplaneraren.

Jag har inte under projektet haft tillgång till MS Excel kalkyltabellerna som använts för planeringen, men planeringsmetoden visades åt mig kort. Detta gjordes för att ge mig en bättre bild om vad problemet är och vad det är som behövs. Planeringssystemet som Resursplaneraren skall ersätta är funktionellt bristande och föråldrat. Det är visserligen smidigt att göra små ändringar i en existerande läroplan med gamla metoden, t.ex. mellan läsår. Problemet är att gamla metoden inte alls är anpassad för flera läroplaner och allting måste göras manuellt. Att modifiera en fil istället för dataobjekt på ett nätverk ställer till med utmaningar för alla som är med och planerar. Det finns alltid bara en fil som är aktuell. Hur skall den distribueras och editeras? Examensansvariga för Institutionen för ekonomi och affärsanalys önskar att flytta sig till en webbaserad version.

2 TEKNIK, ARKITEKTUR OCH OMGIVNING

Detta kapitel presenterar de nyckelord som berör arbetet och programmeringsspråk som använts för att utveckla Resursplaneraren, och vilket syfte de tjänar i detta examensarbete. Det behövs en hel del olika verktyg för att bygga upp något som i sitt utseende kan verka simpelt.

2.1 XML

XML (Extensible Markup Language) är en textbaserad standard för representation av data. Ett XML-dokument består av en hierarki av objekt som kallas *element*, ibland alternativt *block*. Elementen kan sedan i sig innehålla flera *attribut*, och attributen kan tilldelas värden. (W3C, 2015).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<boksamling>
    <bok sprak="engelska">
        <titel>XSLT Cookbook</titel>
        <forfattare>Sal Mangano</forfattare>
    </bok>
    <bok sprak="svenska">
        <titel>Skriv med XML</titel>
        <forfattare>Åsa Blom</forfattare>
   </bok>
</boksamling>
```

Figur 1: Exempel på XML hierarki. Bild från Wikipedia. Tillgänglig: http://sv.wikipedia.org/wiki/XML.Hämtad 18.5.2015.

Som Figur 1 ovan demonstrerar, är XML ett sätt att presentera eller lagra data enligt ett hierarkiskt system. Enligt detta exempel, är det huvudsakliga objektet en bok. Alla böcker tillhör en boksamling. Varje bok har som attribut ett språk, och underelementen titel och författare. Underelementen kallas *barn*, och elementet "ovanför" kallas en *förälder*.

I XML-datat som Resursplaneraren jobbar med är huvudobjektet en kurs, istället för en bok som i exemplet ovan. XML är ingenting jag själv valde använda, utan det är vad som är tillgängligt genom en data dump från ASTA. Jag går djupare in i XML-kodens presentation avsnitt 2.8 XML-data.

2.2 HTML

HTML (HyperText Markup Language) är den allmänna standarden för källkoden överallt på internet. HTML-koden tolkas av en webbläsare och kan sedan ritas ut på skärmen.

```
<!DOCTYPE html >
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=utf-8" />
<meta name="viewport"
content="initial-scale = 1.0, maximum-scale = 1.0, user-scalable = no, width = device-width">
</head>
<body>
<div>
    <p>Hej! Här är en exempelparagraf.</p>
</div>
</body>
</html>
```

Figur 2: Exempel på HTML källkod.

HTML-kod liknar XML-kod. Den bygger också på ett hierarkiskt system med element, attribut och värden. Exemplet i Figur 2 ovan beskriver källkoden för en simpel webbsida. Ifall man skulle öppna denna fil i en webbläsare skulle man endast se en vanlig text högst uppe till vänster där det står "Hej! Här är en exempelparagraf." All annan källkod finns bara där för att berätta om hur webbläsaren skall tolka innehållet. Endast texten innanför elementet ... skulle synas på webbsidan.

2.3 CSS och PureCSS

CSS (Cascading Style Sheets) är ett programmeringsspråk m.h.a. vilket man kan styra utseendet med på en webbsajt. CSS är inte ett måste, men man kan säga att så gott som alla webbsajter använder CSS. Med CSS kan man styra ett HTML element eller en grupp av element, d.v.s. man kan ge en klass eller id CSS egenskaper och sedan kalla på klassen eller id till de element man önskar ärva de egenskaperna som man tilldelat klassen/id. Dessa egenskaper kan vara t.ex. positionering, storlek, färg, effekter m.m.

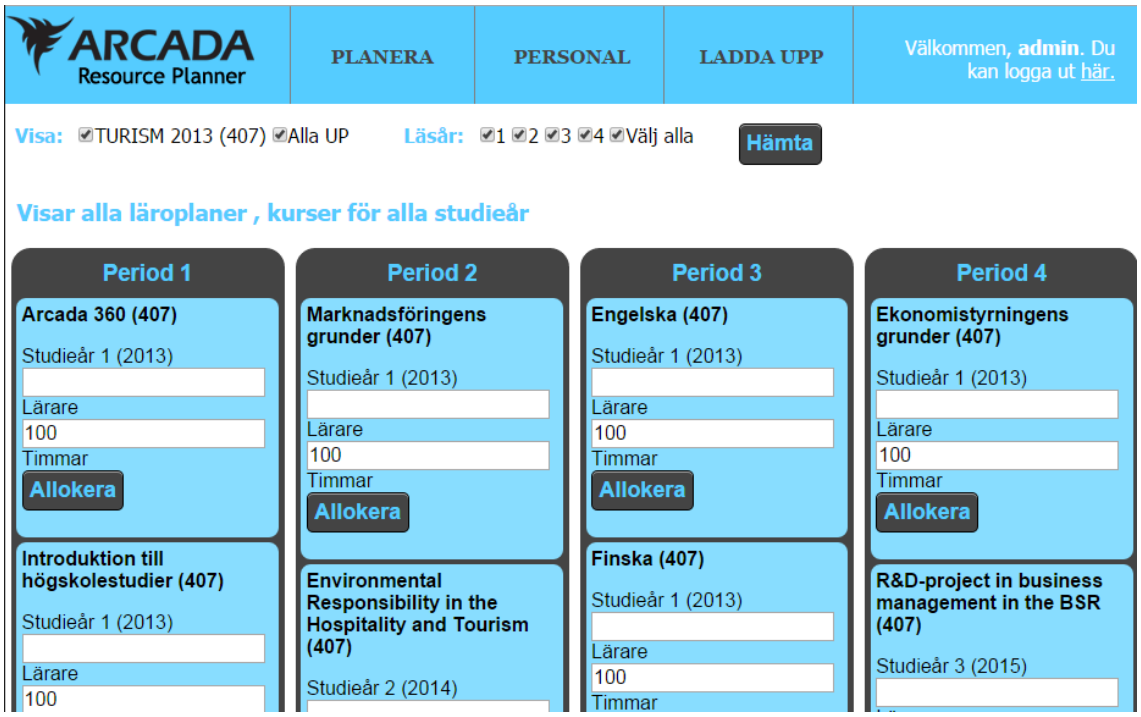
Idén med CSS är att skilja åt utseende från innehåll. CSS reglerar alltså utseendet på HTML innehållet.

Resursplanerarens CSS är inte något vidare komplicerat eller sofistikerat. En bra design och utläggning (layout) är nödvändig, men prototypens simpelhet kräver inte något komplicerat. För att visa flera kolumner smidigt och effektivt valde jag att implementera PureCSS. Pure är ett bibliotek, en öppen källkod, som ger möjligheten att modulmässigt och enkelt kunna framkalla en responsiv design (PureCSS, 2014).

Exempel: Pure möjliggjorde en 4-kolumn-layout väldigt enkelt:

```
.pure-u-1-4 {
        width: 24%;
        margin: 1% 0.5% 1% 0.5%;
}
```

Genom att programmera in 4 kolumner, HTML elementet , och kalla på .pure-u-1-4 klassen för kolumnerna gav det mig den layout jag ville ha för 4 perioder för ett läsår. Varje kolumns bredd blir alltså 24% av deras förälder. 4 x 24% = 96%. Vidare har varje kolumn 0,5% tomt utrymme på både vänster och höger sida. 8 x 0,5% = 4%. Summan blir då 100%.



Figur 3: Skärmkapning av Resursplaneraren med PureCSS fjärdedelskolumner.

2.4 XAMPP-stacken

XAMPP är ett paket mjukvara för server- och webbutveckling. X står för operativsystemet som XAMPP installeras i. Resursplaneraren utvecklades i en Microsoft Windows omgivning. Då kallar vi alltså stacken WAMP (Windows Apache MySQL PHP) som är ett öppet källkodspaket för enkel installation av en webbserver. Jag valde att utveckla Resursplaneraren på en WAMP-server, för det passade bra till projektet. Dessutom sparar det tid att utnyttja WAMP, då jag har erfarenhet av att installera och konfigurera en WAMP-server. Det går alltså inte onödig tid till att studera hur det skall göras.

Webbservern Apache, levererad av ASF (Apache Software Foundation), är världens mest populära HTTP (HyperText Transfer Protocol) servermjukvara. (ASF, 2015).

Apache var valet för servermjukvara därför att det är gratis, öppen källkod, och jag visste från förr att det stöder MySQL och PHP, som är två nyckelkomponenter för Resursplaneraren.

2.4.1 MySQL

Av databashanteringssystem av öppen källkod är MySQL den mest populära i världen. MySQL fungerar med programmeringsspråket SQL (Structured Query Language). SQL är ett väldigt enkelt programmeringsspråk, som grundar sig på idén att man hämtar data från en databas med hjälp av frågor, eller *queries.* (MySQL, 2015).

Resursplaneraren har en MySQL relationsdatabas som består av fyra olika tabeller. Dessa tabeller diskuteras närmare i kapitel 2.9. En relationsdatabas innebär en databas där tabellerna är kopplade till varandra via en eller flera kolumner. Genom detta samband kan man då ställa avgränsningar mellan tabellerna, så att informationen alltid stämmer överens. Exempelvis har Resursplaneraren en tabell i databasen som heter "larare", och en annan som heter "timmar". Ett samband mellan tabellerna kan då skapas så att den ena kontrollerar den andra. Tabell 1 och 2 nedan presenterar fiktiv data.

id	**fornamn**	**slaktnamn**	**anv**
1	Kalle	Karlsson	karlssok
2	Anna	Andersson	anderssa

Tabell 1: Databastabell dit personal sparas.

id	**larar_kod***	**kurs_id**	**timmar**	**period**	**ar**
1	1	15123	50	2	2013
2	1	15124	100	3	2013
3	2	15125	65	1	2013

Tabell 2: Databastabell dit timmar sparas.

I Tabell 1 ser vi exempel på personal information med ett unikt identifikationsnummer för varje lärare i första kolumnen. Andra kolumnerna sparar förnamn, släktnamn och användarnamn. Tabell 2 ser vi exempeldata på arbetstimmar. Varje rad i tabellen motsvarar en allokering mellan en lärare och en kurs. Varje allokering har en unik identifikationsnummer, id. De andra kolumnerna larar_kod, kurs_id, timmar, period och

ar behöver i denna tabell inte vara unika. Vi fokuserar på andra kolumnen som är utmärkt med stjärna. Larar_kod hänvisar till (har ett samband) en kolumn i en annan tabell. Då man skapar tabellen kan man tilldela ett *index* till en kolumn, dessa kallas *nycklar*. En *primärnyckel* i en tabell är ett unikt värde, som man kan skapa samband till. I detta exempel är id-fältet i Tabell 1 en primärnyckel, och i Tabell 2 är larar_kod-fältet en *främmande nyckel*. Detta betyder att lärar_kod måste stämma överens med id fältet i Tabell 1. Sambandet kan specificeras på flera olika sätt, enligt MySQLs egna språk, t.ex. kan man till larar_kod-fältet tilldela egenskaperna "FOREIGN KEY (larar_kod) REFERENCES Tabell1(id) ON UPDATE CASCADE". Detta betyder att ifall man ändrar på informationen i en rad i Tabell 1 ändras informationen i Tabell 2 automatiskt. Man kan i normala fall inte radera en rad som har ett samband. Detta kan endast göras ifall fälten är specificerade så (t.ex. ON DELETE SET NULL). Med denna teknik kan man bygga en större helhet av tabeller som fungerar väl. Detta hjälper till med dataintegritet och samtidigt drar man funktionell nytta. (Connolly & Begg, 2005).

2.4.2 PHP

PHP (PHP Hypertext Processor) är ett programmeringsspråk som körs på en server. PHP kan programmeras objektorienterat som sådant, men PHP används nästan enbart till en webbapplikations logikhantering på en server. PHP källkoden nås inte från webbläsaren förutom med metoder som rätt fram styr information till klientsidan, såsom exempelvis echo() och print() gör. All annan PHP-kod exekveras på webbservern då en fil med PHP-kod i sig öppnas via webbläsaren och det ser inte slutanvändaren. (The PHP Group, 2015).

```php
<?php $page_title = 'sidans titel'; ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
 <title><?php echo $page_title; ?></title>
 </head>
 <body>
 <p>Hello</p>
 </body>
</html>
```

Figur 4: Exempel på PHP-kod blandat in i HTML. Bild från Wikipedia. Tillgänglig: http://sv.wikipedia.org/wiki/PHP#Syntax. Hämtad: 12.5.2015.

I Figur 3 ser vi ett HTML block som kallar på Pure klassen som nämndes i kapitel 2.3. Det följs av PHP avgränsaren "<?php". Allt som följer avgränsaren tolkas av PHP som PHP kod och skall skrivas enligt PHPs egen syntax tills man kommer till den andra avgränsaren "?>". PHP är ett utmärkt programmeringsspråk för att dynamiskt innehåll för en webbsajt, såsom hantering av databasinformation.

I Figur 3 ser vi två rader PHP-kod i HTML källkoden. Man känner igen PHP igenom avgränsaren "<?php", som markerar början på PHP-kod. Allt som följer det tolkas som PHP och skall skrivas enligt PHPs egen syntax tills man kommer till den andra avgränsaren "?>", som markerar kodens slut. I detta exempel sparas teckensträngen "sidans titel" i en variabel ($page_title). I andra raden ser vi "echo $page_title", vilket skriver ut variabeln $page_title. Då denna rad finns i HTML elementet <title> skulle denna text synas högst upp i webbläsaren, som webbsidans namn.

Nästan all logik i Resursplaneraren är i form av PHP. Förutom några få JavaScript funktioner för kontroll på checkboxar är all datainsamling och logik utförd genom PHP. Det är ett utmärkt programmeringsspråk för att dynamiskt kontrollera innehåll på en webbsajt, med massor av funktioner till förfogande. Resursplanerarens PHP-kod presenteras närmare i kapitel 3.2 och 3.4.

2.5 JavaScript och AJAX

JavaScript är ett skriptspråk som körs på klientsidan. Webbläsaren tolkar JavaScript, och JS används oftast för att öka funktionaliteten på en webbsida då HTML inte räcker till, exempelvis ifall någon matematisk beräkning skall göras. JS är händelse-orienterat, vilket betyder att ett JS skript alltid körs när något specificerat händer, exempelvis då man trycker på en knapp.

AJAX (Asynchronous JavaScript And XML) är en samling av sinsemellan kopplade webbutvecklingstekniker m.h.a. vilken man kan bygga mer dynamiska webbsidor. Man kan göra små ändringar på en webbsida utan att ladda hela sidan om, såsom man annars skulle göra. Med AJAX kan man t.ex. skicka XmlHttpRequests från den filen webbläsaren har laddat upp till en annan fil, där det processeras och svaret kan sedan visas upp på webbsidan man ser på. Då du skriver något i sökfältet på Google och Google föreslår olika sökalternativ på basen vad du skriver (Google Suggest) är ett exempel på en tillämpning av AJAX.

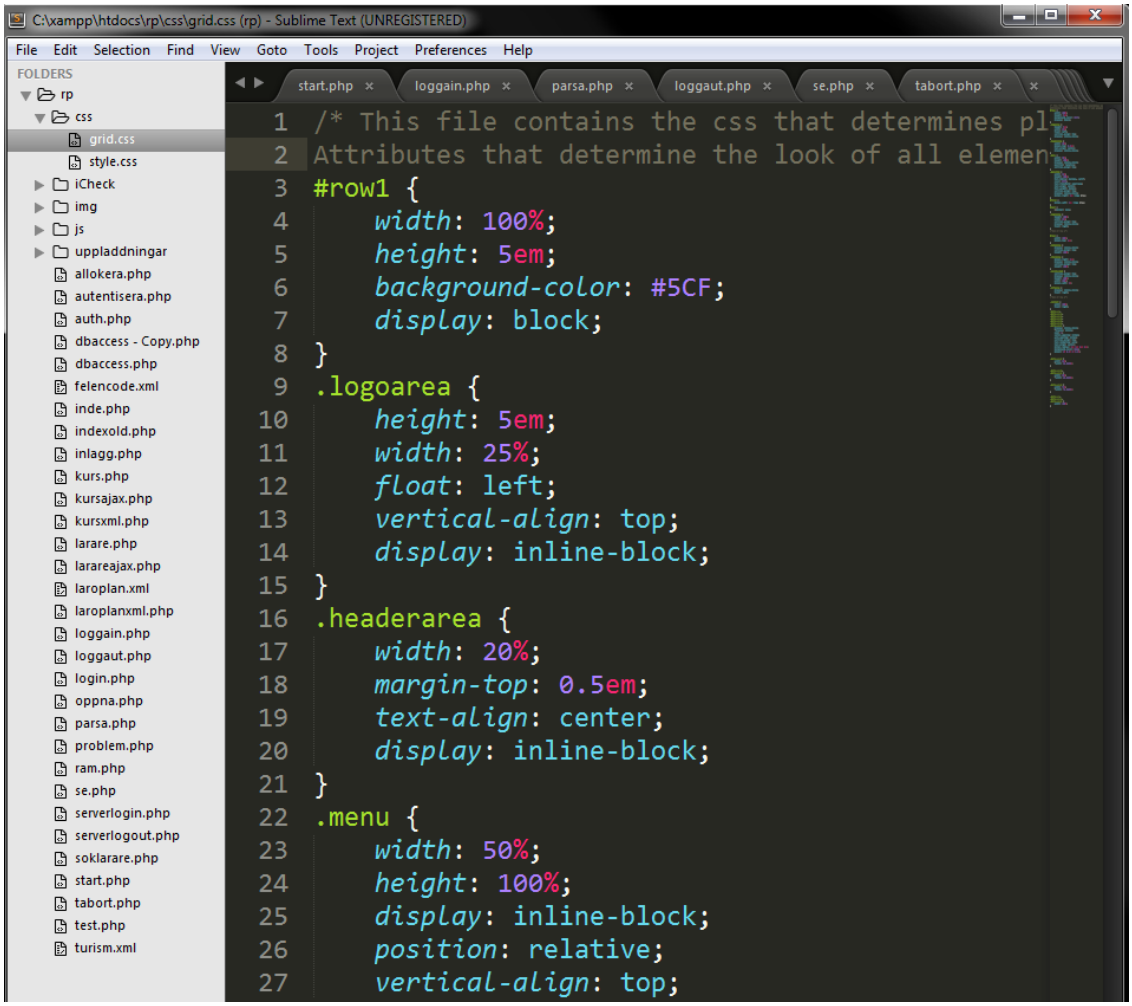
Resursplaneraren behöver AJAX för att uppdatera läro- och resursplanerna direkt då ändringarna görs. Detta är nödvändigt för att det annars skulle vara väldigt klumpigt att varje gång ladda om sidan efter varje ändring bara för att se ändringarna.

```
82      function allokera(id, period, studiear) {
83          var kurs = id;
84          var div_id = "a"+id;
85          var larare = "l"+id;
86          var timmar = "h"+id;
87          var period = period;
88          var studiear = studiear;
89          var lar = document.getElementById(larare).value;
90          var tim = document.getElementById(timmar).value;
91          if(window.XMLHttpRequest) {
92              // code for IE7+, Firefox, Chrome, Opera, Safari
93              xmlhttp=new XMLHttpRequest();
94          } else {  // code for IE6, IE5
95              xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
96          }
97
98          xmlhttp.onreadystatechange=function() {
99              if(xmlhttp.readyState==4 && xmlhttp.status==200) {
100                 document.getElementById(div_id).innerHTML = xmlhttp.responseText;
101             }
102         }
103         xmlhttp.open("GET", "allokera.php?kurs="+kurs+"&lar="+lar+"&tim="+tim+"&studiear="+studiear+"&period="+period, true);
104         xmlhttp.send();
105     }
```

Figur 5: En JavaScript funktion med AJAX.

2.6 Verktyg

Det användes två olika verktyg för att utveckla Resursplaneraren. All källkod skrevs med Sublime Text 3. ST3 är ett av de mest omtyckta verktygen för att utveckla webbsidor bland webbdesigners. Sublime skiljer sig från andra text editorer och verktyg därför att den anpassar sig till användaren i väldigt hög grad. Sublime är anpassande på det sättet, att den är väldigt lättviktig, men det finns massor av tillägg (plugins) som bjuder på ytterligare funktionalitet för Sublime, såsom klass- och funktionsbibliotek. Dessa är lätta att installera och skräddarsy för varje projekt, så allting fungerar precis som man vill för varje projekt man jobbar med. Projektspecifika inställningar hittar man också i andra verktyg, men de är ofta väldigt tunga och tröga att jobba med, t.ex. Eclipse eller Adobes Dreamweaver.

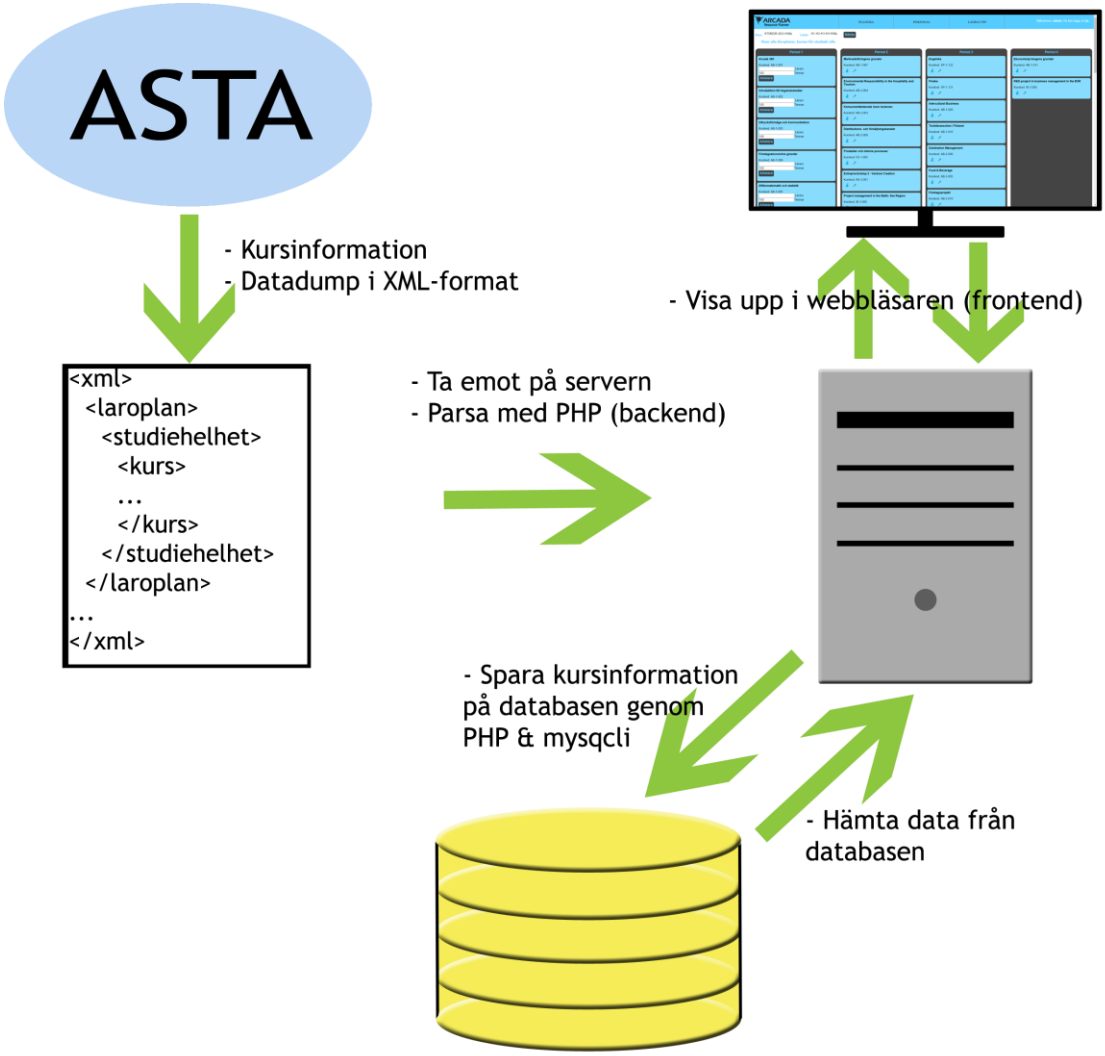


Figur 6: Skärmkapning ur Sublime.

I Figur 6 ovan ser vi en skärmkapning av Sublime Text. Projektets filträd i vänstra spalten, text editorn i mitten och den lilla balken på högra sidan är en komprimerad visuell representation av källkoden som syns i mitten. Man kan trycka med musen var som helst i balken och då hoppar editorn vart man har tryckt. Det är behändigt då man behandlar långa kodsnuttar och hoppar mycket fram och tillbaka i koden.

Med WAMP kommer phpMyAdmin, ett verktyg programmerat med PHP för att administrera MySQL via en webbläsare. PMA användes i projektet för att bygga upp databasen. PMA var ett klart val för mig, då jag har använt den tidigare och den tillhör WAMP-stacken.

2.7 Arkitektur och omgivning



Figur 7: Dataflödesdiagram av Resursplaneraren.

Såsom Figur 7 ovan visar kommer gammal kursinformation från ASTA. Det är det enda data som är tillgängligt för användning i tillräckligt stor skala, hela utbildningsprogram kan dumpas på en gång till databasen. All information man vill ta ut från XML koden är tillgänglig, men Resursplaneraren behöver bara en del av koden.

Resursplaneraren fungerar på vilken som helst server som har XAMPP version 1.4.11 eller nyare. XAMPP 1.4.11 har MySQL 4.1, som är den äldsta versionen av MySQL som är kompatibel med Resursplaneraren. (Apache Friends, 2015).

Applikationen utvecklades i XAMPP (för Microsoft Windows) version 5.6.8, och det är det enda som behövs. En Apache-webbserver, en MySQL-databas samt en PHP5-installation.

2.8 XML data

För utvecklandet av Resursplaneraren fick jag en läroplan att jobba med, Turism 2013. I examensarbetets bilagor kan XML källkoden studeras närmare. I detta avsnitt presenteras endast den information som måste finnas för att Resursplaneraren skall fungera. Med andra ord presenteras den information som sparas i Resursplanerarens databas och måste finnas där.

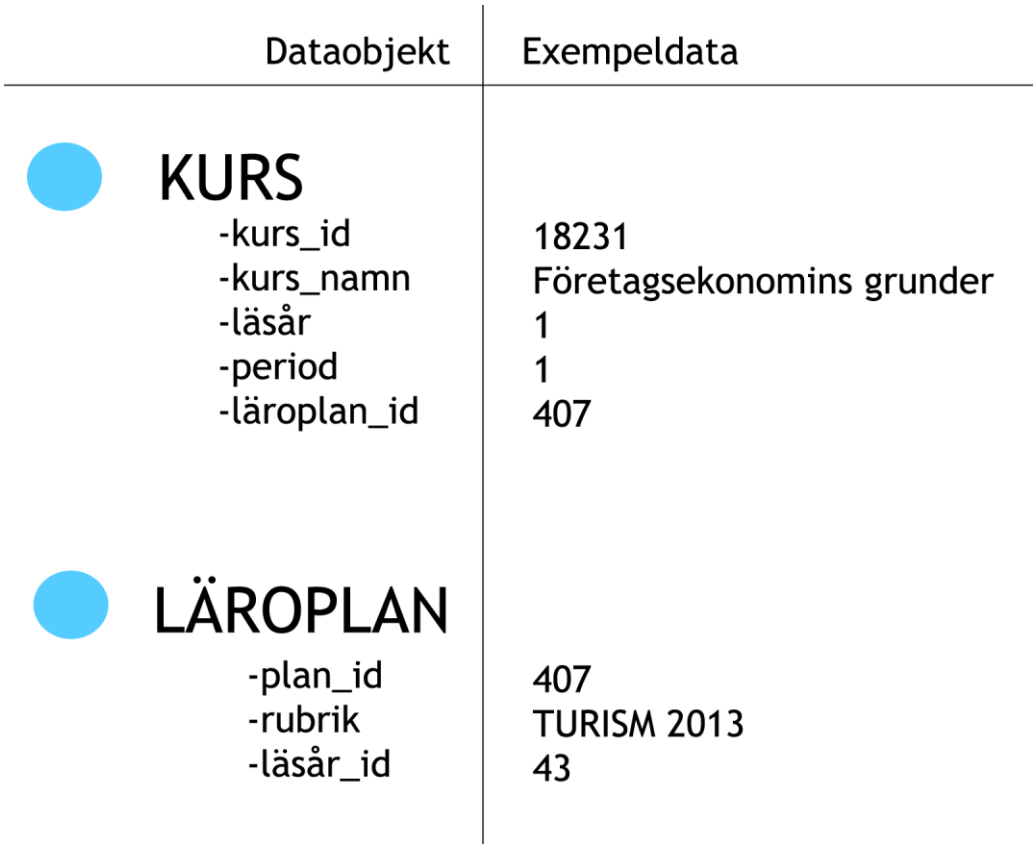
Varje läroplan har som det första blocket <laroplan>. Laroplan -blocket har 11 barn, varav fyra är väsentliga för Resursplaneraren: "plan_id", "rubrik", "lasar_id" och "nivaer". Plan_id -blocket innehåller ett unikt värde, ett tresiffrigt tal. Detta är väsentlig för att särskilja mellan läroplaner, d.v.s. vilken kurs tillhör vilken läroplan. Rubriken ger oss läroplanens namn, i detta fall Turism 2013. Lasar_id innehåller ett unikt värde som berättar vilket år läroplanen startar från. Detta är nödvändig information så att Resursplaneraren kan avgöra vilket år en kurs går i. Nivaer -blocket har ingen information i sig, men dess femte barnbarn (nivaer, niva, innehall, studiehelhet, kurs) är <kurs> blocket, som Resursplaneraren måste ha.

Varje kurs har sitt eget <kurs> -block i läroplanens XML fil, t.ex. enligt följande:

<kurs typ="K" id="18231" kurskod="AB-1-006" grupp="1" sp="5" studiear="1" lasar_id="43" period="1" parent="M5279">Företagsekonomins grunder</kurs>

Denna kurs, "Företagsekonomins grunder", har nio attribut med olika värden för attributen. Resursplaneraren måste ha attributen id, studiear och period. Id-attributet för en kurs är unik, och kan användas för att skilja kurser åt. Med studieår och period värden kan Resursplaneraren avgöra när en kurs går.

Med ovan nämnd information kan Resursplaneraren bygga upp dataobjekt och göra ändringar i dem.



Figur 8: Exempel på data som Resursplaneraren får från en XML läroplan.

Figur 8 visar vilken information är nödvändig för Resursplaneraren. Varje kurs får alltså sin egen unika id, och läroplan_id fältet berättar vilken läroplan kursen hör till. Denna unika plan_id används för att skapa ett samband mellan kurs och läroplan i databasen. M.h.a. läsårets id kan man räkna ut vilket år det är i fråga.

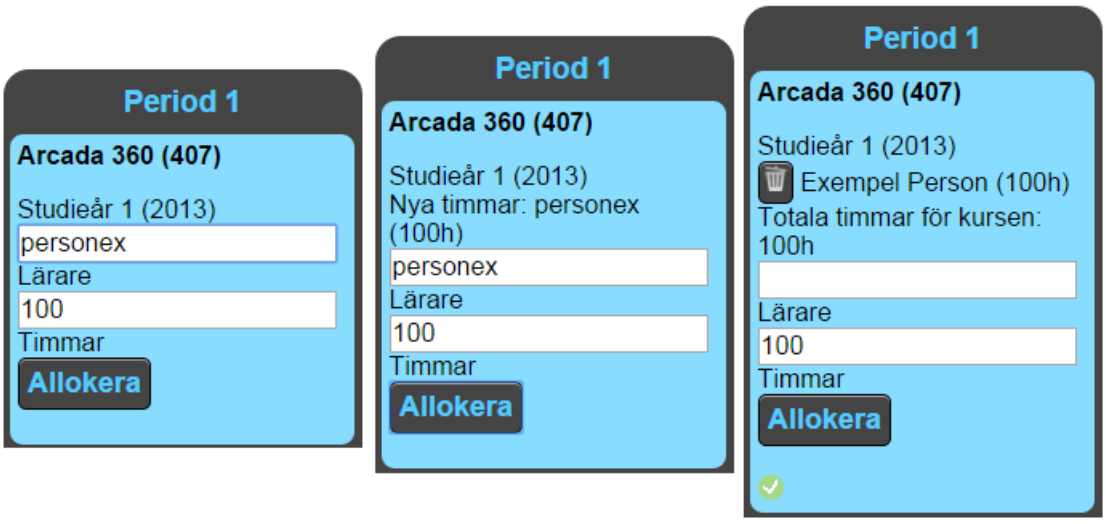
2.9 Databasen

Databasen består av fyra tabeller; kurser, lärare, läroplaner och timmar. Tabellerna lärare och timmar fylls med information direkt från Resursplaneraren, medan läroplaner- och kurser -tabellerna får sin data direkt från en läroplan i XML form. Ifall man har importerat en läroplan till Resursplaneraren finns det en rad i läroplan-tabellen och beroende på läroplanen en varierande mängd rader i kurs-tabellen. Turism 2013 består av 40 kurser, alltså finns det 40 rader i tabellen. Före man lägger till någon personal i tabellen lärare eller allokerar några timmar till en kurs, är både lärar- och kurs-tabellerna tomma.

Varje gång användaren allokerar timmar till en kurs sparas en ny rad i tabellen timmar. Inga allokeringar kan göras utan att man kopplar timmarna till en kurs och en lärare, vilket betyder att användaren måste börja med att lägga till personal.

För att lägga till personal behöver användaren endast fylla i lärarens förnamn och efternamn. Alla andra fält i lärare tabellen sparar applikationen automatiskt då en ny rad skrivs i tabellen. Tabellen består av fälten id, namn, släktnamn och anv. Unika id fylls i genom MySQLs AUTO_INCREMENT funktion, vilket betyder att första raden får id 1, andra raden får id 2 o.s.v. Namnet och släktnamnet fyller användaren i, och användarnamnet (anv) byggs upp på basen av förnamnet och släktnamnet. Sju första bokstäverna från släktnamnet plus första bokstaven i förnamnet blir ett användarnamn. Exempel: Ur "John Rothberg" fås "rothber" plus "j", alltså blir användarnamnet "rothberj". Detta användarnamn används då användaren fyller i allokeringar, och den är lättare och snabbare att skriva in än långa för och efternamn.

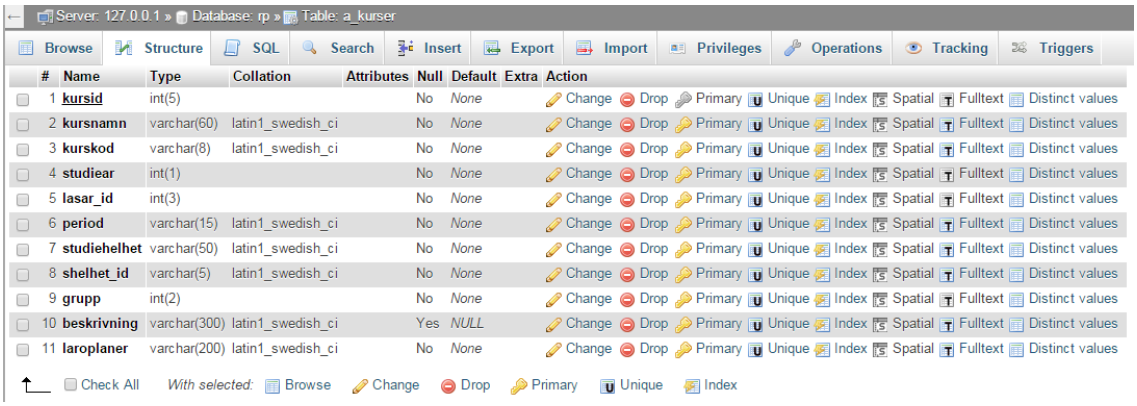
Efter att personalen har sparats i databasen kan man börja allokera timmar till lärare och kurser.



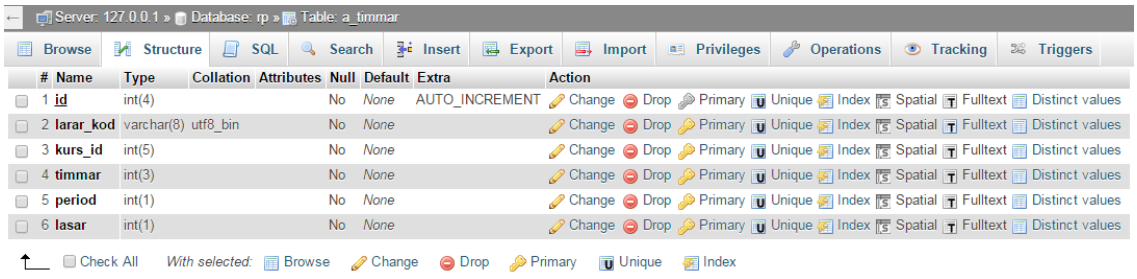
Figur 9: Tre olika tillstånd av en kursdiv vid allokering.

I Figur 9 ser vi ett exempel på en kurs så som den visas i Resursplaneraren, i dess tre olika tillstånd. På vänster före användaren har tryckt på Allokera knappen. I mitten har knappen blivit tryckt på, och JavaScript funktionen allokera() har exekverats. Resultatet visas upp ovanför inmatningsfälten, i detta fall "Nya timmar: personex (100h)". Till höger i figuren ser vi hur denna kurs ser ut efter att sidan har laddats om. Lilla ikonen bredvid namnet är en knapp för radering av allokeringen.

I databasen skrevs i timmar tabellen i detta fall följande information: id: 25, larar_kod: personex, kurs_id: 18225, timmar: 100, period: 1, ar: 2013. Id-fältet ett auto_increment fält. Lärar_koden är vad man skrev in i inmatningsfältet för lärare. Kurs_id fås ur Allokera knappens id, som alltid är samma som id för kursen i fråga. Timmarna fås också från inmatningsfältet som användaren fyllde i. Perioden och året hämtas från Kurser tabellen m.h.a. av kurs_id-numret.



Figur 10: phpMyAdmins strukturvy av tabellen kurser



Figur 11: phpMyAdmins strukturvy av tabellen timmar

3 UTVECKLINGSPROCESSEN

3.1 Funktionella kraven

Uppdragsgivaren ville till en början ha en fullständig webbapplikation, som genast skulle tas i bruk, åtminstone testas. Från början var det också meningen att applikationen också skulle ha stöd till att överföra läroplaner till Arcadas redan existerande system, t.ex. ASTA. Redan tidigt under projektet ändrades kravspecifikationen så, att det skulle utvecklas ett mycket simplare verktyg, som endast hade resurserna i fokus.

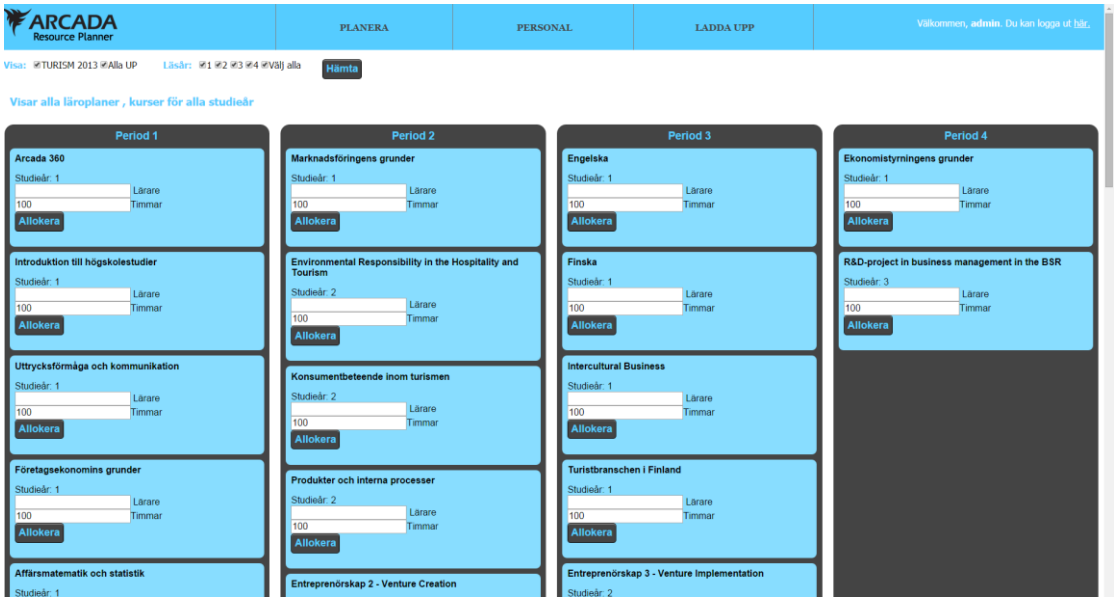
Kraven för Resursplaneraren till att kunna importera en läroplan, spara kursinformationen i en databas och visa upp läroplanen på ett tydligt sätt så, att man kan filtrera vad man kan söka. Vidare förväntades verktyget kunna anknyta resurser (personal) till kurser, och personalens timmar måste man kunna uppfölja genom verktyget.

3.2 Autentisering och sessionshantering

Ingen av Resursplanerarens webbsidor är åtkomlig utan att vara inloggad. Detta var ett direkt krav av beställaren, då Resursplaneraren behandlar personuppgifter. Autentiseringen sköts av PHP så, att det finns en hårdkodad array() som innehåller användarnamn och lösenord som man kan logga in med.
Inloggning och sessionshantering finns i Resursplaneraren, i form av $_SESSION['anvandare'] PHP-variabel, som följer med vem som är inloggad och det visas uppe till höger i <loginarea> -diven. Någon mera tekniskt nyttig sessionshantering implementerades inte, m.a.o. ser man inte från klienten ifall någon annan är inloggad, och ingen kontroll görs ifall samma användare är inloggad flera gånger. Detta leder visserligen till vissa dataintegritetsproblem. Två användare kan vara samtidigt inloggade och manipulera samma data. Detta leder till att den som försöker göra ändringar i databasen sist kommer att stöta på felmeddelanden, då man försöker modifiera information som inte längre finns. Det är lyckligtvis inte någon desto större fallgrop, då Resursplaneraren utvecklades högst som konceptvalidering.

3.3 Utseendet och användargränssnittet

Det som ofta refereras till som "front-end" är den delen av programkoden som i webbutveckling tolkas direkt av webbläsaren, alltså det som användaren ser. Till det hör användargränssnittets händelselyssnande och utseendets programkod. Detta består av HTML-, CSS- och JavaScript-programkod. AJAX-programkod har alltid en del av koden på klientsidan och en del på serversidan. I Resursplanerarens fall är den överväldigande programkoden s.k. "back-end" programkod, som inte syns till slutanvändaren. Back-end programkoden är i Resursplanerarens fall PHP kod som är nödvändig för att få interaktivitet i användargränssnittet och för att dynamiskt kunna filtrera innehållet på webbsidan.



Figur 12: Skärmkapning av användargränssnittet

Såsom Figur 13 ovan visar är färgtemat klart. Ljusblåa färgen med hexadecimala värdet (CSS color egenskapen) #5CF dominerar, och jag har valt den i kombination med en mörkgrå färg #444 som en s.k. "kalla-till-handling" färgkombination, m.a.o. är all väsentlig information uppvisad med dessa färger. Dessa färger valdes för att de ger ett klart och rent uttryck till användaren utan att vara för påträngande eller distraherande. Olika färgtemabotten föreslående webbsajter såsom t.ex. ColorHex.com[i] föreslår ljusgult och en skrikande ljusröd färg som kompletterande färger till #5CF. Då Resursplaneraren är ett arbetsverktyg, inte nöje, och målgruppen är examensansvariga anser jag att klara och kalla färger ger applikationen ett professionellt utseende. Wilbert

O. Galitz, författare och välrespekterad föreläsare i design av användargränssnitt, instämmer med detta påstående. Enligt Galitz (2007, s. 697) är uttryckligen en blå-röd färgkombination ett riskabelt val för verktyg som skall användas länge p.g.a. hur ögat fungerar. Ögat reagerar på olika sätt till färger i mitten av färgspektrumet, såsom gult och grönt, jämfört med färgerna i extremiteterna av spektrumet, blå och röd. Då man ser på en blå-röd bild hamnar ögat anstränga sig för att kunna fokusera på bägge färgerna, vilket i långa loppet leder till trötthet och t.o.m. smärta i ögonen. Det är alltså inte en bra idé att använda dom i ett verktyg som man potentiellt använder applikationen under långa tider i sträck. (Galitz, 2007).

Försök gjordes med att inkludera ljusröda samt ljusgula färger, men de fångade bara oönskat uppmärksamheten. Det skulle underlätta inlärningskurvan genom att använda Resursplaneraren ifall gula eller ljusröda färgerna skulle ha inkluderats, som t.ex. knapparnas eller länkarnas färg, men jag ville att användaren fokuserar på den data som visas upp. Inte behållaren som innehåller data. Hur många timmar en lärare har allokerats i en kurs är kärnan i hela applikationen.

Det anses allmänt som bra praxis att designa användargränssnittet så att målgruppen är bekant med den. Exempelvis, ifall man skulle designa en webbsajt för Coca-Cola, skulle färgtemats ryggrad uppenbarligen väljas som röd-vit.

Figur 13: Skärmkapning av Arcadas hemsida. Delvis samma färgtema syns här som i Resursplaneraren. Källa: www.arcada.fi, hämtad 12.5.2015.

Som vi kan se från Figur 14 ovan är Arcadas hemsida färgmässigt i enlighet med Resursplaneraren. Den ljusröda färgen har också använts för att fånga användarens uppmärksamhet. Det är något som inte behövs i Resursplaneraren och därför inte heller användes.

3.4 Funktionalitet

Resursplaneraren har tre huvudsakliga webbsidor där man kan jobba på planer. Sidan "Ladda Upp" möjliggör uppladdning och syntaxanalys (parsning) av en ny läroplan. På "Personal" -sidan kan användaren lägga till personal till databasen, få en överblick på personalens allokerade timmar, och lärarnas personinformation kan också modifieras

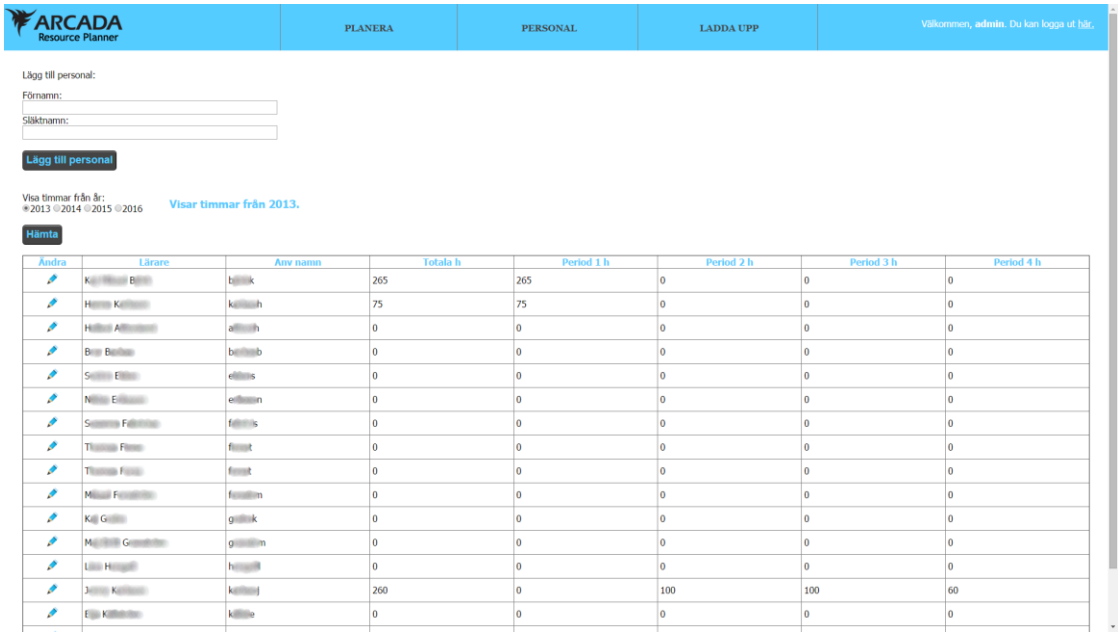
rakt från den sidan. "Planera" är precis som namnet säger; på den sidan planerar användaren resurseringen.

3.4.1 Ladda upp

Webbsidan består av en reguljär FileChooser, som låter användaren bläddra i filsystemet och ladda upp en XML-fil, alltså en läroplan. Serversidans PHP-kod ser till att filändelsen är .xml och att filstorleken inte är för stor, av säkerhetssjäl. Efter att parsningen av XML-filen har lyckats kan läroplanens kurser visas upp på planeringssidan, och resurser kan börja allokeras. Ifall något går fel med parsningen skrivs det ut vad som gick fel, och man stannar på uppladdningssidan.

3.4.2 Personal

På denna sida hämtas personalens information från databasen och så ritas den ut i en tabell, där varje rad i tabellen representerar en lärare och hans/hennes information. Varje rad har också en cell med en knapp som fungerar som en länk till en annan sida; larare.php. Genom att trycka på knappen skickar man den radens (lärarens) information med $_POST-variabler till larare.php, där man kan göra ändringar.



Figur 14: Skärmkapning av Personal -sidan.

Det var viktigt att skilja på planering och överblick. På Personal -sidan kan man endast se allokerade timmar, inte ändra på dem. Det enda man kan ändra på i Personal tabellen är personalens namnuppgifter. Vill man allokera timmar skall det göras på Planera -sidan. Man kan fråga sig, varför behövs Personal -sidan? Jo, därför, att den bjuder på ett annat perspektiv. Denna vy visar allokeringar ur lärarens synvinkel.

3.4.3 Planera

Denna del av funktionaliteten är kärnan till hela applikationen. Det krävde också den längsta tiden att utveckla. Planera -sidan täcker också i sig själv majoriteten av källkoden i hela projektet.

Sidan består av ett filtreringsområde (row2 -div), där man m.h.a. checkboxar kan styra vilken information skall visas upp. Man kan välja att visa kurser från vissa läroplaner eller alla, vissa studieår eller alla.

Under filtreringsområdet ligger innehållsområdet (av CSS-klass "content"). Content -diven delas upp i fyra delar (kolumner) på bredden, ett för varje period i ett läsår. In i kolumnerna placeras kursernas behållare, alltså kursdiv -containers. Varje kursdiv tilldelas en egen id som fås ur en dynamisk SQL-sats vars form beror på vilka checkboxar man har aktiverat. Varje kolumn har alltså sin egen källkod.

PHP har inbyggda funktioner för att jobba med MySQL databaser. Mysqli_connect() är en funktion m.h.a. vilken man gör en anslutning till en databas. Av anslutningen görs ett dataobjekt som lagras i en variabel, t.ex. $connection. Då kan man göra SQL-queries tll databasen man gjorde anslutningen till och därmed få tillbaka information. Exempelvis kan en SQL-sats se ut såhär: *$p1kurser = $connection->query("SELECT * FROM kurser WHERE period = $period");*. Svaret från SQL-satsen sparas då i variabeln $p1kurser. $period är en PHP-variabel vars värde härstammar från kolumnen. Då kan man gå igenom alla rader som hittades i databasen enligt följande:

while($row = $p1kurser->fetch_array()) { ... }

Mellan "{" och "}" placeras all PHP-kod som används för att visa upp data för denna period. Databasens fältinformation kommer man åt med $row['fältnamn']. Via dessa variabler kan man dynamiskt fylla kurs-divarna med information, som demonstrerat i Figur 13 t.ex.

4 RESULTAT

4.1 Målen

Målen för projektet blev mötta, men projektet är försenat. I vissa aspekter överträffar Resursplaneraren de mål som sattes, som t.ex. funktionalitetens omfattning och design aspekterna överträffade kraven drygt.

4.2 Brister

Resursplaneraren är inte robust säkerhetsmässigt. Såsom produkten nu står är den öppen t.ex. till s.k. "SQL-injections". SQL-injections är vad man kallar skadliga SQL-satser som en användare skickar in till systemet via användargränssnittet, t.ex.: Ett registrerings formulär där man skriver in sitt förnamn och efternamn. Ifall man inte skyddar sig mot det, kan en användare registrera sig som "DROP TABLE timmar", vilket skulle radera tabellen. Det är ändå inte så väsentligt då Resursplaneraren inte skall användas såsom den nu står, och ifall någon kommer at testa den är det Arcadas egen personal som kör applikationen lokalt på en dator, i skydd från användare med skadliga SQL-injections i tankarna.

Vidare skulle Resursplaneraren dra stor nytta av att man skulle kunna på planeringssidan söka kurser med lärare som sökord. Det skulle underlätta planeringen i t.ex. sådana fall, där användaren märker på personalsidan att en lärare har alldeles för mycket timmar under ett år eller en period allokerat till sig. Då skulle det vara behändigt att man skulle kunna gå till planeringssidan och visa endast kurser av den läraren.

5 DISKUSSION

Resursplanerarens komplexitet överskred långt det som hade beställts och planerats. Efteråt är det lätt att konstatera att applikationens krav borde ha begränsats mycket mer. Projektet borde ha planerats bättre. Det blev för mycket programmering för ett projekt som i grund och botten skulle uppfylla en väldigt simpel uppgift. För mycket tid investerades på att slipa på utseendet och dynamiska filtreringsfunktioner. Kanske skulle projektets räckvidd hållits bättre i styr ifall jag skulle ha haft tillgång till kalkyltabellerna som uppdragsgivaren hittills har planerat med. Kanske jag då skulle ha bättre lyckats fokusera på uppdragets kärnmoment. Det blir lätt så för en utvecklare; man börjar gå vilse och gör saker onödigt komplicerade.

Med den kritiken vill jag inte alls nedvärdera resultaten i sig. Resultaten var bra. Snarare kritiserar jag min egen arbetsmodell och bristande kommunikation mellan mig och uppdragsgivaren och handledaren.

Det finns delar av källkoden som väl kan användas för vidareutveckling, t.ex. är Resursplanerarens användargränssnitt väl skräddarsytt för Arcadas kursindelning. Källkoden för parsning av en läroplan är också något som bra kan återanvändas eller vidareutvecklas.

Avsikten med arbetet har varit att skapa en prototyp, att bevisa att ett webbaserat planeringsverktyg kan utvecklas utgående från data som redan finns. Slutprodukten är bevis för att det går att göra. Jag är väldigt nöjd med Resursplaneraren, i synnerhet användargränssnittet och dess utseende. Den positiva feed-back jag fick av uppdragsgivaren och handledaren under projektets gång tyder på att jag har orsak att vara nöjd.

KÄLLOR

Connolly, Thomas & Begg, Carolyn. 2005. Database Systems: A Practical Approach to Design, Implementation, and Management. 4th Edition. s157-176. Pearson Education Limited. ISBN-13: 978-0-321-21025-8

Galitz, Wilbert O. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. Third Edition. s691-725. Wiley Publishing Inc. ISBN: 978-0-470-05342-3

Arcadas Wiki, Wikisidan för ASTA [www]. Tillgänglig: https://wiki.arcada.fi/index.php/ASTA. Hämtad 18.5.2015

W3C 2015, XML (Extensible Markup Language) [www]. Tillgänglig: http://www.w3.org/TR/2008/REC-xml-20081126/. Hämtad: 11.5.2015.

Apache Friends 2015, Release Versions [www]. Tillgänglig: https://www.apachefriends.org/community.html . Hämtad 12.5.2015.

PHP.net, Mysqli requirements [www] Tillgänglig: http://php.net/manual/en/mysqli.requirements.php. Hämtad: 12.5.2015

PureCSS.io. [www] Tillgänglig: http://purecss.io/ . Hämtad: 12.5.2015

Apache Software Foundation. About Apache, Overview [www] Tillgänglig: http://www.apache.org/foundation/. Hämtad: 12.5.2015

The PHP Group. Echo Manual. [www] Tillgänglig: http://php.net/manual/en/function.echo.php. Hämtad 12.5.2015

BILAGOR

Bilaga 1: XML-data, en läroplan (Turism 2013)

Följande data är en komprimerad version av en läroplan. För att inte ha nio sidor kod i examensarbetet har jag modifierat XML-koden så att man ser bara en <studiehelhet> och tagit en skärmkapning:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<laroplan>
    <plan_id>407</plan_id>
    <version>15</version>
    <rubrik>TURISM 2013</rubrik>
    <omfattning>210</omfattning>
    <up_nr>401000</up_nr>
    <lasar_id>43</lasar_id>
    <publicerad>2014-08-13 16:33</publicerad>
    <publicerare login="[redacted]" arcadaId="7860">[redacted]</publicerare>
    <asta>1.4.1</asta>
    <nivaer>
        <niva id="20" sp="30">
            <niva_rubrik>Allmänna studier</niva_rubrik>
            <innehall>
                <studiehelhet typ="M" id="5278" sh_id="0" kurskod="" sp="30" parent="N20" optional="0">
                    <sh_rubrik>Den globala ekonomin</sh_rubrik>
                    <kurs typ="K" id="18225" kurskod="AB-1-001" grupp="0" sp="5"
                    studiear="1" lasar_id="43" period="1:0,2:0,3:2.5,4:2.5" parent="M5278">Arcada 360</kurs>
                    <kurs typ="K" id="18226" kurskod="AB-1-002" grupp="0" sp="5"
                    studiear="1" lasar_id="43" period="1:2.5,2:2.5" parent="M5278">Introduktion till högskolestudier</kurs>
                    <kurs typ="K" id="18227" kurskod="AB-1-003" grupp="0" sp="5"
                    studiear="1" lasar_id="43" period="1:2.5,2:2.5" parent="M5278">Uttrycksförmåga och kommunikation</kurs>
                    <kurs typ="K" id="18228" kurskod="SP-1-132" grupp="3" sp="5"
                    studiear="1" lasar_id="43" period="3:2.5,4:2.5" parent="M5278">Engelska</kurs>
                    <kurs typ="K" id="18229" kurskod="SP-1-131" grupp="3" sp="5"
                    studiear="1" lasar_id="43" period="3:2.5,4:2.5" parent="M5278">Finska</kurs>
                    <kurs typ="K" id="18230" kurskod="AB-1-009" grupp="0" sp="5"
                    studiear="1" lasar_id="43" period="3:2.5,4:2.5" parent="M5278">Intercultural Business</kurs>
                </studiehelhet>
            </innehall>
        </niva>
        <niva id="40" sp="90">
            ...
        <niva id="80" sp="30">
            ...
        <niva id="110" sp="30">
            ...
        <niva id="120" sp="30">
            ...
    </nivaer>
</laroplan>
```

Bilaga 2: PHP-kodsnutt för parsandet av en läroplan:

```
<?php

        /*Get courses*/

        if(!is_object($xml->nivaer)) {

        die("Problem!!");

        } else {
```

```
include('dbaccess.php');
$conn = mysqli_connect($dbhost, $dbusername, $dbpassword, $dbname);
$conn->set_charset('utf8');
if($conn->error) {
die("Kunde inte koppla till databasen. Error: ". $conn->error);
}
$kursnamn = array();
$laroplaner = $xml->plan_id; //denna plans id, så att man kan se varje kurs höra till läroplaner
foreach ($xml->nivaer as $nivaer) {
foreach ($nivaer->niva as $niva) {
$niva_rubrik = $niva->niva_rubrik;
foreach ($niva->innehall as $innehall) {
foreach ($innehall->studiehelhet as $studiehelhet) {
$shelhet = $studiehelhet->sh_rubrik;

foreach ($studiehelhet->kurs as $kurs) {
$kursid = $kurs->attributes()->id;
$kursnamn = $kurs;
$kurskod = $kurs->attributes()->kurskod;
$studiear = $kurs->attributes()->studiear;
$lasar_id = $kurs->attributes()->lasar_id;
$period = $kurs->attributes()->period;
                                    //shelhet
$shelhet_id = $kurs->attributes()->parent;
$grupp = $kurs->attributes()->grupp;
```

```
$insertquery = $conn->query("INSERT INTO a_kurser (kursid, kursnamn, kurskod, studiear, lasar_id, period, shelhet_id, grupp, laroplaner, studiehelhet)

VALUES ('$kursid', '$kursnamn', '$kurskod', '$studiear', '$lasar_id', '$period', '$shelhet_id', '$grupp', '$laroplaner', '$shelhet')");



if($conn->error) {

die("Kunde inte sätta raden in i databasen. Error: ". $conn->error);

}

}

}

}

}

}



if(!is_object($xml->laroplan)) {

die('Läroplan är inte ett objekt.<br>');

} else {



$plan_id = $xml->plan_id;

$version = $xml->version;

$rubrik = $xml->rubrik;

$omfattning = $xml->omfattning;

$up_nr = $xml->up_nr;

$lasar_id = $xml->lasar_id;

$publicerad = $xml->publicerad;

$publicerare = $xml->publicerare;

$pub_login = $xml->publicerare->attributes()->login;
```

```
$pub_arcada_id = $xml->publicerare->attributes()->arcadaId;

/*$asta_v = $xml->asta;

$test = $xml->studiehelhet;*/

$insertlaroplan = $conn->query("INSERT INTO a_laroplaner (plan_id, version, rubrik, omfattning, up_nr, lasar_id, publicerad, publicerare, pub_login, pub_arcada_id)

        VALUES ('$plan_id', '$version', '$rubrik', '$omfattning', '$up_nr', '$lasar_id', '$publicerad', '$publicerare', '$pub_login', '$pub_arcada_id')");

    if($conn->error) {

        die("Kunde inte sätta raden in i databasen. Error: ". $conn->error);

    }

echo "Läroplanen parsad. Se på resultatet med knappen nedan.<br>";

echo "<form action='se.php' method='POST'>

        <input type='hidden' name='laroplan' value='$rubrik'>

        <input type='submit' value='Se på planen'>

        </form>";

        }

        $conn->close();

?>
```

[i] För mera information om #5CF och ColorHex, gå till http://www.color-hex.com/color/55ccff