CHALMERS

Modulärt web-baserat resursplaneringsverktyg

Examensarbete inom Högskoleingenjörsprogrammet i datateknik

SIMON LINDHÈN

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

Förord

Jag vill tacka Zetup AB för chansen att få uträtta detta arbete och deras goda mottagande. Jag vill även tacka Uno Holmer, handledare hos Chalmers tekniska högskola, för hans hjälp med rapporten.

Sammanfattning

I rapporten byggs ett resursplaneringsverktyg om. Resursplaneringsverktyget används av företaget Zetup AB och låter anställda ansöka om ledighet. Företaget vill bygga ut systemet med mer funktionalitet men det är svårt med den nuvarande strukturen så systemet behöver byggas om. Det beslutas att ett ramverk skall utvecklas för att garantera en seperation mellan logiken i systemet och utseendet på hemsidan. Det befintliga resursplaneringsverktyget byggs sedan om för att fungera med ramverket. Ramverket gör det möjligt att utveckla moduler till resursplaneringsverktyget vilket förenklar vidareutveckling av systemet. Ramverket tillhandahåller en rad funktioner som förenklar hantering av en databas och formulär. Ramverket är inte specifikt utvecklat för resursplaneringssystemet och kan användas av vilken hemsida som helst.

Abstract

In this report a resource planner is restructured. The resource planner is used by the company Zetup AB and makes it possible for employees to apply for vacation. The company wants to extend the software but the current structure makes it hard so the system needs to be restructured. A decision is made to develop a framework to separate responsibility between the logic and the visual presentation. The existing resource planner is then rebuilt to fit the developed framework. This allows the resource planner to easily be extended and modified using modules. The framework provides a number of functions to ease database management and handling forms. The framework is not specifically developed for the resource planner and can be used by any site.

Tekniska termer och förkortningar

AJAX
Asynchronous JavaScript and XML

Annotation
En markering ovanför en funktion, klass eller variabel som ger ytterligare information om funktionen, klassen eller variabeln.

Bibliotek
En samling klasser och funktioner.

Händelse
Strukturellt koncept. En händelse består av en väckare samt noll eller flera lyssnare. Väckaren väcker en händelse som lyssnarna sedan kan hantera. Det finns olika händelsetyper som används för att väcka händelser vid olika tillfällen.

LDAP
Lightweight Directory Access Protocol – Protokoll för kommunikation med katalogtjänst.

MVC
Annagram för *Model, View, Controller*. Programmeringsmönster med mening att dölja den bakomliggande strukturen (*Model*) och låta användaren styra strukturen med en *Controller* för att få resultatet i en *View*. [3]

MySQL
En relationell databas som använder SQL för kommunikation och förfrågningar.

Namnrymd
I PHP kod: *namespace*. En namnrymd fungerar som paket i Java. Det är en samling klasser och funktioner som kan specificeras unikt med <namnrymd>\<klassnamn|funktionsnamn>.

PHP
PHP: Hypertext Prepocessor – Scriptspråk för webutveckling

Relationell databas
En databas med tabeller. Kopplingar mellan de olika tabellerna kan göras med så kallade relationer.

SQL
Structured Query Language – Språk som används för kommunikation med relationella databaser.

Innehållsförteckning

1 Inledning

1.1 Bakgrund

Zetup är ett företag som ansvarar för sina kunders infrastruktur. Zetup fokuserar på att, istället för att göra enskilda insatser, ta hand om strukturen helt och hållet. På så sätt minskar risken för att något går fel och om något skulle gå fel löses det snabbt.

Sommaren 2013 utvecklades systemet Zetup Resursplanering för att strukturera upp semesteransökningar. Innan systemet utvecklades användes Excel-filer och en stor mängd mailkonversationer vilket ofta blev rörigt. Företaget ser potential i systemet och vill lätt kunna lägga till funktioner, såsom mötesbokning och rapporteringsverktyg.

Då man redan sett behov av att kunna utveckla systemet i framtiden behöver det undersökas hur man kan göra systemet lätt att bygga ut. Därför ska ett system som är uppbyggt av moduler utvecklas.

1.2 Syfte

Syftet med detta exjobb är att undersöka samt implementera lämpliga metoder för att omstrukturera ett existerande PHP-baserat system till att stödja moduler som ökar systemets funktionalitet. Det skall vara möjligt att komma åt data ur systemet från andra system via ett API. Kunskaper inom PHP skall fördjupas och hur ett system smidigast läggs upp skall undersökas och användas.

1.3 Frågeställningar

Frågor som skall besvaras och i förekommande fall implementeras:

- Kan man skapa ett web-system som är lätt att bygga ut med moduler utan att någon konfiguration i kärnsystem behöver ändras?
- Hur kan man utöka en moduls existerande funktionalitet?
- Kan det utvecklade systemet användas av vilken sida som helst, oberoende av den sidans funktionalitet?
- Är ett sådant system skalbart?

1.4 Avgränsningar

Cache är en viktig funktionalitet som drastiskt kan minska tiden det tar för en sida att laddas och arbetet som läggs på servern. Det finns dock inte utrymme inom ramen för examensarbetet att utveckla cache funktionalitet tillgänglig för skalet eller moduler. Den cache som kommer att användas är för att spara databasmodeller då dessa är mycket krävande att generera.

Ett stort antal hemsidor har sin information i flera olika språk för att besökare från olika länder skall kunna läsa innehållet. I ett ramverk som presenteras i rapporten skulle språkhantering önskas. Det finns inte tid att utveckla denna funktionalitet.

2 Metod

Till att börja med skall en fallstudie av det befintliga systemet göras. Här skall undersökas fördelar och nackdelar med det befintliga systemet. En kort beskrivning skall även läggas upp för att förstå hur systemet fungerar.

I utvecklingen av det nya modulära resursplaneringssystemet skall en reflektion göras om hur man skall undvika brister i det gamla systemet samt hur moduler skall hanteras. Den valda metoden skall sedan implementeras. Den bakomliggande och generella logiken skall prioriteras då mestadelen av hemsidans funktionalitet borde kunna tas från det befintliga systemet.

Under tiden systemet utvecklas skall begrepp som behöver förtydligas att beskrivas i *tekniska termer och förkortningar* eller vid behov i *teknisk bakgrund*.

3 Teknisk Bakgrund

Cache

Cache innebär att man sparar beräknad data för framtida bruk. Detta för att slippa göra samma beräkning flera gånger. Cacheteknik utnyttjas i bred utsträckning i allt från upprepande operationer i processorer till statisk data från hemsidor.

På många hemsidor lagras all text som visas för en besökare i en databas, även om texten sällan ändras. Att hämta samma data för varje besökare är onödigt resurskrävande. Istället skapas en cache som informationen sparas i. Informationen kan sedan hämtas ur cachen för varje besökare för att minska belastningen på servern. När sedan innehållet ändras uppdateras cachen så att besökarna alltid ser den aktuella informationen [7].

PHP

PHP är en rekursiv akronym för PHP: Hypertext Preprocessor. PHP är ett script-språk som utvecklades för webutveckling. Eftersom det är ett script-språk finns inga kompilerade filer utan koden läses av under tiden den körs. Detta gör det möjligt att skriva PHP-kod direkt i HTML med följande markeringar: “<?php ?>”. Ett exempel på en liten PHP-fil:

```
<body>
<?php print “Hej”; ?>
</body>
```

Magiska metoder

PHP har ett antal så kallade magiska metoder. Dessa metoder är något som tillåter dold funktionalitet. Ett exempel på en magisk metod är __get(). Med den metoden kan en användare använda variabler i en klass som inte är definierade. Ett exempel på detta demonstreras:

```
class Magic{

    public function __get($varName){
        if($varName == "wand")
            return new Wand();
        else
            return null;
    }

}
```

Figur 3.1 Klass med magisk metod

I figur 3.1 ovanför syns tydligt att ingen variabel existerar. Däremot kan vi simulera en variabel $wand genom att specificera en magisk __get() metod. Detta gör att koden i figur 3.2 nedan kommer fungera och variabeln $wand kommer ha en instans av klassen Wand instansierad. Detta eftersom funktionen __get() kommer kallas med sin parameter deklarerad till ”wand”.

```
$magic = new Magic();
$wand = $magic->wand; //samma som $wand = $magic->__get(“wand”);
```

Figur 3.2 Exempel på användning av magisk metod

Inkludering

Inkludering används när en PHP-fil ska använda sig av en annan PHP fil. Ett exempel är om en klass A i fil A.class.php skall använda en klass B i fil B.class.php. Då måste B.class.php inkluderas längst upp i filen A.class.php. Annars vet inte klass A om klass B.

Om klassen C som ligger i C.class.php sedan inkluderar A.class.php då kommer även C kunna komma åt klass B. Detta sker eftersom inkludering kan ses som att man slår ihop två filer.

AJAX

AJAX (Asynchronous JavaScript and XML) är en metod för att, från en hemsida, ansluta till en server och skicka/hämta data asynkront. AJAX använder flera teknologier för att fungera men den delen som sköter kommunikationen mot en server är JavaScript. Med hjälp av objektet XMLHttpRequest kan JavaScript ansluta till en server och hämta data utan att användaren märker denna kommunikation. JavaScript-koden använder svaret från servern och modifierar den hemsida besökaren befinner sig på för att ge en upplevelse att användaren är konstant uppkopplad mot servern. Tekniken möjliggör en interaktivitet utan att användaren behöver besöka flera hemsidor [9].

AD-System

AD (Active Directory) är ett system utvecklat av Microsoft. Active Directory är ett system som sparar information om nätverksobjekt och användare på en viss domän. En användare innehåller dels information som e-post, telefonnummer och namn men tar även hand om användarspecifika inställningar och applikationer [10].

Externa system kan ansluta till AD-systemet för att hämta information om medlemmar med hjälp av LDAP.

HTML

HTML (HyperText Markup Language) används för uppbyggnad av struktur för en hemsida med hjälp av taggar. Ett HTML-dokument består av fyra huvudsakliga taggar som måste finnas. Dessa är doctype, html, head och body. doctype specificerar att dokumentet är ett HTML-dokument och vissa inställningar om hur en webläsare skall tolka HTML-koden. html-taggen specificerar var själva HTML-koden börjar och slutar. head-taggen specificerar metadata gällande dokumentet. Till exempel vilken titel hemsidan har. body-taggen är innehållet på hemsidan, dvs. det som besökaren ser.

```
<!doctype html>
<html>
    <head>
        <title>Hemsidans titel</title>
    </head>
    <body>
        Detta vissas för en besökare
    </body>
</html>
```

Figur 3.3 Grundläggande HTML-dokument

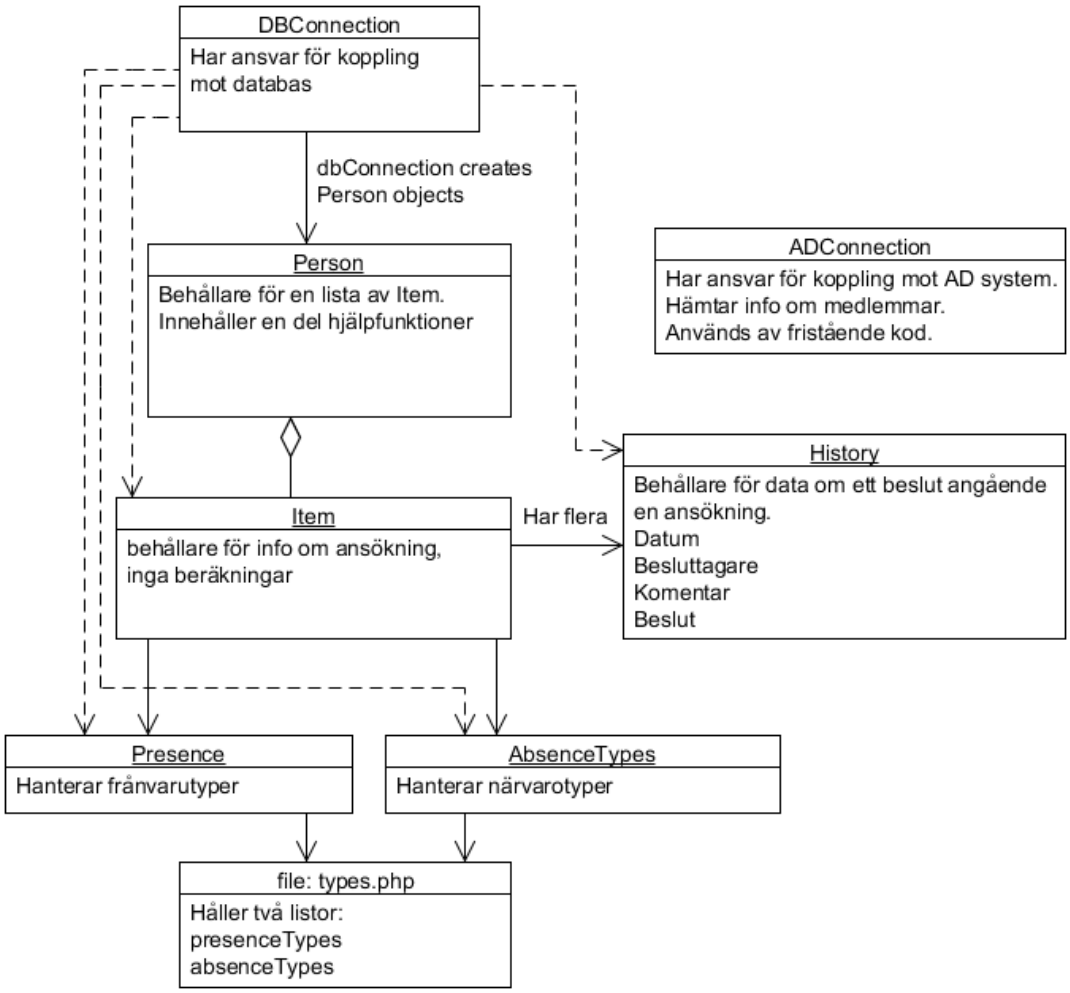
4 Fallstudie av befintligt system

Det existerande resursplaneringsverktyget är strukturellt låst. Systemet utvecklades med en kravspecifikation utan planering för framtida utökningar. Det har redan uppstått några problem vid förändringar. Ett problem uppstod när företaget ville blockera ansökningar under ett tidsintervall och var tvungna att kontakta utvecklaren som inte längre var anställd för att ta reda på vad som behövde ändras.

Företaget använder ett AD-system för att hantera anställdas konton. Resursplaneringsverktyget ansluter till AD-systemet för att hämta information om användare, till exempel vem den anställdes överordnade är.

Det finns två tabeller i databasen: *system* och *history*. *system* lagrar alla ansökningar medan *history* lagrar en ansöknings historik. Historiken är vilka som har godkänt en ansökning och när.

Flera klasser har hög koppling. En av dessa är DBConnection som hanterar databasen. Med den befintliga implementationen behöver DBConnection veta om flera klassers struktur för att kunna lägga in dessa i databasen (figur 4.1).



Figur 4.1 Klassdiagram över befintligt resursplaneringsverktyg

4.1 Händelseflöde

För att demonstrera hur det befintliga systemet fungerar, kommer ett exempel att presenteras. Samtliga formulär behandlas på liknande sätt där tanken bakom flödet är detsamma. Uträkningarna görs av fristående kod då annat inte anges. Med fristående menas att koden inte ägs av någon funktion eller klass.

Ett formulärscenario

Då en person ansöker om ledighet, får denne fylla i relevant information i ett formulär och skicka in det. Användaren dirigeras med formulärdatan till index.php (dvs. huvudsidan) där en rad if-satser kontrollerar om något och vilket formulär som är insänt. Systemet registrerar att en ansökan har skickats in och kontrollerar ett antal grundläggande krav, till exempel att början av ledigheten inte är senare än slutet av ledigheten. Informationen sparas i klassen Item, som läggs in i databasen med klassen DBConnection. Ett mail skickas till den anställdes överordnade med hjälp av klassen ADConnection.

Om den anställde matar in ogiltig information kommer en dialogruta upp med ett felmeddelande efter att användaren har dirigerats vidare från formuläret. Den anställde måste stänga rutan, skapa en ny ansökan och åter igen fylla i sin information.

Fördelar

- Det är en lätt metod att implementera. Det ligger inte så mycket tanke bakom metoden. Det är en metod som fungerar som den ska men inte mycket mer. Det finns inget direkt strukturupplägg eller ansvarsseparation.

Nackdelar

- En nackdel visar sig då man behöver ändra på formulärhanteringen. Eftersom formulärhanteringen ligger på huvudsidan räcker det inte att stänga av formulären vid testning. Vid syntax-fel kommer hela hemsidan att påverkas. Hela systemet behöver stängas ner varje gång formulärhanteringen behöver ändras.
- Klassen DBConnection har kunskap om den interna strukturen i Item för att kunna spara informationen i databasen. Om man behöver ändra klassen Item måste man även ändra DBConnection.
- Felmeddelanden hanteras dåligt vilket kan upplevas som irriterande.

5 Ett modulärt resursplaneringssystem

I systemet kommer programmeringsspråket PHP att användas av flera anledningar. Det är ett språk som är objektorienterat och funktionellt, vilket underlättar struktur och kopplingar av händelser. Det finns dock språk som har bättre objektorienterad struktur, såsom ASP.NET. En av de största fördelarna med PHP är dess portabilitet, vilket är viktigt för ett system som ska fungera på flera maskiner. En annan stor fördel är att ett system skrivet i PHP är lätt att underhålla [4].

För dokumentation används phpDocumentor, eftersom det är ett bra gratis verktyg. Den har även en utförlig dokumentation på vilka dokumentationsannotationer som finns tillgängliga. Metoden som används för detta verktyget liknar det välkända javadoc.

Ett av målen med omstruktureringen är att separera ansvar. För att förenkla detta behövs ett ramverk. Om mestadelen av funktionerna och logiken är separerad från presentationen av hemsidan kan man garantera att kopplingarna blir låga. Om databashanteringen är placerad i ramverket kan man undvika den höga kopplingen som databashanteraren i det befintliga systemet har.

Existerande ramverk analyseras för att se om det som önskas finns tillgängligt. Vid en mindre fördjupning i ramverk som CakePHP och Phalcon hittades inte de funktioner som sökts. Båda tillhandahåller många bra funktioner men stöder varken moduler eller lämnar över ansvaret till skalet.

Vid avsaknad av existerande ramverk med de önskade funktionerna har beslutet tagits att utveckla ett ramverk. Ramverket skall innehålla en samling funktioner som sköter mestadelen av logiken. Ramverket skall inte vara utformat för resursplaneringssystemet utan skall vara oberoende av hemsidans funktionalitet. På så sätt kan man garantera låg koppling och hög kohesion.

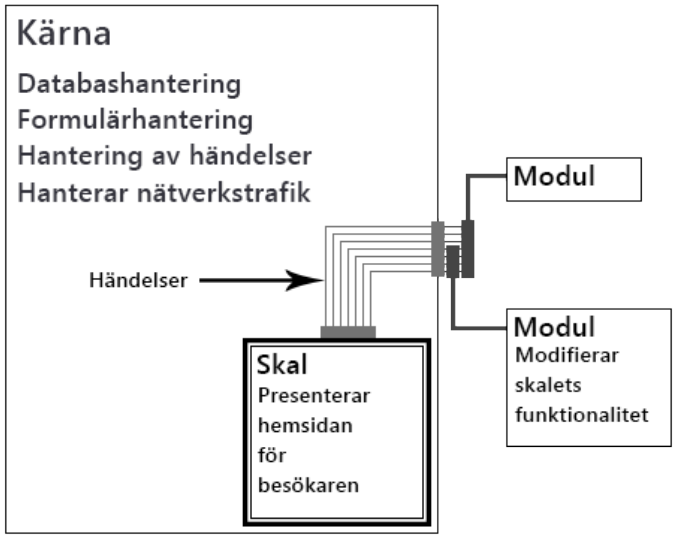
5.1 Kravspecifikation för ramverket

Ramverket skall förse resursplaneringsverktyget med ett antal verktyg som förtydligar ansvar och tillåter modifiering av systemet utan att behöva vetskap om systemets interna metoder. En lista läggs upp om önskade funktioner.

- Automatisk klassladdning – I det nuvarande systemet är det rörigt med inkluderingar av filer. Alla filer inkluderas på förstasidan och är inte beroende av vilka klasser som faktiskt används. Automatisk klassladdning skulle förenkla klasser som använder andra klasser och förbättra strukturen.
- Databashantering - I det nuvarande systemet vet databasanslutningen om intern struktur hos flera objekt i systemet. Databashantering skall därför implementeras.
- Formulärhantering - I det nuvarande systemet hanteras samtliga formulär i samma fil, separerade med if-satser. Detta görs dessutom i samma fil som besökare visar när de besöker hemsidan. Formulärhantering skall implementeras för att förhindra detta.
- Stöd för moduler - Det skall finnas möjlighet att ändra systemets funktion på ett enkelt sätt. Detta skall göras med moduler. Hur moduler behandlas skall undersökas.

5.2 Ramverk

Ramverket består av tre delar: kärna, skal och moduler (se figur 5.1). Kärnan innehåller alla hjälpfunktioner och sköter all nätverkstrafik. Skalet utvecklas till ramverket och presenterar hemsidan som besökaren ser. Moduler bygger på och bygger om skalets funktionalitet.



Figur 5.1 Ramverkets struktur

Filstrukturen är uppdelad i mappar enligt ramverkets tre delar: Core, Mods och Shell (se figur 5.2). I mappen Core är kärnan placerad. I Core är filerna som startar upp systemet placerade samt ett antal bibliotek och hjälpklasser för att förenkla utveckling av en hemsida. I mappen Mods placeras moduler. I mappen Shell placeras skalet.

Core
- Admin – Klasser och resurser som sköter administrationspanelen
- Css – Standard resurser tillgängliga för skalet
- Database – Klasser för hantering av Databaser
- Event – Klasser för hantering av händelser
- Form – Klasser och resurser för formulärhantering
- JavaScript – Standard resurser tillgängliga för skalet
- Login
- config.php
- core.php
- Core_functions.php
- L.class.php
- ModBase.class.php
- Page.class.php
- ShellBase.class.php

Mods

Shell
- Css
- JavaScript
- Pages
- PageLoader.class.php

index.php

.htaccess

Figur 5.2

I PHP används inkludering för att en klass skall kunna använda en annan klass (se teknisk bakgrund). Inkludering kan ofta bli rörigt och samma klass kan lätt inkluderas flera gånger. Därför kommer automatisk klassladdning att implementeras. Automatisk klassladdning gör att inkludering av klasser görs automatiskt. Detta kräver att man har en standard för specificering av namnrymder. Standarden som används i detta system är den vanligaste: Namnrymden är samma som sökvägen till filen. Till exempel har vi klass Page (i fil Page.class.php). Denna har namnrymden Core eftersom klassen ligger i mappen Core. Klassen EventData som ligger i mappen Event har namnrymd Core\Event eftersom mappen Event ligger i Core. Med denna standard definierad kan den automatiska klassladdningen upptäcka en klass som inte är laddad och lista ut var filen ligger med hjälp av namnrymden och klassnamnet. På så sätt kommer aldrig samma fil att inkluderas flera gånger och inkludering av klasser behöver inte göras manuellt.

Efter en fördjupning i PHP hittades funktionen __autoload(). När en klass som refereras i koden inte är inkluderad kommer __autoload() att kallas med klassnamnet som parameter. Med hjälp av detta kan vi inkludera vår klass i __autoload() och skapa den ovan önskade funktionaliteten [12]. Däremot är funktionen ineffektiv och ett tillägg spl_autoload bör användas istället [13]. Vid fördjupning i tillägget hittades den funktionalitet som önskas. Genom att specificera systemets rotmapp samt vilken filändelse de önskade filerna har kan spl_autoload lokalisera och inkludera en fil när en refererad klass saknas [11].

5.2.1 Kärna

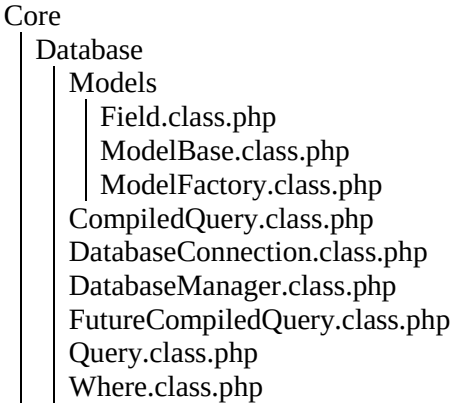
Kärnan är den del av ramverket som tillhandahåller alla hjälpfunktioner. Dessa hjälpfunktioner kan användas av skalet och moduler för att skapa en bra struktur på hemsidan. Nästa steg är att utveckla de önskade funktionerna.

5.2.1.1 Databashantering

En databas är central för många hemsidor. Inte sällan används flertalet tabeller med kopplingar sinsemellan. I detta system görs en översättning mellan objektrelationer och en relationell databas för att slippa skriva stora mängder SQL för hämtning, uppdatering av data och att skapa tabeller.

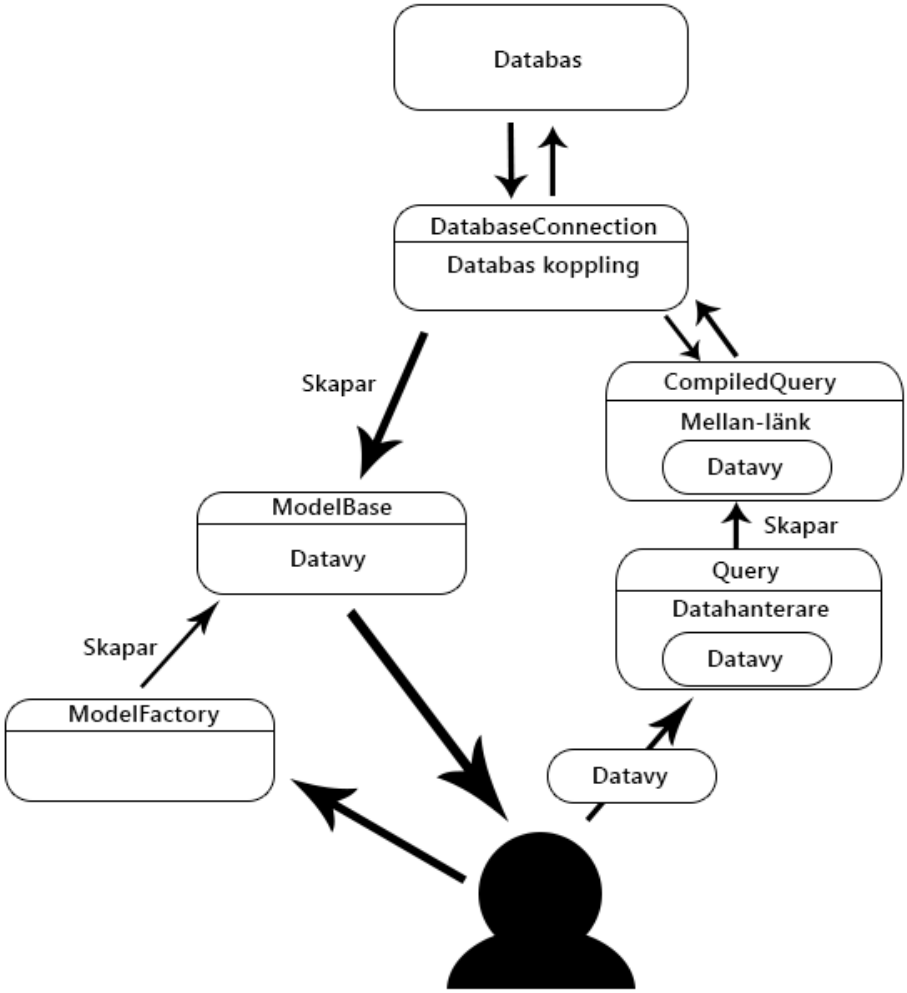
I projektet kommer MySQL att användas som databas eftersom det är ett gratis och välkänt system. Det ingår i de flesta web-server-paket som använder Apache, till exempel wamp och xamp. Det följer vanligen med vid beställning av webhotell [1][2].

Databasbibliotekeket (figur 5.3) innehåller en rad klasser för att man ska kunna använda databasen utan att skriva SQL-kod. Det minskar arbetet som behöver göras när tabeller skall läggas till och senare hanteras.



Figur 5.3 databasbibliotekets filstruktur

För att modellera databasen kommer ett MVC-mönster att användas (se figur 5.4). MVC är ett vanligt mönster använt av flera existerande ramverk till php. Mönstret döljer den bakomliggande strukturen och kopplingen mot databasen [3]. En del förvirring kan skapas när mönstret diskuteras eftersom begreppet ”modell” har en mening i ramverket och en annan mening i MVC-mönstret. För att förenkla detta kallar vi MVC-mönstrets modell för strukturmodell. Strukturmodellen är den del av MVC-mönstret, som är den bakomliggande funktionaliteten, som skall döljas. Strukturmodellen i detta systemet är klasserna CompiledQuery, DatabaseConnection samt själva databasen.



Figur 5.4 Databasbibliotekets struktur

Vyn i systemet är klassen ModelBase. När en förfrågan har gjorts mot databasen kommer svaret att presenteras i en vy. Vyn innehåller en datamodell som representerar en tabell i databasen. Datamodeller definieras i mappar med namnet Models. Mappen Models kan placeras både i skalet och i moduler. En datamodell är en klass med ett antal publika variabler. Varje variabel har en annotation som beskriver hur fältet skall hanteras av databasen.

Controller:n är klassen Query. Det är med den som man kan generera SQL-kod med från en modell. Eftersom Query-objektet behöver en instans av ModelBase kan ModelBase skapas med hjälp av ModelFactory som egentligen inte ingår i MVC:mönstret. På detta sättet kommer man in i den annars slutna cirkeln.

När en datamodell initieras används DatabaseManager. När DatabaseManager skapas sänds en instans av DatabaseConnection in för att DatabaseManager skall ha tillgång till databasen. Sedan kan en lista av datamodeller lämnas över till DatabaseManager för analysering. Eftersom inte PHP har stöd för annotationer, anlyseras kommentarsblocken för att identifiera annotationer. För att få tillgång till kommentarsblocken används den inbyggda PHP-klassen ReflectionClass. En omfattande analysering görs av annotationerna som tagits fram ur kommentarsblocken och två saker utförs:

1. SQL-kod genereras för att skapa tabellerna för modellerna. När tabellerna skapas specificeras inga referenser. Detta för att eliminera risken att något fält refererar en tabell som inte har skapats än. Först när alla tabeller är skapade modifieras tabellerna och alla relationer läggs till.
2. Utöver SQL skapas en struktur av tabellen i en multidimensionell lista (se Appendix A). Listan innehåller information om tabellnamn, vilken klass modellen är skapad från och vilka fält som finns. För varje fält finns information om vilken typ fältet har, dess namn och eventuella referenser till andra tabeller. När en förfrågan görs med Query-klassen används denna lista för att generera SQL-kod för förfrågan.

När en användare skapar en modell med hjälp av ModelFactory returneras en instans av ModelBase. ModelBase har en privat variabel som håller den specificerade modellen. Genom att utnyttja PHP:s magiska __set()- och __get()-metoder kan användaren komma åt variabler som definierats i den ursprungliga datamodellen där värdena är lagrade (figur 5.5, rad 5). Metoden __set(”hairColor”, ”red”) kommer kallas på objektet ModelBase. I metoden undersöks om det är en variabel som finns och ändrar variabeln i så fall till ”red”.

För att förtydliga databasens struktur visas ett exempel (figur 5.5). Modulen Test har tabellen Users. Vid en punkt i koden önskas en uppdatering göras på en person. Jämför exemplet med figur 5.4. Följande kod används:

```
$db = new DatabaseConnection(/.../);
$model = ModelFactory::modelByMod("Test", "Users"); //Skapar ModelBase
$simon = $db->query(Query::select($model)->where
    ($model->getField('name')->equalTo('Simon'))); //Skapar ModelBase
$simon->hairColor = "red";
$db->query(Query::update($simon));
```

Figur 5.5 Exempel på uppdatering med hjäp av databasbiblioteket

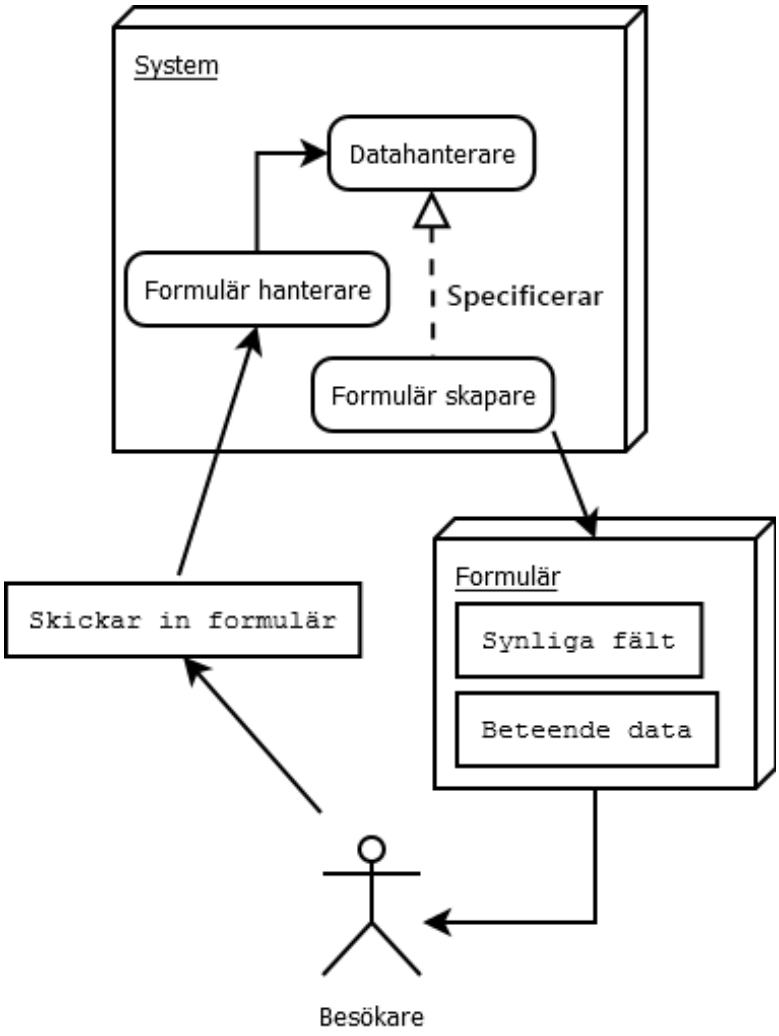
I DatabaseConnection kommer det specificerade Query-objektet att göras om till CompiledQuery som sedan körs.

Förfrågningarna som görs med biblioteket stöder inte MySQL-funktioner såsom SUM och COUNT. Sådana mer komplicerade förfrågningar kan göras direkt med klassen CompiledQuery.

5.2.1.2 Formulär

Formulär är ett av de mest använda verktygen för hemsidor och är det enda sättet för användaren att skicka information till hemsidan. Formulär har ofta en liknande utformning och en hel del kod dupliceras för varje formulär. Därför implementeras ett bibliotek för formulärhantering som underlättar utformning av formulär och datahantering.

När ett formulär ska skapas används biblioteket för att bygga upp rader och kolumner med fält och förklarande texter (formulärskapare, figur 5.6). Vilken klass som ska hantera formuläret och om hanteringen skall skötas med AJAX eller inte specificeras. Formuläret byggs upp med de specificerade synliga fälten samt ett antal dolda fält som förklarar beteendet hos formuläret. På så sätt får även JavaScript-koden vetskap om hur formuläret skall hanteras.



Figur 5.6 Formulärhanteringens struktur

Formuläret presenteras för besökaren som skriver in uppgifter och skickar in formuläret. Om formuläret behandlas över AJAX sänder JavaScript-kod in datan och behandlar svaret. Om det inte sker över AJAX dirigeras användaren till formulärhanteraren, datan behandlas och användaren dirigeras tillbaka till den gamla sidan. När datan har behandlats vidtas korrekt åtgärd utifrån resultatet. Om det uppstod något fel visas det för användaren så denne kan justera uppgifterna.

5.2.2 Skal

För att utveckla en hemsida med ramverket har begreppet skal (engelska shell) tagits fram. Med skalet måste man ha möjlighet att specificera precis hur en hemsida ser ut. Även fast det egentligen är kärnan som producerar koden för hemsidan. För att detta skall vara möjligt specificerar vi en klass ShellBase som innehåller ett antal metoder som förklarar hur hemsidan ska presenteras. Skalet får sedan utöka denna klass för att presentera hemsidan.

Skalet består av en klass PageLoader, som utökar ShellBase, samt ett antal resurser. PageLoader specificerar, beroende på vilken sida klienten befinner sig på, vilken mall som skall användas. En mall är en PHP-fil som inkluderas av kärnan för att presentera en hemsida. Filen skall producera den HTML-kod som kommer att visas för en besökare men ska endast specificera HTML-taggen body. Skalet definierar head-taggen.

När skalet genererar HTML-kod kan händelser väckas för att låta moduler modifiera sidan.

5.2.3 Moduler

Den ursprungliga tanken med moduler var att dessa skulle ersätta vissa klasser i skalet för att byta ut funktionalitet. Som exempel kan vi ta ett skal som ska hantera ledighetsansökningar hos ett företag. Företaget bestämmer sig för att skriva en modul som gör att det inte går att skicka in ansökningar under ett visst tidsintervall. Företaget skriver då en klass som ersätter den existerande funktionaliteten och lägger till en möjlighet att blockera vissa datum.

Efter fördjupning i flera system som är uppbyggda på samma tanke om moduler, såsom Wordpress [5] och Drupal [6], har tanken för upplägget ändrats en del. Istället för klassersättning används händelser. På så sätt kan moduler inte byta ut funktionalitet utan att systemet har specificerat att detta ska vara möjligt.

Det finns flera fördelar med att använda händelser istället för klassersättning. En av de största fördelarna är att man inte behöver skriva om funktionalitet som redan implementerats. Med detta systemet kan man lösa det tidigare presenterade exemplet med en mer dynamisk och lättare lösning. Detta görs genom att väcka en händelse när en ansökning ska skickas in. En modul kan utvecklas för att lyssna på händelsen. När händelsen väcks kan datan analyseras. Om datumen för ansökningen är inom det blockerade intervallet kan modulen förhindra ansökningen och visa ett felmeddelande till användaren.

En annan stor fördel med händelser är att flera moduler som vill hantera samma funktionalitet kan göra det eftersom alla som lyssnar på en händelse kommer att väckas. Om man istället skulle ha klassersättning kommer frågan upp om vilken modul som skall få ersätta klassen.

Händelser

Hantering av händelser görs med klassen Events. Klassen har en samling statiska funktioner för att koppla lyssnare och väcka händelser. En händelse väcks med någon typ av data som väckaren skapar. Lyssnarna får tillgång till ett objekt som innehåller datan som väckaren skickade och funktioner som tillåter lyssnaren att ändra på händelsens funktionalitet.

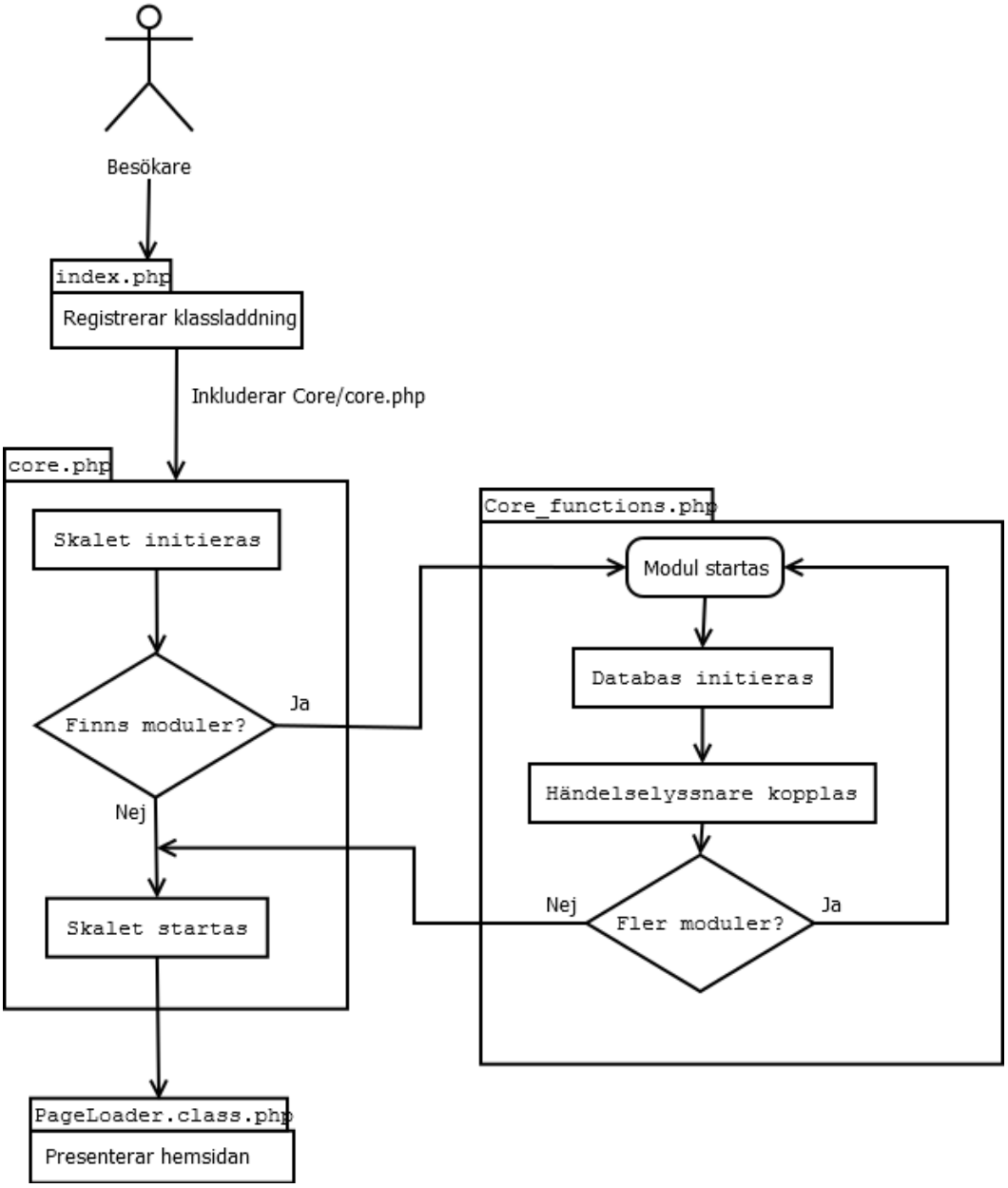
Lyssnarna kan ändra på funktionaliteten hos händelser på två sätt:

- Det första är att förhindra att en händelse propagerar till andra lyssnare. Lyssnarna blir väckta en efter en. Om någon lyssnare önskar att avbryta propageringen av händelsen kommer denna lyssnare att vara den sista som väcks.
- Det andra är att förhindra väckarens slutförande. Denna funktionalitet använder en flagga som kan ignoreras av väckaren men som den bör ta hänsyn till. Om till exempel en ansökning skall skickas in men denna, enligt en modul, inte ligger i det tillåtna tidsintervallet kan modulen markera att åtgärden ej bör slutföras. Väckaren kan ta hänsyn till denna rekommendation, strunta i att spara informationen och visa ett felmeddelande för användaren.

I den statiska Event-klassen sparas lyssnare i en lista. PHP har stöd för att indexera listor med strängar vilket gör det naturligt att ha händelsetyperna som index i listan. För varje händelsetyp finns en lista med alla lyssnare. Detta gör att antalet händelsetyper ej påverkar tiden det tar för en händelse att väckas eftersom strängindexeringen gör att varje index har en adress direktkopplad till sig [se Appendix B].

5.2.4 Uppstart

Filen index.php kommer att öppnas när någon besöker hemsidan. Filen index.php initierar automatisk klassladdning och inkluderar filen som startar upp systemet (Core/core.php) (se figur 5.7).



Figur 5.8 Systemets uppstart

Uppstarten av systemet görs i ett antal steg. Till att börja med startas kärnan som sätter upp regler för klassladdning och startar de övriga delarna i systemet. Det första kärnan gör är att initiera skalet. Det som görs här är att undersöka om användaren har tillåtelse att visa sidan och låta skalet ansluta sig till händelser innan modulerna gör det.

Nästa steg är att starta alla moduler så att de är redo när händelser väcks. Detta görs genom att gå igenom alla mappar i den specificerade “Mods”-mappen. Systemet letar efter de mappar som har filerna som krävs för att sedan initiera dem.

Initieringen består av två delar där det första är databasinitiering. De moduler som förlitar sig på egna databastabeller analyseras och systemet skapar de tabeller modulen kräver. Systemet håller reda på vilka moduler som har fått sina databaser initierade så att detta inte görs varje gång en användare öppnar hemsidan då det skulle skapa fel och ta tid. Eftersom analyseringen av datan är tidskrävande sparas lättåtkomlig information om tabellerna för snabbare hantering i framtiden. Initieringens andra del är då moduler ansluter till systemet. Modulerna kan lyssna på en samling händelser som kärnan och skalet har definierat.

När modulerna har startat kan skalet starta och information om hemsidan börjar skickas till klienten. Under tiden sidan skapas väcker skalet händelser som låter moduler ändra på innehållet. Skalet har ingen kunskap om vilka eller ens om några moduler har modifierat datan.

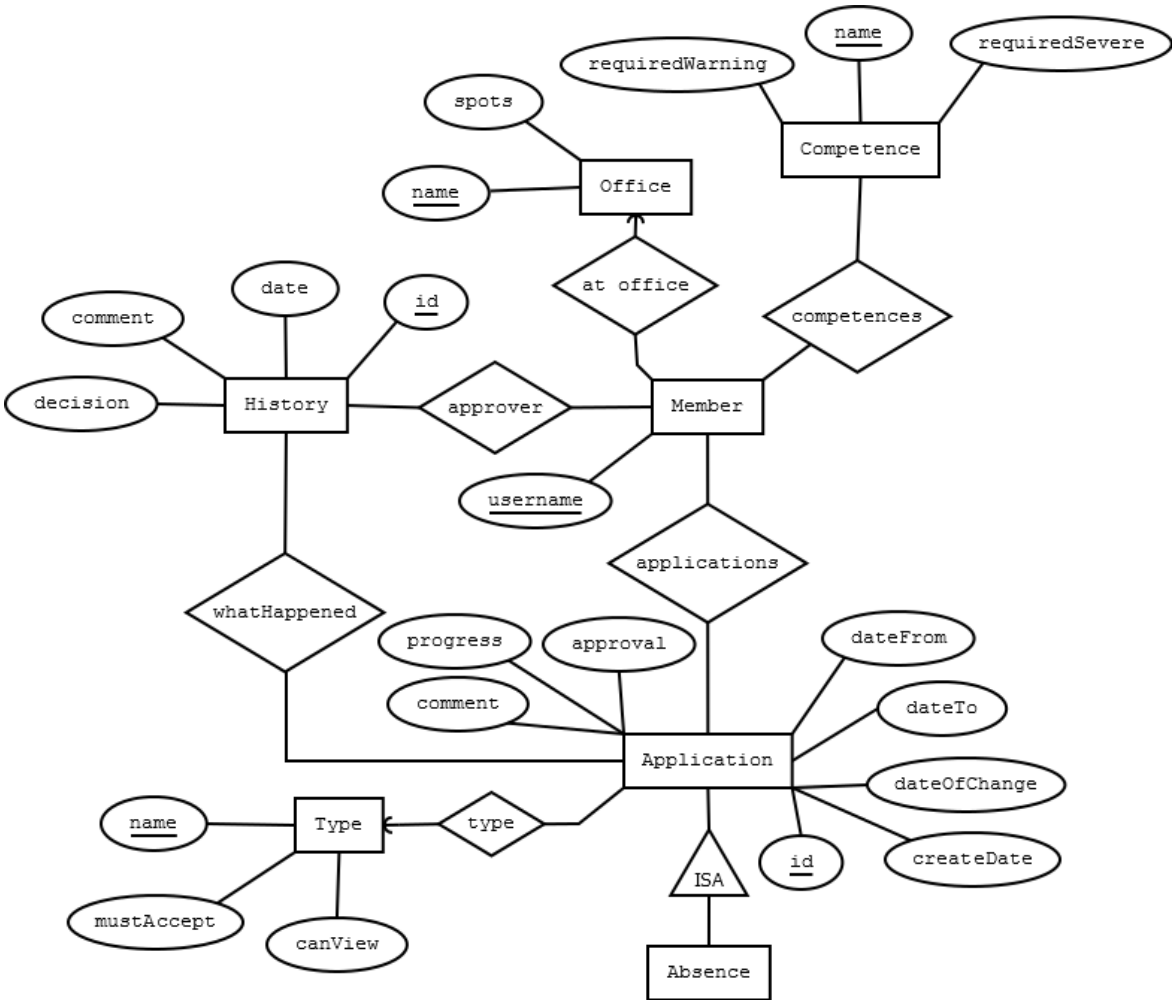
5.3 Resursplaneringsverktyg

Verktyget byggs om från början och den kod som kan återanvändas kopieras efter hand. Resursplaneringsverktyget skall ha samma funktionalitet som det befintliga systemet. Det nya systemet skall ha samma funktionalitet som det befintliga. Systemet skall:

- Låta anställda logga in med sitt AD-konto.
- Låta anställda ansöka om ledighet.
- Skicka ut mail till de berörda personerna vid statusändring hos en ansökan.
- Låta administrationen behandla de inskickade ansökningarna.
- Visa ansökningar i ett schema.

5.3.1 Databas

En ny databasstruktur utvecklas som är mer dynamisk och håller mer information (figur 5.9). Detta för att göra det lättare för moduler att samla information som kan behövas. Med åtanke på att en rapportmodul skall utvecklas skall systemet hålla reda på en anställds kontor och kompetens.



Figur 5.9 Databasstruktur

Strukturen har utbyggnad i åtanke. Det är därför tabellen Absence finns. Tabellen innehåller endast en referens till en ansökan men gör det möjligt att lägga till andra typer av ansökningar. Till exempel om det bestäms att man ska ha möjlighet att byta kontor en vecka eller att man besöker en kund. Då behövs en annan typ av ansökning än just ledighet.

Databasstrukturen byggs upp med datamodeller för att integreras med ramverkets databashantering. En av modellerna presenteras i figur 5.10:

```
class History {

    /**
     * @DBSettings{TYPE:INT;A_I;PRIMARY KEY}
     */
    public $id;

    /**
     * @DBReference{MAPS:Member;FIELD:username;TARGETTYPE:VARCHAR(6);
     *              RELATION:MANY-TO-ONE}
     */
    public $approver;

    /**
     * @DBSettings{TYPE:ENUM('DENIED','APPROVED')}
     */
    public $decision;

    /**
     * @DBSettings{TYPE:TIMESTAMP;DEFAULT:CURRENT_TIMESTAMP}
     */
    public $date;

    /**
     * @DBSettings{TYPE:VARCHAR(1000)}
     */
    public $comment;

}
```

Figur 5.10 Datamodell

5.3.2 Händelser

Vilka händelsetyper som skall deklareras avgör vad som kan modifieras av moduler. Därför är det viktigt att skapa händelsetyper för så många funktioner som möjligt. Först och främst skapas händelser för formulär (se figur 5.12).

Händelsetyp	Egenskaper	Data som skickas med
Formulärhändelser		
applicationPublishing	Händelsen väcks när en ansökning sänds in av en anställd. Om händelsen avbryts av en lyssnare kommer inte ansökningen skickas in och den anställde får ett felmeddelande.	Application-modell
adminAddsForEmployee	Samma som ovan. Väcks när en administratör ansöker för annan personal.	Application-modell
decisionMaking	Händelsen väcks när en överordnad sänder in ett beslut angående en ansökan. Om händelsen avbryts av en lyssnare sparas inte beslutet och ett felmeddelande visas.	Lista med: Application-modell History-modell
editApplication	Händelsen väcks när en anställd modifierar sin ansökning. Om händelsen avbryts av en lyssnare sparas inte ändringarna och ett felmeddelande visas.	Lista med: Application-modell History-modell
Övriga händelser		
createApplicationType	Väcks när ansökningsformuläret skapas. Tillåter lyssnare att lägga till formulär för andra typer av ansökningar en den ursprungliga.	Tom lista med möjlighet att lägga till instanser av Form
createNavButtons	Väcks när navigationsknapparna skapas. Tillåter moduler att lägga till sidor på hemsidan	Tom lista med möjlighet att lägga till en lista med storlek två innehållande den synliga texten samt länkens mål.

Figur 5.12 Händelsetyper

Det skall finnas stöd för att bygga till typer av ledighet. För att detta skall vara möjligt måste dialogrutan för ansökningar kunna ha flera formulär. Dessa skulle kunna ligga som flikar. Händelsetypen createApplicationType skapas.

Moduler kan även behöva lägga till sidor på hemsidan. En händelsetyp createNavButtons skapas.

5.3.3 Användarspecifika data

Viss data som behövs för systemet finns inte i det centraliserade AD-systemet. Dessa uppgifter är vilket kontor en anställd hör till och vilka kompetenser en anställd har. Då en anställd loggar in i systemet första gången kommer en dialogruta visas där den anställde får mata in kontor och kompetenser.

5.3.4 Utveckling av modul för blockering av tidsintervall

En modul utvecklas som blockerar ansökningar under ett visst tidsintervall (se figur 5.13). En mapp Blocker skapas i mappen Mods. En klass Blocker.class.php skapas i mappen Blocker. En lyssnare kopplas till händelsetypen applicationPublishing. Administratörer skall kunna lägga in ansökningar även under det blockerade intervallet. Därför kopplas inte lyssnaren till adminAddsForEmployee.

I funktionen som lyssnar på händelsen kontrollerar vi om ansökningen är inom det angivna intervallet och blockerar i så fall ansökan. Då ansökningen blockeras kan en anledning specificeras som väckaren kan visa för användaren.

```php
<?php

namespace Mods\Blocker;

use \Core\Event\Events;

class Blocker extends \Core\ModBase{

    public function __construct(){
        Events::addListener("applicationPublishing",
                            array($this, "checkDates"));
    }

    public function checkDates($data){
        $dateFrom = $data->getUserData()->dateFrom;
        $dateTo = $data->getUserData()->dateTo;
        $startInterval = mktime(0,0,0,7,1,2014); // första Juli 2014
        $endInterval = mktime(0,0,0,7,31,2014); // sista juli 2014
        if(($dateFrom <= $startInterval && $dateTo >= $endInterval) ||
           ($dateFrom >= $startInterval && $dateFrom < $endInterval) ||
           ($dateTo > $startInterval && $dateTo <= $endInterval))
                $data->preventDefault("Ansökningen får inte vara i Juli");
    }

}

?>
```

Figur 5.13 Modul för blockering med tidsintervall

6 Diskussion

I rapporten utvecklas ett ramverk för att underlätta modularisering av ett resursplaneringsverktyg. Resursplaneringsverktyget utvecklas som ett skal till ramverket (refereras till som kärna). Kärnan ger utrymme för skalet att bygga upp en hemsida på skalets villkor och skalet behöver inte ta hänsyn till kärnan. Eftersom skalet har kontroll över beteendet hos en hemsida läggs mycket ansvar på utvecklaren. För att kunna utnyttja den ökade separation och modularitet som kärnan erbjuder måste skalet utvecklas aktivt för detta ändamål. Om inte skalet definierar några händelsetyper kan inte skalet modifieras av moduler. Man bör fundera över om det ska finnas fler inbyggda händelser i kärnan. På så sätt kan man lyfta bort en del av ansvaret från kärnan och låta moduler modifiera skalet även fast skalet inte definierat några händelsetyper.

Systemet är skalbart. Det finns ingen anledning för att ett stort skal sänker hastigheten. När händelser väcks används PHP:s möjlighet att indexera listor med strängar. Detta gör att antalet händelser inte påverkar tiden det tar för en händelse att väckas då strängarna är kopplade till en adress i minnet. SQL-kod som genereras vid förfrågningar genereras snabbt eftersom den mesta informationen är sparad i datamodellerna som representerar tabellerna i databasen.

Det finns möjlighet att utveckla resursplaneringssystemet men passar troligen bara en viss typ av företag. För de företag som systemet passar, finns en stor möjlighet för förändring med hjälp av moduler. Moduler kan ändra på ansökningsfunktioner, utöka databasen och lägga till nya sidor med nya verktyg.

För att utveckla ett skal behöver utvecklaren läsa på en del om hur ramverket fungerar. Det kan därför diskuteras om PHP var rätt språk att utveckla systemet i. Rasmus, skaparen av PHP, uttalar att mening med PHP var att man skulle slippa skriva klasser för varje hemsida man gjorde för att skapa interaktivitet. Syftet med PHP var att man skulle kunna skapa PHP genom att endast lägga till kod i existerande HTML-dokument[8]. I det utvecklade ramverket tas ett steg bakåt ur denna synvinkel. Däremot kan det vara svårt att skapa en bra struktur genom att placera den mesta koden i HTML-dokument. Istället skulle Java EE kunna användas. Java EE använder en del tankesätt som använts i projektet.

6.1 Vidareutveckling

Ramverket borde kunna hantera både språk, cache och ett API. Detta är viktiga koncept hos en modern hemsida. För att en oinsatt person skall kunna utveckla till systemet bör en manual utvecklas om hur man gör ett skal och moduler.

7 Slutsats

För att möjliggöra den önskade modulariteten hos resursplaneringssystemet utvecklades ett ramverk. Ramverket gör det möjligt att utveckla vilken hemsida som helst, om hemsidan utvecklas som ett skal till ramverket. Efter att en del initieringar har gjorts av ramverket har skalet full kontroll över vad som händer. Ramverket förser hemsidan som utvecklas med en rad funktioner för att lättare separera ansvar i systemet. Ramverket hanterar händelser och moduler som tillåter hemsidan att stödja moduler genom att väcka händelser. De funktioner som inte ramverket tillhandahåller kan utvecklas i skalet.

En hemsida som utvecklas till ramverket refereras till som skal. Ramverket refereras till som kärna. Skalet i denna rapport är ett modulärt resursplaneringssystem. En grundläggande funktionalitet finns implementerad men denna kan utökas och modifieras. Den grundläggande funktionaliteten låter anställda ansöka om ledighet som sedan hanteras av överordnade.

Moduler kan utvecklas för att hantera de händelser skalet väcker. På så sätt kan moduler modifiera funktionalitet hos skalet. Det krävs av den som utvecklar skalet att definiera händelsetyper för att väcka händelser vid lämpligt tillfälle.

I syftet anges önskan om API. Detta har inte utvecklats eftersom det inte fanns utrymme inom ramen för examensarbetet.

Efter fördjupning i språket PHP har automatisk klassladdning implementerats. Den automatiska klassladdningen eliminerar behovet av att manuellt inkludera PHP-filer.

8 Referenser

[1] Beställ webhotell och domännamn. *Binero.* https://www.binero.se/bestall. (2014-06-15)

[2] Produkt. *One* http://www.one.com/sv/produkt. (2014-06-15)

[3] Pitt, C (2012) Pro PHP MVC. Apress: New York.

[4] Amadin, I F; Nwelih, E. (Oct 2010) An Empirical Comparison Of: HTML, PHP, COLDFUSION, PERL, ASP.NET, JAVASCRIPT, VBSCRIPT, PYTON AND JSP. Global Journal of Computer Science and Technology, vol. 10, Issue 12, ss. 9-17.

[5] WordPress Codex. http://codex.wordpress.org/Plugin_API (2014-05-12)

[6] Drupal API. https://api.drupal.org/api/drupal/includes!module.inc/group/hooks/7 (2014-05-12)

[7] Hansson, R (2000) Cache – teknisk förklaring. Svenska datatermgruppen. http://www.datatermgruppen.se/index.php?option=com_content&view=article&id=89&Itemid=91&obj=A2&uttr=cache (2014-05-14)

[8] Lerdorf, R (2007-04-26). "PHP on Hormones – history of PHP presentation by Rasmus Lerdorf given at the MySQL Conference in Santa Clara, California". *The Conversations Network.* http://web.archive.org/web/20130729204354id_/http://itc.conversationsnetwork.org/shows/detail3298.html (2014-05-16).

[9] Garret, J (2005-02-18) Ajax: A New Approach to Web Applications. *Adaptive Path.* https://web.archive.org/web/20080702075113/http://www.adaptivepath.com/ideas/essays/archives/000385.php (2014-05-23)

[10] Active Directory Architecture. *Microsoft TechNet*. http://technet.microsoft.com/en-us/library/bb727030.aspx (2014-05-23)

[11] http://www.php.net/manual/en/function.spl-autoload.php (2014-06-12)

[12] PHP Manual: __autoload. *PHP: Hypertext Preprocessor.* http://www.php.net/manual/en/function.autoload.php. (2014-06-15)

[13] PHP Manual: Autoloading classes. *PHP: Hypertext Preprocessor.* http://www.php.net/manual/en/language.oop5.autoload.php. (2014-06-15)

Appendix A: Datamodell

```
$modelArray=array(
    "TableName" => "serri_DatabaseS",
    "ClassSource" => "Shell\Models\DatabaseS",
    "PrimaryKey" => "id",
    "Fields" => array(
        "id" => array(
            "name" => "id",
            "type" => "i",
            "mysqlType" => "INT"),
        "t2" => array(
            "name" => "t2",
            "type" => "i",
            "references" => array(
                "table" => "TEST_DATABASET2",
                "field" => "ID",
                "relation" => "MANY-TO-ONE")
            )
        )
    );
```

Appendix B: List-statistik

Tid i mikrosekunder för att hämta ut ett element mitt i en lista med olika stora listor:

Test #	Lista med 2 element	Lista med 500000 element
Test 0	6.9	2.1
Test 1	1.9	1
Test 2	0.9	1.9
Test 3	1	1
Test 4	1	1
Test 5	1	1
Test 6	1.2	1.9
Test 7	1	1
Test 8	1.9	1
Test 9	1.1	1
Test 10	2.1	1
Test 11	1.9	2.1
Test 12	1	1
Test 13	1.9	1.9
Test 14	1	2.1