Webbutveckling

Produktion av ett anmälningssystem

Thai Nguyen

Examensarbete för ingenjörsexamen (YH)

Utbildningsprogrammet för informationsteknik

Vasa 2016

Examensarbete

Författare: Thai Nguyen

Utbildningsprogram: Informationsteknik

Handledare: Susanne Österholm

Titel: *Webbutveckling – Produktion av ett anmälningssystem*

Datum: 16.05.2016 Sidantal: 47 Bilagor: 1

Abstrakt

Detta arbete utfördes på begäran av Vasa Veteranbilssällskap. Uppgiften var att skapa ett webbaserat anmälningsformulär till evenemanget Vaasan Ajo 2015. Föreningen ville ha möjligheten att visa formuläret och tillhörande information på de båda inhemska språken, samt ladda ned deltagaruppgifter till ett externt Excel-dokument.

I detta examensarbete beskrivs dialog med uppdragsgivaren, system- och behovsanalys och det praktiska utförandet för att bygga ett användbart anmälningssystem med hänsyn till dagens webbtekniker.

Språk: svenska Nyckelord: webbutveckling, Bootstrap, jQuery, HTML, PHP, CSS

BACHELOR'S THESIS

Author: Thai Nguyen

Degree Programme: Information Technology

Supervisor: Susanne Österholm

Title: *Web Development – Production of Application System*

Date: 16.05.2016 Number of pages: 47 Appendices: 1

Summary

The thesis was carried out as part of a request from the Vasa Veteranbilssällskap association. The task was to create a web application form for the ”Vaasan Ajo 2015” event. The association wanted the possibility to view the form in Finnish and Swedish, as well as the ability to download all incoming applications in Excel format.

This thesis describes how the form's structure was planned based on communications with the client, and how the programming was done with respect to today's web technologies.

Language: Swedish Keywords: Web development, Bootstrap, jQuery, HTML, PHP, CSS

Innehållsförteckning

Bilagor

Figurförteckning

Figur 29. Aktivitetsdiagram över administrationspanelens session.

Ordförklaringar

AJAX	Asynchronous JavaScript and XML är en teknik för webbutveckling. Den kan skicka data från klienten till servern utan att sidan behöver laddas om i sin helhet.
Klient	Den person som använder en programvara eller webbapplikation som kommunicerar med en server.
Server	En dator som tillhandahåller tjänster, som kan utnyttjas av, exempelvis klienten eller andra servrar.
Webbhotell	En tjänst som tillhandahåller diskutrymme som kunden kan hyra för att lägga upp webbapplikationer och hemsidor.
App	Uttryck för applikationer, dvs programvaror.
Hybridapp	Mobila programvaror som byggts med webbteknik, t.ex. HTML5, CSS och JavaScript. Appar som byggts med verktyg anpassade för mobiltelefonernas operativsystem kallas däremot "Native Apps".
SEO	Står för sökmotoroptimering, vars målsättning är att genom åtgärder få en webbsida att komma högre upp i sökresultaten hos sökmotorer. Används i marknadsföringssyfte.
Hacker	En person, vars mål är att olovligen få tillgång till data eller funktionalitet som tillhandahålls av en tjänst med hjälp av t.ex. programmering och nätverksteknik.
UML	Står för Unified Modelling Language och är ett sätt att uttrycka planer på hur ett system är tänkt att fungera, t.ex. med grafiska diagram.
CSV	Comma Separated Values är ett filformat som använder skiljetecken för att separera data på varje rad. Formatet kan läsas av Microsoft Excel, OpenOffice Calc och LiberOffice Calc där datan visas i rader och kolumner.
OpenSSL	Kodbibliotek som innehåller funktioner för säker datatransport och datakryptering.

SSL	Secure Sockets Layer är en teknik för kryptering av datatrafik som skickas över ett nätverk.
Responsiv webbutveckling	En princip för att få en webbapplikation att anpassa sitt grafiska utseende och sina funktioner till alla enheter där den används.
CMS	En förkortning av Content Management System, och är ett system som tillåter användaren att hantera innehållet i en webbapplikation.
Front End	Syftar på programmering av användargränssnittet på en webbsida. Tekniker som oftast förekommer tillsammans med detta uttryck är bl.a. HTML, CSS och JavaScript.
Back End	Syftar på arbete på serversidan. Det kan röra sig om datahantering och programmering av underliggande funktionalitet som driver en webbsida (t.ex. hämtning av data från en databas).
HTTP	Hypertext Transfer Protocol är ett protokoll som används i t.ex. en webbläsare för att överföra information via internet. T.ex. en webbsida anropas och laddas in i webbläsaren via detta protokoll.
Administrationspanel	Grafiskt användargränssnitt varifrån man kan använda administrativa funktioner för att styra eller ta ut data från ett system.
XOR	Logikoperator som används inom matematik och binär datahantering. 0 XOR 1 = 1. I andra fall är den 0.
Hash	Funktion som utför envägskryptering och vars resultat inte är menat att återskapas. Resultatet kallas ”hash”-värde, eller ”hash value” på engelska.
Hashtabell	”Ordböcker” som innehåller hashar och deras motsvarigheter i okrypterad form.
AES	Advanced Encryption Standard är en krypteringsspecifikation och samtidigt en algoritm som slagits fast av NIST. Standarden baserar sig på Rijndael som utvecklades av Joan Daemen och Vincent Rijmen.
Salt	Slumpmässigt valt tilläggsvärde som krypteras tillsammans med egenskriven information, t.ex. lösenord, för att göra det svårare för hackare att komma åt ursprungsinformationen.

1 Uppdragsgivare och uppgift

Vasa Veteranbilssällskap är en liten förening beläget i Vasa, som riktar in sin verksamhet på visning och historia kring 1900-talets bilmodeller. Föreningen beslöt år 2015 att arrangera nöjesevenemanget Vaasan Ajo 2015, som fokuserade på underhållning och tävlingar med temat ”veteranfordon”.

För detta ändamål efterfrågade föreningen ett webbformulär som kunde ta emot och lagra anmälningar från tävlingsdeltagare via Internet. Gustaf Sten, styrelsemedlem i föreningen, fungerade som uppdragsgivare.

Evenemanget ägde rum i Sandviken strax utanför Vasa Stads centrum under juni 2015.

1.1 Syfte

Tanken med det här examensarbetet var att visa hur utvecklingen av ett litet system skulle kunna utföras på ett effektivt sätt och samtidigt uppfylla uppdragsgivarens önskemål. Ur ett helhetsperspektiv skulle krav- och behovsaspekterna granskas, innan verktyg och arbetssätt kunde väljas.

Målet var att slutresultatet skulle vara en säker produkt som är användbart och enkelt för både uppdragsgivaren och personer som ville delta i evenemanget.

1.2 Dialog med uppdragsgivaren

Uppdragsgivaren framförde att föreningen ville ha ett anmälningsformulär som motsvarade föreningens utskrivna pappersblankett (se bilaga 1). Deltagarna skulle ha möjligheten att ange sina personuppgifter, fordonsmodell och välja tilläggstjänster. Dessa uppgifter skulle sedan lagras i en databas för att senare hämtas av arrangörerna i Excel-format.

Utöver formuläret skulle det finnas en sida som visade information om Vaasan Ajo 2015, tjänsterna och aktiviteterna som skulle erbjudas under tiden som evenemanget pågick. Både formuläret och informationen skulle vara tillgängliga på finska och svenska.

1.2.1 Utredning över tänkbara behov

Eftersom man endast beskrev vad man ville ha, så fanns det frågor som behövde svar innan planeringen kunde börjas.

Under utredningen uppgavs det att evenemanget skulle vara öppet för alla. Emellertid skulle alla som anmälde sig till tävlingarna vara 18 år eller äldre, och äga ett fordon som var minst 30 år gammalt. Minderåriga kunde vara med som passagerare.

Uppdragsgivaren uppskattade att antalet deltagare inte skulle överstiga ett par hundra stycken, eftersom evenemanget också var en del av den inhemska storföreningen Suomen Ajoneuvohistoriallinen Liitto - och därmed byter arrangör och ort varje år. Huruvida de andra arrangörerna skulle återanvända denna applikation diskuterades inte.

Uppdragsgivaren bedömde också att det var osannolikt att föreningen skulle arrangera ett liknande evenemang under de kommande åren – och uppgav att man själv skulle sköta marknadsföringen, samt betalningskontrollen.

1.2.2 Plan för lansering och underhåll

Föreningen hade en hemsida sedan tidigare, men tyvärr gick det varken att modifiera dess källkod eller att ladda upp egna kodfiler, eftersom hemsidan skapats med gratistjänsten Weebly som inte tillåter egna modifikationer på servernivå.

Ett tillfälligt avtal om domännamn och diskutrymme på ett webbhotell skulle användas, och dessa skulle vara tillgängliga tills anmälningstiden gått ut och efter att alla uppgifter hade laddats ned av uppdragsgivaren.

2 Analys

Enligt vad som framkommit i kapitel 1 så bedömdes projektet inte vara komplicerat, främst för att man uppskattade att antalet deltagare inte skulle överstiga några hundra stycken, men även för att externa betalhanteringstjänster som t.ex. Klarna och PayPal inte behövdes implementeras, eftersom föreningen valt att kontrollera de inskickade uppgifterna manuellt.

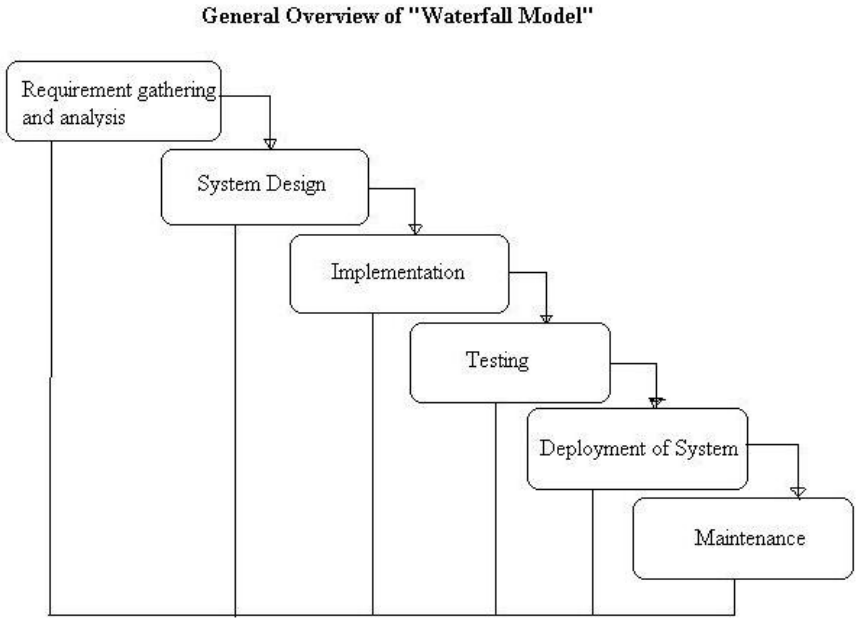
Ur arbetsperspektiv så rörde det sig inte längre om att bygga ett formulär, utan snarare ett mindre system, som skulle ha en publik del och en administrationspanel. Den publika delen skulle bestå av anmälningsformuläret och informationssidan, som skulle kunna visas på finska och svenska. Administrationspanelens viktigaste funktion var nedladdningen av Excel-dokumentet.

På grund av att sannolikheten att man skulle ordna nya evenemang av liknande slag under den närmaste tiden var liten, så drogs slutsatsen att systemet skulle vara statiskt.

2.1 Arbetsprocess: Vattenfallsmodellen och agil tillämpning

”Vattenfallsmodellen” användes som utgångspunkt för hela arbetsprocessen. Modellen innebär att man delar upp arbetet i delmoment enligt diagrammet i figur 1, där man börjar med att analysera alla krav och samlar information.

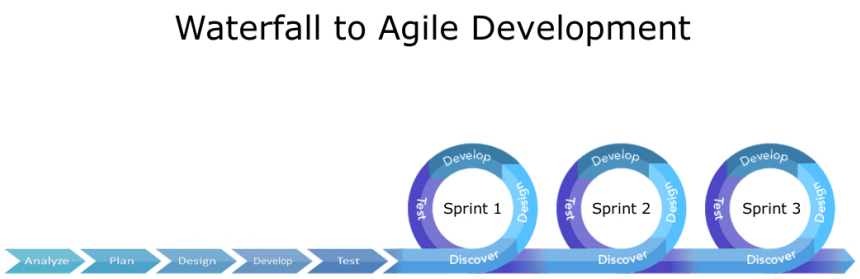
Nyckeln i denna modell är att man arbetar färdigt med ett delmoment, innan man går över till nästa. Tanken är att man undviker att blanda ihop material från ett moment i ett annat, och på så sätt kan fokusera på att lättare gå framåt.



Figur 1. Grafiskt diagram av Vattenfallsmodellen. [1]

Vattenfallsmodellen är lämplig för mindre projekt, om man har analyserat och förstår exakt vilka krav som ställts på basen av samlad information. Extra viktigt är emellertid att man har en klar bild av hur resultatet ska se ut och hur länge arbetet kan tänkas pågå för varje delmoment. Om arbetet bedöms som komplext, och om man arbetar i grupp där resursfördelningen riskeras variera, så är vattenfallsmodellen kanske fel väg att gå.[1]

Tyvärr vore det i detta fall alldeles för riskfyllt och kanske lite arrogant att endast förlita sig rakryggat på vattenfallsmodellen, på grund av att den är så framåtgående och inte heller tar hänsyn till kommunikation och samarbete *under* processens gång. Det här leder till att tidsåtgången per moment kan variera bortom de planerade tidsgränserna.



Figur 2. Den agila programmeringsmetodiken tillämpad på vattenfallsmodellen.[2]

Lösningen är att tillämpa en mera iterativ modell, den agila modellen som tillåter felkorrigeringar i de föregående momenten samtidigt som man kontinuerligt kan ta hänsyn till frågor, förslag och efterfrågan från uppdragsgivaren. Det här är viktigt eftersom föreningen förväntar sig att allting fungerar och att man slutligen får in en inkomst från evenemanget.

En tillämpning av den agila modellen på vattenfallsmodellen är ett bra val för det här arbetet. Då undviker man lättare eventuellt slarv, samtidigt som man kan hålla sig fokuserad och ändå gå framåt samtidigt som man har tid till att ta hänsyn till eventuella överraskningar, samt kan samarbeta med uppdragsgivaren.[3]

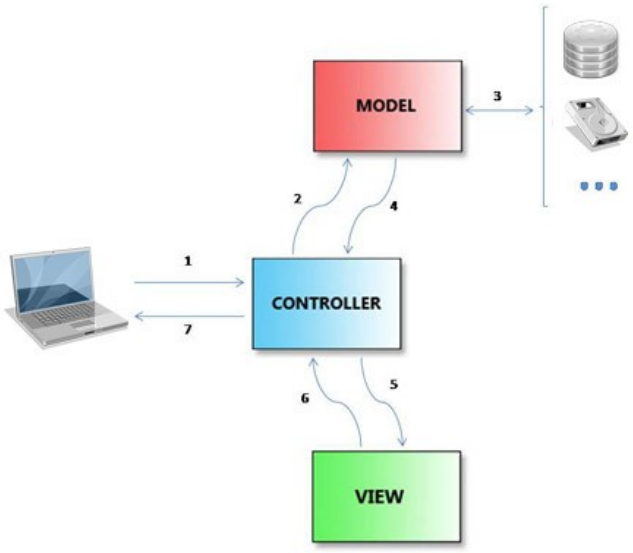
En beskrivning av den agila modellen kan återges i 12 punkter[4], varav de viktigaste i sammanhanget är:

- Högsta prioriteten är att kunden är nöjd.
- Att vara öppen för krav och förslag, även i ett sent skede av utvecklingsprocessen.
- Att det mest effektiva sättet att kommunicera är ansikte mot ansikte.
- Att fungerande mjukvara är det första som beaktas när framsteg mäts.

2.2 Programmeringsmönstret Model View Controller

Programmeringsmönstret Model View Controller, eller MVC, är lämpligt för mindre projekt eftersom den ger klar översyn av alla filer och deras roller.

Om man utgår ifrån analysen i kapitel 2, så kan man konstatera att det rör sig om ett system för insättning i en databas, och nedladdning av samma data – ett litet system där data kan gå två vägar baserat på vad som bestämts i systemet. Se figur 3 för exempel.



Figur 3. Model View Controller-principen i bild.[5]

Den största fördelen med MVC är att programmeringen blir mera modulär. Om ändringar behöver göras för ett delmoment i systemet, så vet man var man skall ändra utan att påverka resten av funktionaliteten.[5]

2.2.1 Model

Modellen hanterar logiken och strukturen bakom ett system, och beskriver hur den skall tolkas. Ett enkelt exempel på hur en modell kan se ut, visas i kodexempel 1.

Kodexempel 1. Modell av ett anmälningsformulär.

```
<?php
class application_form {
        public $name;
        public $birthday;
        public $db;
        public function save( $name , $birthday ){
                $this->name = $name;
                $this->birthday = $birthday; }
        public function submit() {
                $this->db->query("...");
        }
}
?>
```

I kodexempel 1 visas en formulärmodell. Funktion save() lagrar formulärdata i klassens medlemsvariabler och funktionen submit() lagrar datat i databasen, om dessa anropas av en controller.

Att ha en formulärklass är användbart om man skapar flera sidor som innehåller samma formulär. Man behöver inte skriva logiken varje gång man använder den, eftersom man kan anropa den från de sidor som använder formuläret.

En annan egenskap hos modellen är att den används för att kommunicera med externa tjänster (t.ex. databaser, hårdskivor, molntjänster etc.).

2.2.2 Controller

En controller styr vad som händer i ett system. Controllern utför ett kommando som ges av användaren eller av modellen. I kodexempel 2 visar hur controllern använder modellklassen application_form från kodexempel 1.

Kodexempel 2. En controller som använder modellen application_form.

```
<?php
	$form = new application_form();
	$form->save("Thai", "12 december 1982");

	if( $form->name == "Thai" ){
		echo $form->name . " " . $form->birthday;
	} else {
		die("Jag skriver endast ut namnet Thai")
	};
?>
```

En annan viktigt egenskap är att controllern även bestämmer vad man ser på skärmen[5][6]. Den väljer ut en eller flera sidor i systemet som anses lämpliga för en situation och visar innehållet därefter. Vid behov använder den data som hämtats in via modellen, och bestämmer hur och var dessa data skall visas.

2.2.3 View

”View” är den del av systemet som styr systemets grafiska utseende och beteende gentemot användaren. Med beteende menas bl.a. ”popup”-rutor, felmarkeringar och andra funktioner som påverkar upplevelsen.

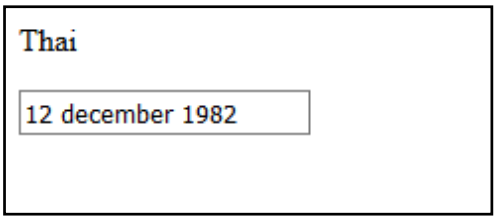
Kodexempel 3. En controller som använder modellen application_form, och anropar en ”view”.

```
<?php
	$form = new application_form();
	$form->save("Thai", "12 december 1982");

	request_once("views/myView.php");
?>
```

Kodexempel 3 visar att en controller skapar ett nytt formulär. Värdena som sparas är fortfarande "Thai"och "12 decenber 1982". Till skillnad från kodexempel 2 så anropas en sida, som är en "view". Kodexempel 4 och figur 4 visar hur myView.php kan se ut.

Kodexempel 4. Koden i myView.php.

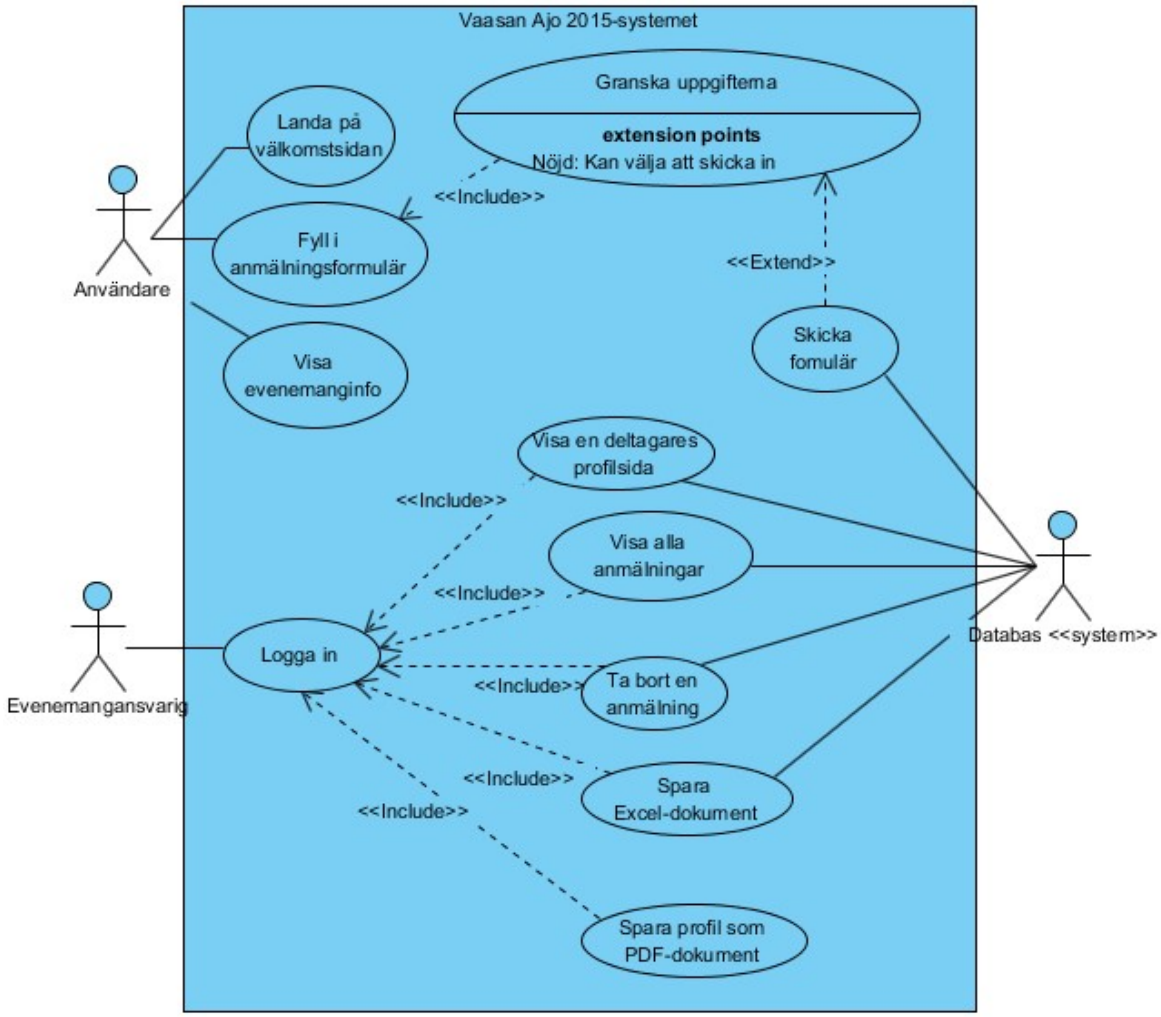
```
<p><?php echo $form->name; ?></p>
<input type="text" value="<?php echo $form->birthday; ?>">
```



Figur 4. Kodexempel 4 i myView.php. "Thai" visas i en paragraf och datumet i en textruta.

2.3 Användardiagram

Utgående från dialogerna med uppdragsgivaren skapades ett användardiagram. I figur 5 ser man vilka kopplingar till systemet som både användaren (klienten) och den evenemangsansvarige har.



Figur 5. Use Case-diagram över rollerna i systemet.

En användare kan visa systemets välkomstsida och information om evenemanget. Denne kan även fylla i anmälningsformuläret, för att sedan granska och skicka in det. De evenemangsansvariga har tillgång till administrativa funktioner, förutsatt att denne loggat in i administrationspanelen.

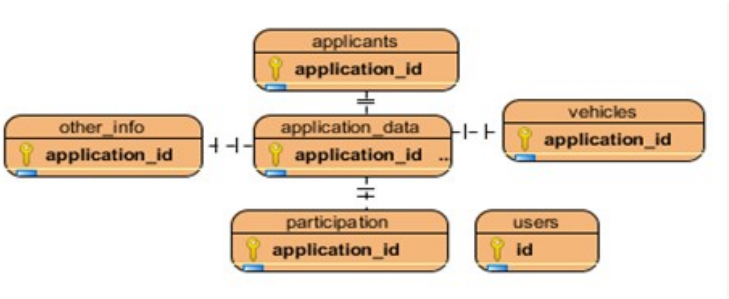
2.4 Databashantering

I detta examensarbete användes en relationsdatabas, vars främsta uppgift var att lagra information från de inskickade anmälningarna. Den andra uppgiften var att tillhandahålla inloggningsuppgifter åt arrangörerna.

Det som kännetecknar en relationsdatabas är att information kan lagras i flera tabeller med rader och kolumner.[7] Databasen användes i det här examensarbetet via anrop från ”model”-klassen i systemets källkod.

2.4.1 Databasdiagram

Utgående från pappersblanketten (se bilaga 1) så var det 23 fält som skulle fyllas i. Det här betydde att databasen skulle hantera lika många fält. I det här fallet togs beslutet att placera fälten i flera tabeller.



Figur 6. Databasdiagram för hela systemet.

Det här valet var en säkerhetsåtgärd inför det praktiska utförandet, ifall uppdragsgivaren skulle vilja få möjligheten att visa detaljsidor för enskilda delar av en inskickad anmälning. Om så hade varit fallet så skulle data kunna hämtats snabbare från en enskild tabell vid inladdning, än från en stor tabell med flera kolumner.

Dessutom var det här ett sätt att få en bättre överblick över var all data skulle placeras. Om inte uppdragsgivaren skulle ge ytterligare förslag under arbetsprocessen, så fanns möjligheten att slå ihop alla tabellerna igen eftersom slutmålet var att samma data skulle läggas in i ett Excel-dokument.

2.5 Säkerhetsaspekter

Säkerhetshanteringen i det här examensarbetet utgick ifrån följande punkter:

- Att textfält och webbläsarens adressfält behövde kontrolleras.
- Att administrationspanelen borde skyddas.
- Sannolikheten för attacker, och på vilka sätt.

Eftersom formuläret varken skulle hantera bankuppgifter, personsignum, betalningsinformation eller andra uppgifter som kunde vara intressanta för brottslingar, så drogs slutsatsen att endast grundläggande säkerhetsåtgärder behövdes vidtas för att skydda systemet. Därav skulle endast följande åtgärder vidtas:

- Skydd mot Session Hijack och Session Fixation.
- Skydd mot Cross Site Scripting och SQL-injektioner.
- Kryptering av personuppgifter.

Under examensarbetet fanns tankar om åtgärder mot Cross Site Request Forgery-attacker, men eftersom systemet endast skulle vara aktiv under en kort tidsperiod, och att den sannolikt endast används en gång per deltagare så bedömdes risken för en sådan attack som obefintlig. Sannolikheten att systemet skulle utnyttjas via arrangörerna själva bedömdes också som obefintlig.

2.5.1 Session Hijacking och Session Fixation

I det här fallet betyder uttrycket "session" en instans på serversidan, där sessionsvariabler lagras. Till skillnad från variabler utanför sessioner så raderas inte sessionsvariabler från servern vid t.ex. sidbyte som standard. En session kan gälla för alltid, eller enbart under en begränsad tid.[8]

När servern startar en session, så skapar den även ett unikt sessions-id och sessionsnamn (statisk på de flesta system) som den sparar. Samma information sparas i en "cookie" som lagras i klientens webbläsare. Sessionsvariabler inom denna id är endast tillgängliga för den klient som har samma id på sin cookie.

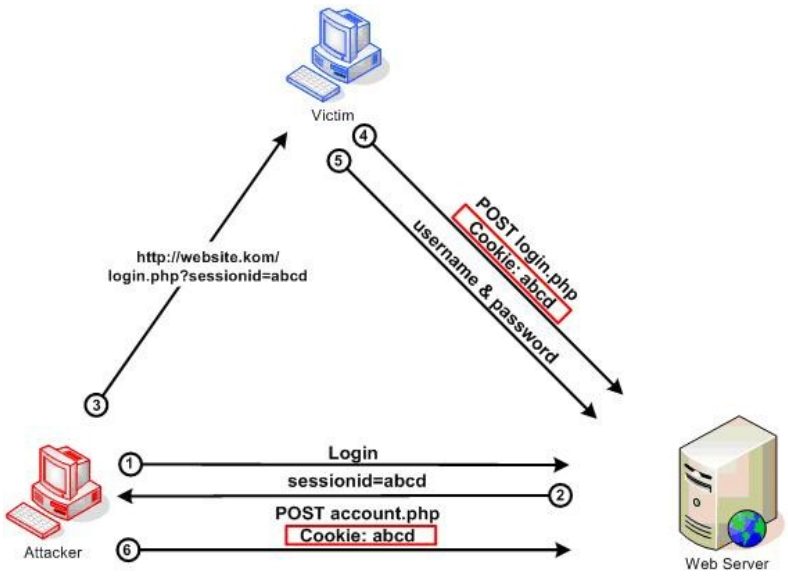
Sessioner används främst för att lagra privat data, som t.ex. systeminställningar och användarrättigheter för en webbapplikation. Därför vill man att data som lagras i sessioner endast kan användas av klienter som de tilldelas. Emellertid finns det åtminstone två olika sätt för tredje part att komma åt dessa data.

Den första kallas "Session Hijacking". Metoden går ut på att avlyssna trafiken mellan en klient och en server, där information om sessioner och cookies skickas fram och tillbaka (inklusive sessionsnamn och id).[9]

Avlyssnaren kan på detta sätt använda samma information för att skapa en identisk cookie som motsvarar sessionen på servern, och på så sätt få tillgång till samma information som den ursprungliga klienten.

Figur 7. Trafikavlyssning mellan ett offer och en server.[9]

Det andra sättet, Session Fixation, går ut på att lura någon annan att använda förinställda sessioner. En klient kan själv skapa sin egen session hos en webbapplikation, och sedan be någon annan att använda det. Den som ursprungligen skapade sessionen kan själv tar ut informationen som kommer in.[10]



Figur 8. En anfallare skapar sin egen session och ger bort den till sitt offer, t.ex. som länk.[10]

Sessioner skulle automatiskt kunna skapas om webbapplikationen ifråga tillåter det, t.ex. via en länk http://localhost/?start_my_session=ettvärde. Samma länk kan skickas till "offret" som man vill hämta information från.

Kodexempel 5. Kod som startar session med id som bestäms av utomstående.

```
<?php
        session_id($_GET["start_my_session"]);
        session_start();
?>
```

2.5.2 Cross Site Scripting

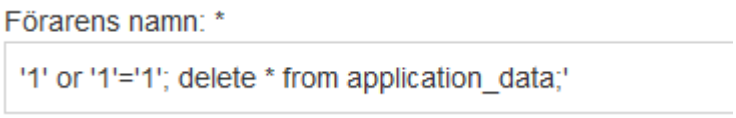
Cross Site Scripting, eller XSS som det också förkortas, går ut på att lägga in skadlig kod i ett system[11]. Det kan göras i syfte att t.ex. manipulera data och funktioner som det använder. Det här sker genom att en person t.ex. skriver skadlig kod i ett textfält, i webbläsarens adressfält eller med JavaScript och sedan skickar in det.

Om värdena i dessa fält inte granskas av systemet så kan insändarens kod göra saker som systemet inte är tänkt att utföra. I kodexempel 6 och figur 9 visas hur ett SQL-kommando kan manipuleras av en okontrollerad variabel (i detta fall kallas det SQL-injektion)[12].

Kodexempel 6. SQL-kod använder en variabel vars värde kan bestämmas av utomstående

```
<?php $sql = "select * from table where name = '" . $_POST["driver_name"] . "'"; ?>
```

$_POST["driver_name"] är en variabel som innehåller ett namn som skrivits in i ett textfält. Det finns inget som antyder att den kontrollerats på något sätt.



Figur 9. Exempel på inskriven kod i ett textfält. Fältet är egentligen menat för att ta emot ett namn.

Om värdet i figur 9 skickas in så kommer SQL-satsen att se ut som kodexempel 7, som raderar all data från en databastabell om den utförs.

Kodexempel 7. Resultat av SQL-injektionen i kodexempel 6, där $_POST["driver_name"] inte kontrollerats

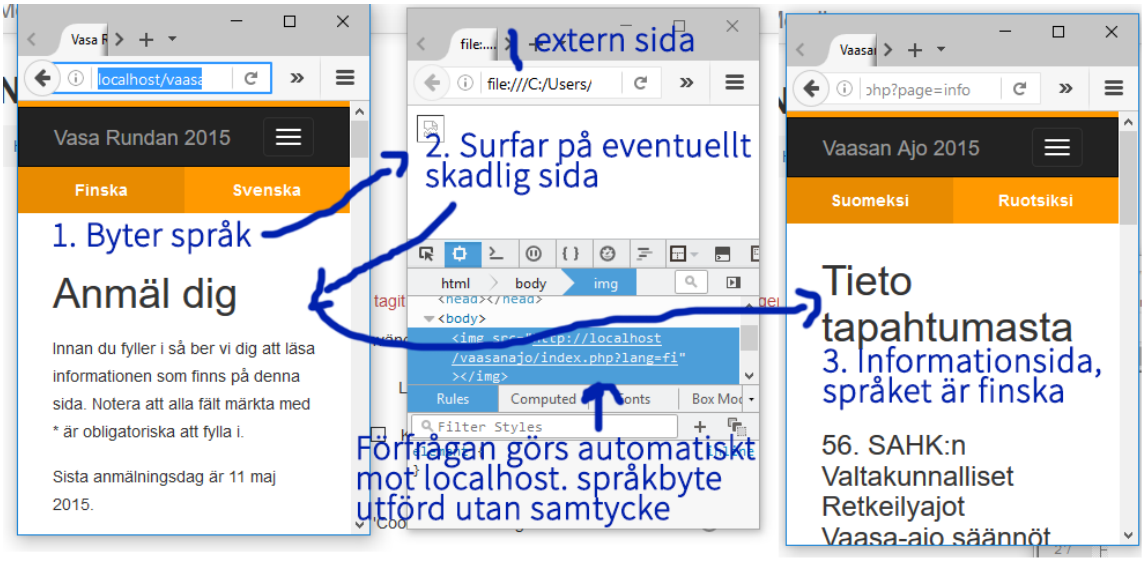
```
<?php sql = "select * from table where name = '1' or '1'='1'; delete * from application_data"; ?>
```

2.5.3 Cross Site Request Forgery

XSRF eller CSRF, som den också förkortas, riktas bl.a. mot klienter som använder en tjänst. Metoden baserar sig på att offret omedvetet skickar anrop över http, t.ex. från externa system.

Låt säga att en användare surfar in på Vaasan Ajo 2015-anmälningsblankett och byter språk till svenska. Länken som används för att utföra detta är http://localhost/vaasanajo/index.php?page=application_form&lang=sv

Om användaren väljer att öppna ett nytt fönster för att göra annat, så kan det hända att denne surfar in på en annan sida, som tvingar webbläsaren att skicka direkta http-anrop till evenemangets hemsida. Problemet är bara att anropet även består av en parameter som utför språkbyte.



Figur 10. Exempel på hur externa sidor kan ändra inställningar.

När användaren återvänder till formuläret och väljer att t.ex. läsa information om evenemanget, så har det grafiska gränssnittet återgått från svenska till finska. Detta gjordes med förinställda anrop från den externa sidan.

Samma princip kan riktas mot administrationssidan. Om arrangören glömmer att logga ut från administrationspanelen, och samtidigt går in på en webbsida som byggts i syfte att förstöra för evenemanget, så kan denna sida utnyttja funktioner som vanligtvis kräver administrationsrättigheter. Det här beror på att anropen sker via arrangörens webbläsare.[13]

En annan aspekt är att XSRF också kan utföras av en hackare. Exempelvis kan denne kopiera ett formulär från en webbapplikation och sedan lägga in det på sin egen webbsida, för att sedan modifiera det. Från webbsidan kan hackaren göra en insändning mot samma adress som originalformuläret. Exakt samma princip gäller om denne väljer att sända in via AJAX.

Kodexempel 8. Ajax-kod med jQuery.

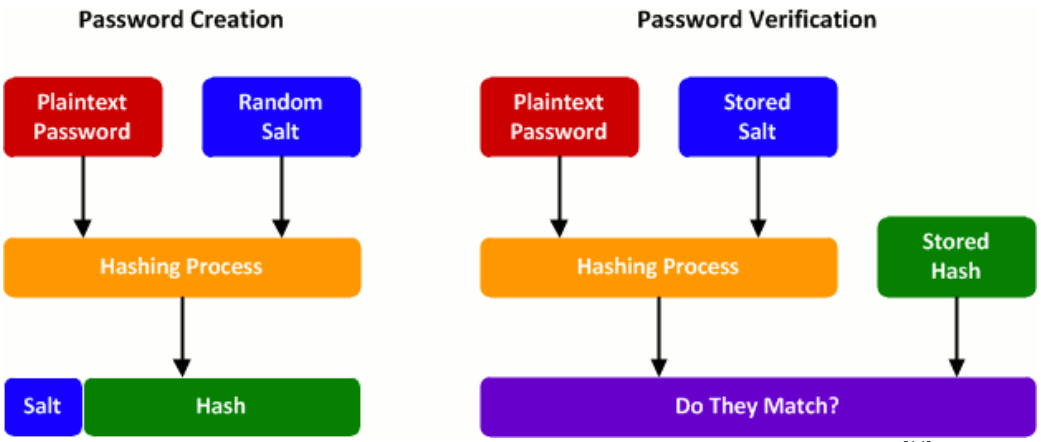
```
<script type="text/javascript"">
$.ajax({
        url: "http://sidansadress.com/index.php?page=skicka_in",
        type: "POST",
        data: {…}
        …
});
</script>
```

2.5.4 Kryptering av information

Att dölja information för obehöriga personer var nödvändigt eftersom personuppgifter behandlades. Dessa behövde skyddas vid eventuella databasläckor. Det var också viktigt att kryptera arrangörernas lösenord, eftersom dessa är menade att vara personliga. I detta examensarbete användes envägskryptering (också kallad ”hashing”) och tvåvägskryptering.

Meningen med ett hash-värde är att den inte skall gå att återskapa. Syftet är att lagra den hashade informationen, och sedan jämföra det med ett värde från samma ”hash”-algoritm[14].

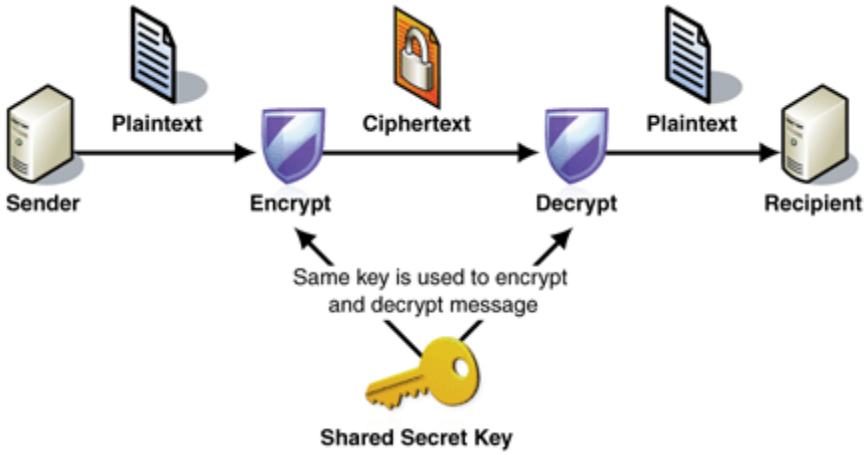
Den här metoden används bl.a. för att bekräfta att ett inskrivet lösenord är korrekt, utan att någon förutom ifyllaren behöver känna till själva lösenordet. Lösenordet lagras endast som ett hash-värde, eventuellt tillsammans med en s.k. salt.



Figur 11. Skapande av ett hashat värde, och jämförelse av hashade värden.[14]

Saltets uppgift är att göra det svårare att få ut det egentliga lösenordet från hash-värdet. T.ex. Om arrangören har ett vanligt förekommande lösenord vars krypterade motsvarighet är osaltad, så skulle lösenordet avslöjas omgående om databasen hade läckt ut och någon hade jämfört de lagrade ”hash”-värdena med ”hash”-tabeller på Internet.

Risken minskar emellertid om lösenordet krypteras tillsammans med en lång och slumpvald teckensträng som genererats av någon algoritm[14]. Det här skapar ett hash-värde som kan vara tillräckligt stark och som osannolikt kan återfinnas i några listor.



Figur 12. Kryptering och dekryptering av text med hjälp av en nyckel.[15]

Tvåvägskrypteringens däremot, innebär att information förvandlas till något oläsligt med hjälp av en egenskriven nyckel. [15] Samma nyckel används för att återskapa den krypterade informationen. Den här metoden användes för att kryptera personuppgifter när en anmälning lagrades, och för att återskapa samma information t.ex. vid nedladdning av Excel-dokumentet.

3 Teknikval

Mot bakgrund av analysen i kapitel 2, har lämpliga ramverk och programmeringsspråk valts. Meningen är att ha en enkel sammansättning verktyg.

3.1 Mjukvaror och miljö

Då Vasa Veteranbilssällskap är en liten förening, och eftersom systemet varken var tänkt för framtida bruk eller vidareutveckling, så valdes verktyg som är gratis att använda för att utveckla webbapplikationen.

I det här fallet valdes WAMP (en sammansättning av Apache, MySQL och PHP[16] för Windows) som utvecklingsmiljö, främst för att den installerar och automatiskt konfigurerar alla nämnda mjukvaror som behövs för att snabbt börja utvecklingsprocessen.

3.1.1 Apache HTTP Web Server

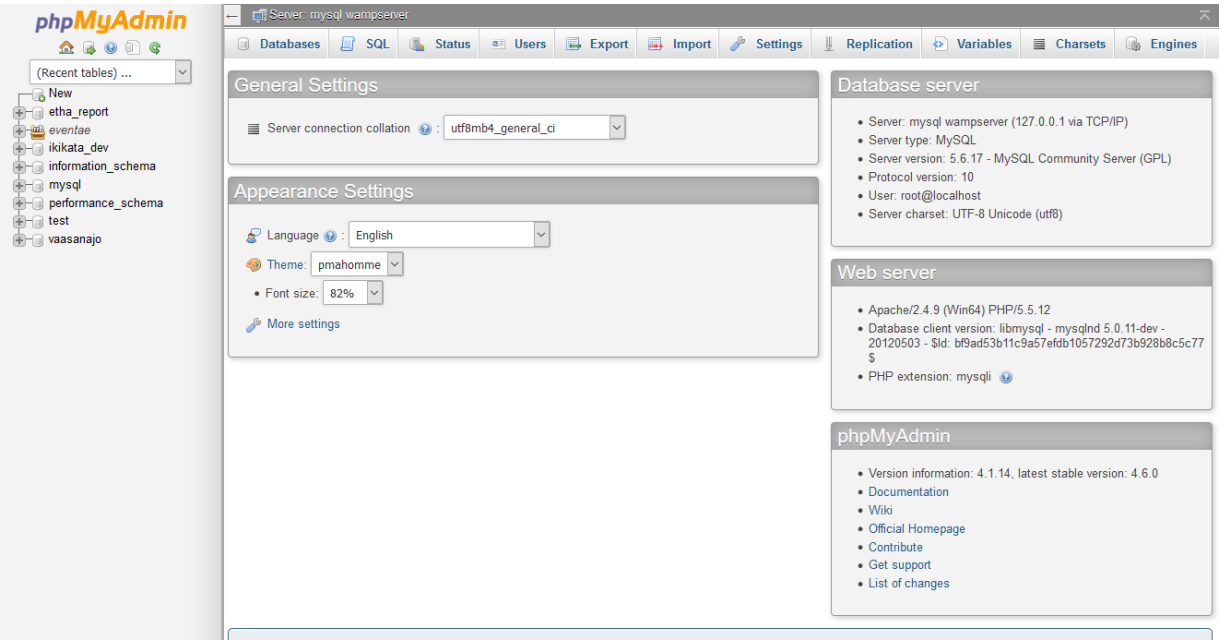
Denna webbserver, som underhålls av Apache Software Foundation, driver idag majoriteten av alla webbsidor och applikationer på internet. Bland dessa hittar man bl.a. Adobes, PayPals och Apples officiella webbsidor. Enligt en rapport från W3Techs så har Apache haft en marknadsandel på över 50% de senaste sex åren, om än ständigt sjunkande.[17]

Exempel på programmeringsspråk som stöds av Apache Server är Python och PHP.

3.1.2 MySQL

Liksom Microsoft SQLServer och PostgreSQL, så är MySQL ett databashanteringssystem för relationsdatabaser. Den tillhandahålls idag av Oracle Corporation, och är ett av de mest populära databashanteringssystemen ute på marknaden .[18]

MySQL administreras antingen via kommando-prompten eller genom ett grafiskt användargränssnitt som t.ex. PhpMyAdmin eller Workbench.



Figur 13. PhpMyAdmin. Webbaserad administrationsgränsnitt för MySQL-hantering.

Via PhpMyAdmin kan man bl.a. skapa databaser och tabeller. Man kan även ta bort, lägga till och redigera befintlig data, samt exportera dessa till textfiler. Andra funktioner är manipulering av tabellstrukturer (t.ex. lägga till primärnycklar och ställa in tabellreferenser). Detta grafiska gränssnitt stöder även provkörning av SQL-kommandon.

3.2 Programmeringsspråk

Eftersom målet var att producera ett webbaserat system så användes standardsspråk (HTML, CSS och JavaScript) för att bygga systemets grafiska användargränssnitt. För datahanteringen användes PHP och SQL som är kompatibla med de mjukvaror som valts.

3.2.1 HTML

Grunden i ett grafiskt webbgränssnitt är HTML (Hypertext Markup Language), som är ett märkspråk som utvecklats och upprätthålls av World Wide Web Consortium (W3C). Språket tillför struktur och placering av olika element, t.ex. textrutor, tabeller och länkar.[19]

HTML beskriver gränssnittet hos en webbsida, som sedan tolkas av webbläsaren efter att den laddats in via http-protokollet. Senaste versionen är HTML5, som bl.a. möjliggör lokal lagring av data, ljud- och videouppspelning etc.

Kodexempel 9: HTML-kod med attribut och text.

```
<div class="application-component row" id="application-menu">
        <div class="col-md-12">
                <p>40-åriga Vasa Veteranbilssällskap rf önskar alla fordonsentusiaster välkomna på en solig
                runda i Vasas sjönära omnejd! Cykel-, moped- och traktorivrare erbjuds en kortare rutt.</p>
        </div>
</div>
```

Oftast är det en fördel att vara medveten om när man skall och inte skall använda olika typer av element. I många fall anses det vara "good practice" att endast låta HTML beskriva strukturer, medan betéende och design lämnas åt andra tekniker. Exempelvis är följande inkorrekt eftersom färgsättning och radbrytning sker direkt i HTML:

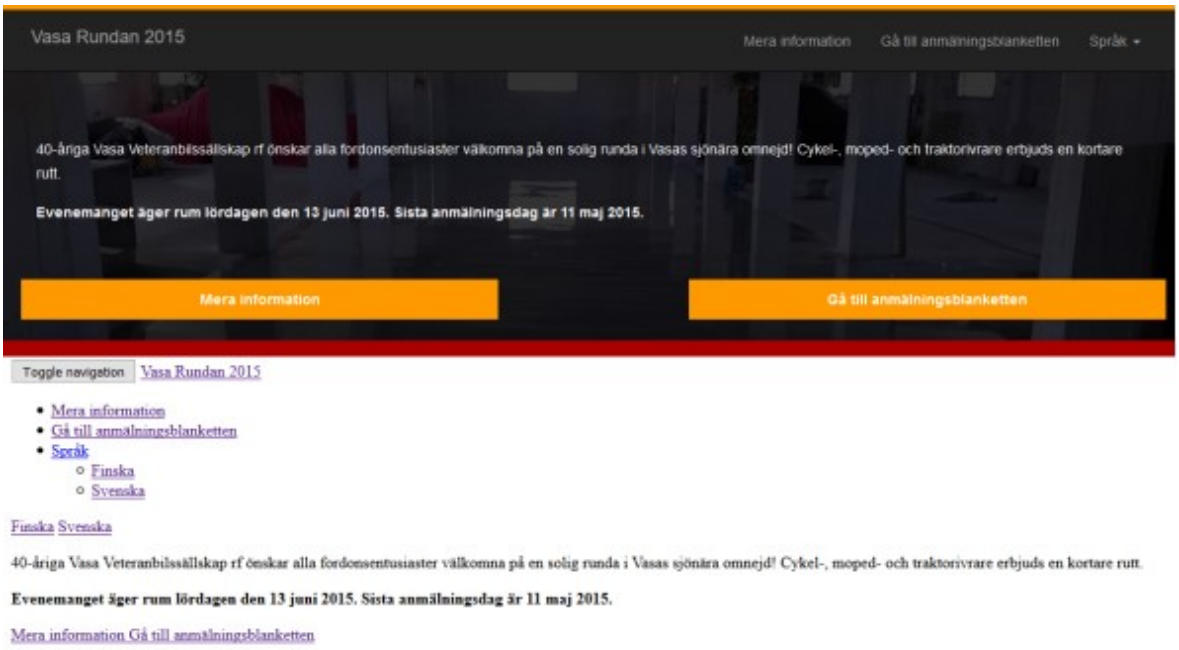
```
<span bgcolor="#000000" id="test">Hello World!</span><br>
```

3.2.2 CSS

CSS är ett stilspråk (Cascading Style Sheet) som används för att definiera hur HTML-element ska se ut ur ett grafiskt perspektiv. Det kan röra sig om allt från textstorlek, bakgrundsfärg till ramutseende på olika element. [19]

Kodexemel 10: CSS-kod

```
.hidden-text {
        background: #000000;
        font-weight: bold;
        border: 3px dashed #ff0000;
}
```



Figur 14. Samma webbsida med och utan CSS.

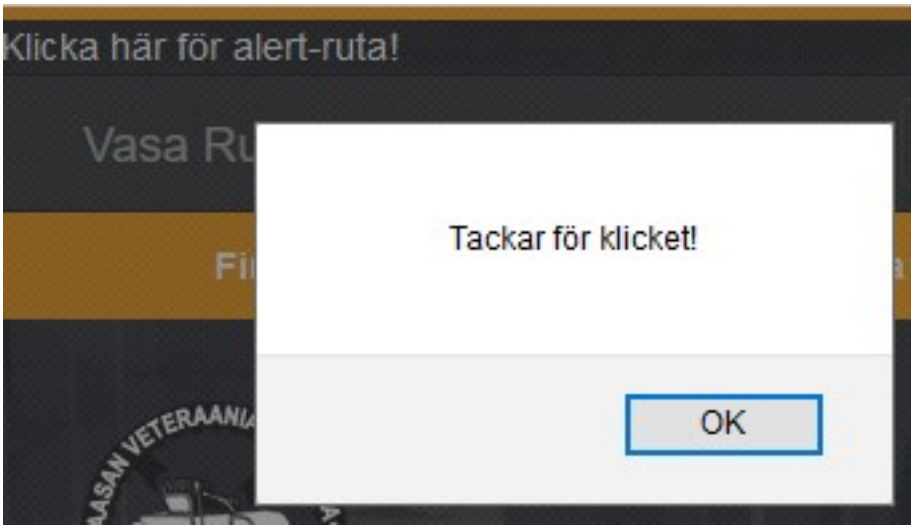
En av de större fördelarna med CSS är att man kan använda samma kod på flera typer av HTML-element, och därmed inte behöver definiera samma CSS-kod för varje element man skapar.

3.2.3 JavaScript

Medan HTML och CSS utgör grunden för det grafiska gränssnittet så kan gränssnittet manipuleras, t.ex. genom användarens interaktion. Detta utförs med JavaScript, som är ett klientbaserat programmeringsspråk.[20] Ett enkelt exempel är att man klickar på en länk för att få upp en ruta med ett textmeddelande.

Kodexempel 11: Klicka på HTML-länken så dyker ett textmeddelande upp.

```
<a id="alertruta" href="javascript:;" >Klicka här för alert-ruta!</a>
<script type="text/javascript">
        document.getElementById("alertruta").onclick = function(){
                alert("Tackar för klicket!");
        }
</script>
```



Figur 13: "Alert"-ruta.

Idag används JavaScript i många ramverk och kodbibliotek för olika ändamål, t.ex. för kommunikation i realtid. Två exempel på ramverk som drar nytta av detta är Cordova och Ionic, som är två populära verktyg för produktion av applikationer till mobiler.

3.2.4 Json

Javascript Object Notation är ett datalagringsformat (.json-filer), och även en öppen standard, som är ämnad för datalagring[21], på samma sätt som XML (eXtended Markup Language). Json har växt fram under de senaste åren eftersom den anses vara lättare att hantera och läsa än XML, som är taggbaserad.

Data som lagras i Json skrivs enligt samma mönster som JavaScript-arrays, dvs som attribut-värde-par.

Kodexempel 12: Json-kod

```
{
	"attribut": "värde",
	"attribut2": {
		"under_attribut": "värde 2";
	}
}
```

Kodexempel 13: XML-kod

```
<attribut>värde</attribut>
<attribut2>
	<under_attribut>värde 2</under_attribut>
</attribut2>
```

Som man kan se är så tar Json mindre plats i ett dokument, och det är väldigt lätt att hitta vad man söker, gentemot XML som kan nästla värden i flera nivåer samtidigt som otydligheten blir större ju fler värden man lagrar. Jsons fördel är att den klarar av att kategorisera på ett snyggt sätt.

Exempel på sammanhang där Json kan användas, är när man vill spara enskilda inställningar, eller lagra förbestämda värden utan att använda databasen. Det här är användbart när man t.ex. byter språk på statiskt innehåll, eller sparar inställningar i en applikation.

3.2.5 PHP

PHP (PHP: Hypertext PreProcessor) är ett serverbaserat programmeringsspråk och används för datahantering. PHP tolkas vanligtvis i dokument som har filtypen ”.php”, om inte annat konfigurerats på servern, och kan användas tillsammans med klientbaserade språk som HTML-, JavaScript- och CSS.[22]

PHP är känt för att köras tillsammans med Apache Server, men stöds idag även av Microsoft IIS.

Kodexempel 14: PHP-kod

```
<?php
    if( !isset( $_POST[”values”] )){ echo ”Nothing has been submitted!”; }
?>
```

Med PHP kan man även lagra variabler under en ”instans” i ett dokument, för att använda den vid någon senare skede. Se kodexempel 15.

Kodexempel 15: Variabel som skapats inom PHP-taggarna, och används senare i samma dokument

```
<?php $text = ”hello world”; ?>
<div>Ett HTML-element</div>
<?php echo $text; ?>
```

3.2.6 SQL

Förkortningen står för "Structured Query Language" och används för hantera data i relationsdatabaser[23]. I det här projektet används SQL för att lägga till anmälan från deltagare, samt för hämtning och borttagning av inkomna anmälningar via administrationspanelen.

För att lägga till en ny användare kunde t.ex. följande SQL-kommando användas:

```
insert into users (user_name, password) values ("foo", "bar");
```

För att ta reda på hur många anmälningar som finns i databasen, kunde denna sats användas:

```
Select count(*) as application_count from application_data;
```

3.3 Ramverk och bibliotek

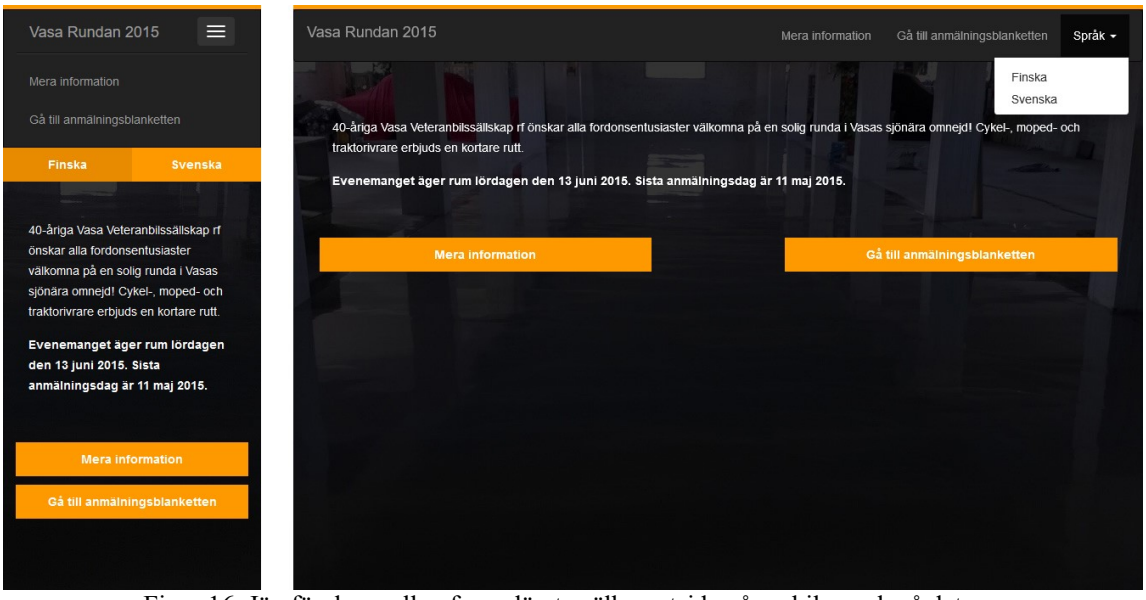
Att allt fler människor har börjat använda smarttelefoner och surfplattor de senaste åren, vilket har gjort att även nätbaserade tjänster blivit tvungna att anpassa sig. Det här gäller främst användargränssnittet och dess beteende i både funktion och utseende.

Man kunde kanske inte uppskatta hur många människor som skulle tänkas anmäla sig till Vaasan Ajo 2015 via andra enheter än datorer, men ingen arrangör vill riskera att förlora potentiella deltagare. Därför måste formuläret kunna anpassa sig till alla enheter och kännas bekvämt när den används. För detta ändamål valdes front end-ramverken Bootstrap och jQuery.

3.3.1 Bootstrap 3

Det här ramverket utvecklades av Twitter och innehåller ett stort antal komponenter skrivna i JavaScript och CSS. Bootstrap används för att utveckla responsiva webbapplikationer, dvs applikationer vars gränssnitt och beteende automatiskt anpassar sig till de enheter där de körs (mobiler, surfplattor och datorer).[24]

Ett av Bootstraps nyttigaste egenskaper är att man inte behöver ladda ned ”hela” ramverket för att det skall fungera felfritt, man kan välja att ladda ned enskilda komponenter tillsammans med ”kärnan”.



Figur 16. Jämförelse mellan formulärets välkomstsida på mobilen och på datorn.

Den mest intressanta komponenten är ”grid”-komponenten, som gör det möjligt för ett webbgränssnitt att automatiskt anpassa sig till olika enheter och skärmupplösningar. En annan relevant komponent är navigationsmenyn, som kan öppnas och stängas med ett klick på smarttelefoner.

Kodexempel 16: Implementation av "grid"-komponenten i HTML.

```
<div class="container">
        <div class="row">
                <div class="col-md-12"></div>
        </div>

        <div class="row">
                <div class="col-md-6"></div>
                <div class="col-md-6"></div>
        </div>
        ...
</div>
```

"Gridden" utgörs av rader och kolumner inne i ett block-element. Både rader ("row") och kolumner ("col-") definieras normalt som div-element, och maximala antalet breddenheter för varje rad är 12, därav värdena på deras ”class”-attribut. Se kodexempel 16 och figur 16.

För varje rad där kolumnerna tillsammans bildar 12 enheter, skapar man en ny rad för att fortsätta fylla innehållet. Bredden för varje kolumnklass på mindre enheter tar upp hela enhetens breddupplösning, och placerar sig under varandra i den ordning de skrivits i HTML-koden.

3.3.2 jQuery och jQuery UI

jQuery- och jQuery UI-bibliotekens roll i det här examensarbetet var att bidra med komponenter som förenklar användarupplevelsen på alla plattformar. Medan Bootstrap riktar in sig på anpassning ur enhetsperspektiv, så bidrar jQuery med användargränssnittets beteende genom färdigprogrammerade komponenter.[25]

.

Figur 17. Kalenderfunktion i t.ex. mobilen med jQuery UI.[26]

En annan fördel med jQuery är att den gör det lättare att hantera HTML-element än när dessa manipuleras med JavaScript. Om man är van med CSS så kan man använda dess ”selection”-syntax för att markera vilka HTML-element som jQuery skall manipulera.

Kodexempel 17: En div-element väljs med hjälp av CSS-syntax.

```
$(”div.textruta”).hover(
        function(){
            alert(”Hej!”);
        }, function(){
                alert(”Hejdå!”);
        }
);
```

I kodexempel 17 visas att ett HTML-element har valts med hjälp av en så kallad ”selection”-syntax, som i detta fall är ”div.textruta”. När man lägger musen över detta HTML-element så uträttas funktionen alert(”Hej”) och när man styr bort musen från elementet så uträttas funktionen alert(”Hejdå”). Det här är möjligt tack vare hover(), som är en ”event” i jQuery.

3.3.3 Uteslutning av back end-ramverk

Projektet använde varken färdiga back end-ramverk eller system för datahantering, trots att många andra tjänster använder sig av dessa. Exempel på tilltänkta ramverk var CodeIgniter, Drupal eller Joomla.

Beslutet grundade sig på analysen i kapitel 2, ur vilket slutsatsen var att ett sådant system vore överflödigt. CodeIgniter[27], som vore lämpligast i sammanhanget på grund av dess MVC-kapacitet, har för många irrelevanta moduler – samtidigt som många av de nödvändiga funktionerna har en helt annan struktur och ett helt annat tillvägagångsätt än traditionell PHP.

Samtidigt är Drupal och Joomla mera lämpliga för större och mångsidigare projekt, och är därmed inte ett alternativ för detta examensarbete.

Efter avvägning mellan tidsåtgång, deadline och behov, så drogs slutsatsen att ett mindre system skulle gå att bygga snabbare utan ramverk då ingen tid behövde användas till att lära sig det. Under samma tid kunde man även kommunicera med arrangören och samtidigt göra framsteg i arbetet.

4 Det praktiska utförandet

I det här fallet planerades det grafiska gränssnittet först innan den överfördes till kod. Efter att gränssnittet programmerats så implementerades logiken och diverse säkerhetsåtgärder för att skydda systemet mot eventuella intrång. Den praktiska delen av detta examensarbete fokuserade mycket på att följa MVC-modellen.

4.1 Planering av det grafiska användargränssnittet

Det grafiska gränssnittet planerades i ett bildbehandlingsprogram. Idén var att skapa ett enkelt gränssnitt som kunde lyfta fram de viktigaste aspekterna av systemet: anmälningsformuläret, informationssidan och funktionerna i administrationspanelen. Detta gjordes för att skapa en mall, vilken man snabbt kunde överföra till HTML och CSS.

I det här fallet skapades gränssnittet utifrån användardiagrammet i figur 5. Eftersom det var innehållet som fokusen skulle ligga på, så användes endast tre färger: svart och vitt, samt en komplementfärg, för att göra utseendet tilltalande.

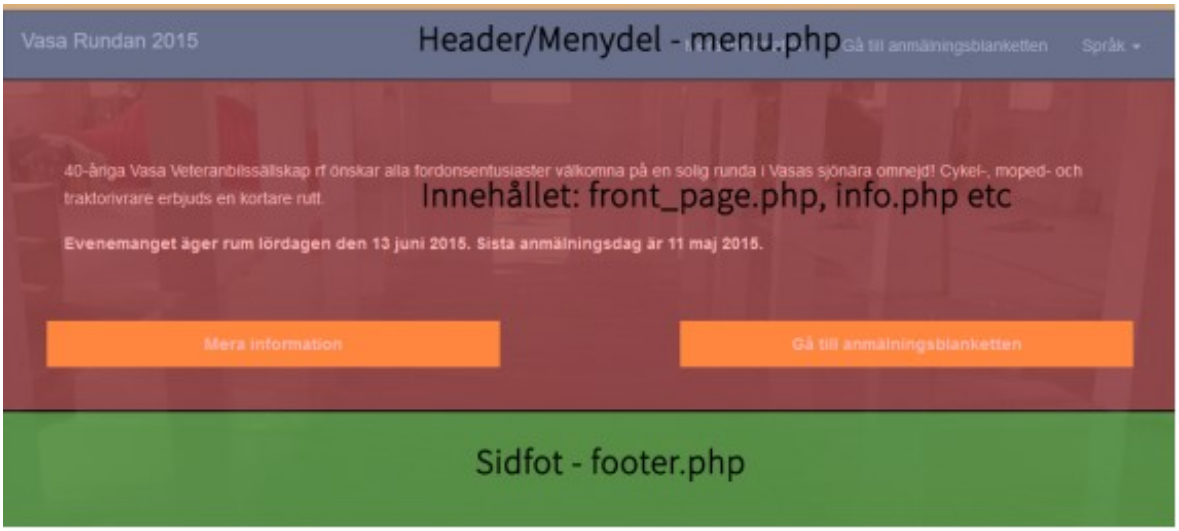
För att komplettera detta så framhävdes även länkarna i menyn på alla sidor, för att hjälpa betraktaren att navigera och få kännedom om vad Vaasan Ajo 2015 var för något. Se figur 14 för exempel.

4.1.1 Validering av HTML- och CSS-kod

W3C tillhandahåller valideringsverktyg som gör att man kan granska och rätta sin kod enligt standard[28]. Det anses inte endast vara en fördel eller "good practice" hos de professionella utvecklarna, utan även som ett sätt att minska tid och arbetsbelastning.

I det här fallet validerades den skrivna koden för att undvika strukturfel i det grafiska gränssnittet. Det här steget var viktigt innan serverprogrammeringen kunde utföras, för då kunde man utgå ifrån att det inte var fel på den ursprungliga HTML-koden om något skulle se felaktigt ut (t.ex. om HTML läggs till via PHP etc.). På detta sätt kunde tidsåtgång till eventuella felsökningar minskas.

4.2 Tillämpning av MVC och serverprogrammering



Figur 18. Controllern är hela sidan. View:erna är de färgade områdena.

Figur 18 visar hur användargränssnittet är strukturerat. De färgade delarna är "views", och visas enligt de villkor som man ställt in i controllern. Modellen syns inte eftersom den inte har någon koppling till användargränssnittet enligt figur 3.

I det här fallet var det index.php som utgjorde controller, medan menu.php, front_page.php och footer.php var "views". Koden i index.php visas i kodexempel 18 (grovt förenklad).

Kodexempel 18: Exempel av controller-kod i index.php.

```
<?php
	session_start();
	require_once( "models/modellib.php" ); //modellbibliotek
	if(!isset($_GET["lang"]) { $_GET["lang"] = "fi"; }

	require_once("views/head.php");  // Statisk
	require_once("views/menu.php");  // Statisk

	if( isset($_GET["page"]) ){ // Vilkorsbaserad view
		$page = new page( $_GET["page"], $_GET["lang"] ); // Definiera sidans egenskaper
		if( file_exists($file)){
			require_once( $file );
		} else {
			require_once( "views/404.php" );
		}
	} else {
		$page = new page( "front_page", $_GET["lang"] );
		$file = "views/" . $page->get_id() . ".php";
	}
	require_once("views/footer.php");  // Statisk
?>
```

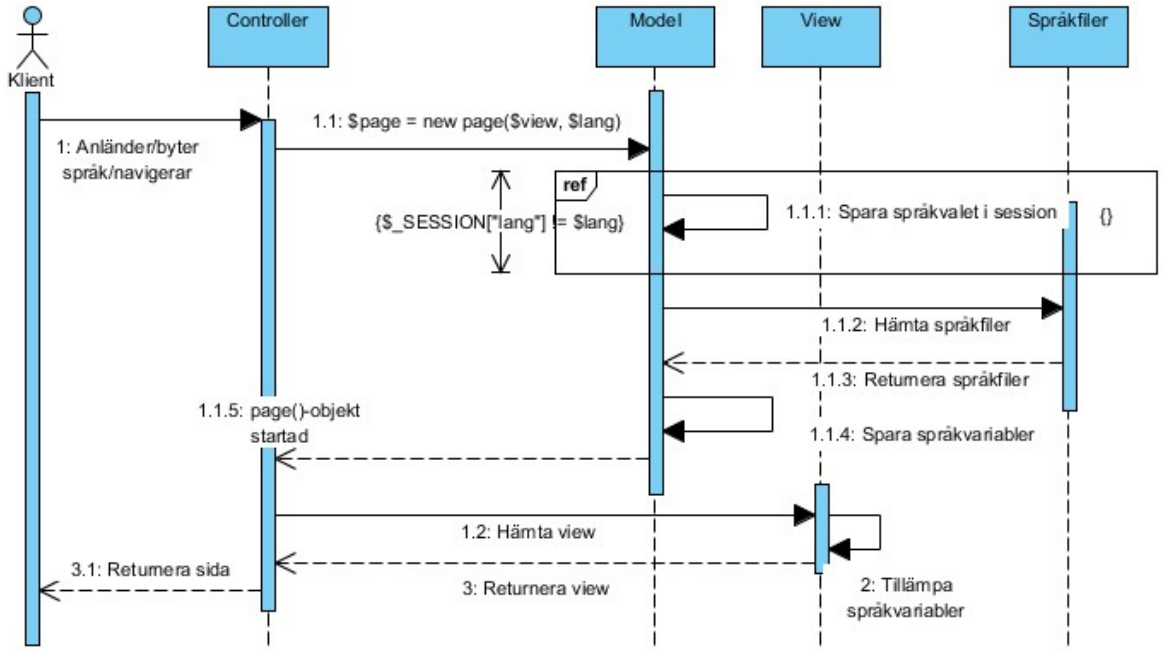
4.2.1 Navigations- och språkhantering

Utifrån kodexempel 18 navigerade man i systemet genom att lägga till en "page"-parameter i adressfältet. Om denna parameter inte definierats så visades framsidan. På motsvarande sätt kunde man ändra språk genom att lägga till en ”lang”-parameter. Tabell 1 visar hur navigationen fungerade.

Tabell 1. Exempel på hur navigationen fungerade.

URL med parametrar	**Plats på sidan**	**Språk**
index.php?page=application_form&lang=sv	Formulärsida	Svenska
index.php?page=info&lang=fi	Informationssida	Finska

Alla ”views” laddades in i controllern vid anrop från en webbläsare och specifika inställningar gjordes med $page-objektet, bland annat språkhanteringen. Figur 19 visar denna process närmare.



Figur 19. Sekvensdiagram av navigations- och språkhanteringen vid en sidladdning.

Språkhanteringen använde en sessionsvariabel som gällde över hela systemet. Om ”lang”-parametern inte längre skulle stämma överens med den så bytte systemet språk.

Varje ”view” som kunde laddas in i det rödmarkerade området i figur 18 hade egna språkfiler i json-format. Det blåmarkerade området hade också sina egna språkfiler. Vid varje anrop slogs värdena från dessa filer ihop, och sparades i en array, som sedan användes i de berörda ”view”:en.

För att vara säker på att navigationslänkarna inte innehöll dubbletter av parametrarna, så tillämpades kodexempel 19 i controllern för att bygga länkar som kunde användas för att navigera och för att byta språk.

Kodexempel 19. PHP skapar länkar som tillåter språkbyte från alla view utifrån enskilda $_GET-variabler.

```php
<?php
	$url_param_fi = array_merge($_GET, array("lang" => "fi"));
	$url_fi = http_build_query($url_param_fi);
?>
```

Flera språkbyten efter varandra skulle aldrig kunna likna följande exempel:
Index.php?page=application_form&lang=sv& lang=sv&lang=sv&lang=sv&lang=sv

Utan istället förkortas till en adress med en enda ”lang”-parameter:
index.php?page=application_form&lang=sv

4.3 Anmälningsformuläret

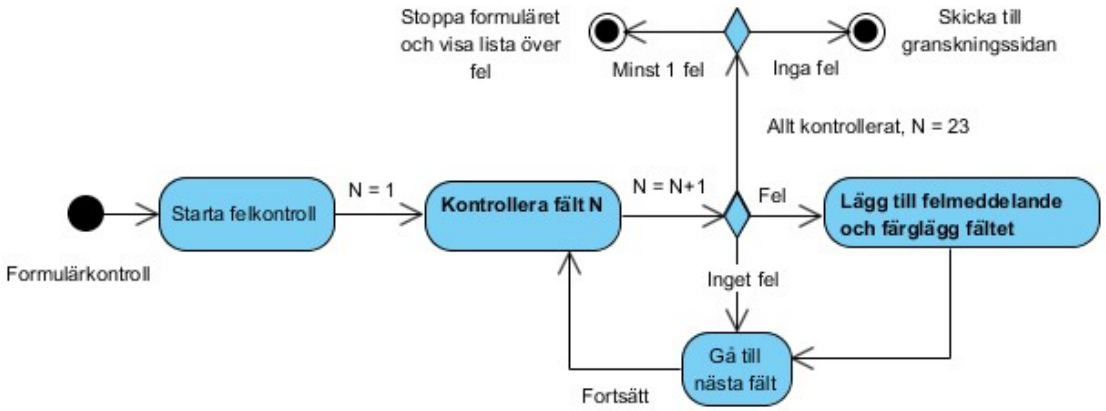
Detta formulär tillät klienten att fylla i nödvändiga uppgifter för deltagande i Vaasan Ajo 2015, och bestod av samma fält som fanns i pappersblanketten. Via formuläret kunde man också kontrollera sina uppgifter innan man skickade in sin anmälan.

Enligt pappersblanketten kunde endast tre fält lämnas tomma av ifyllaren. I detta formulär erhöll motsvarande fält värdet "null" om de lämnades tomma. Fält där ifyllaren skulle ange siffor för kostnadsberäkningar, tilldelades automatiskt värdet 0 om de inte fylldes i.

Varje gång formuläret laddades in i webbläsaren tilldelades den en unik ”nyckel”-variabel, som användes för att kontrollera att alla insändningar gjorts från originalformuläret, och inte från kopierade formulär på externa system.

4.3.1 Felhantering

Formuläret hade en mekanism på klientsidan och en mekanism på serversidan som hanterade fel. Den klientbaserade felhanteringen utfördes på själva anmälningsformuläret med hjälp av JavaScript och jQuery. Dess enda syfte var att meddela klienten om något fyllts i på fel sätt.



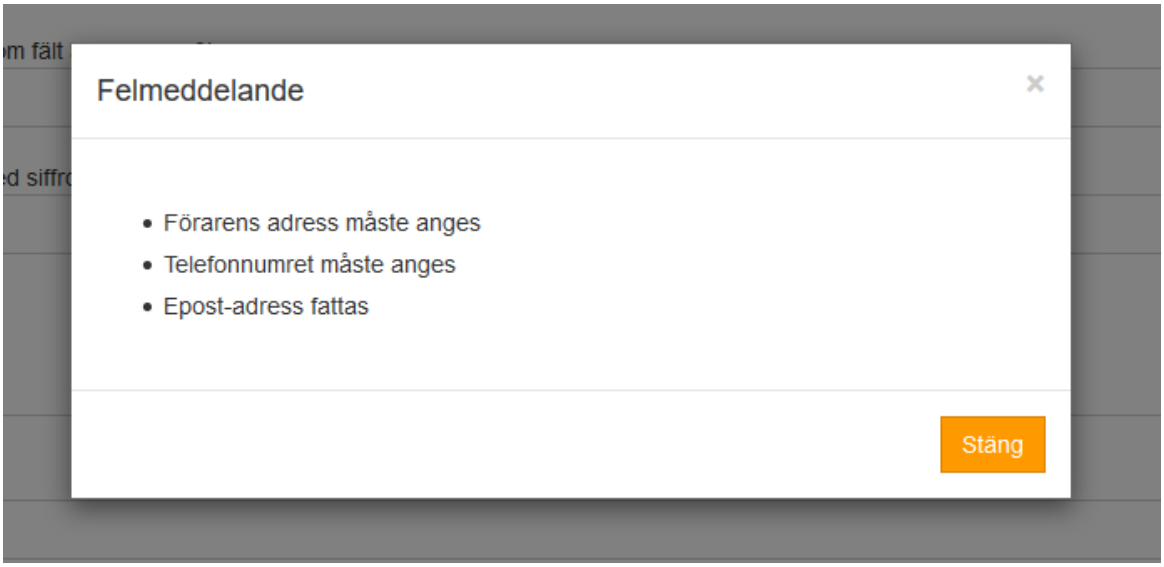
Figur 20. Aktivitetsdiagram över formulärkontrollen på klientsidan.

När man klickade på "Förhandsgranska-knappen" så aktiverades en funktion som kontrollerade alla fält utifrån förinställda krav. Figur 20 och kodexempel 20 förklarar detta mera ingående.

Kodexempel 20: Figur 20 återskapad i kod.

```
$("#application-form").submit( function(e){
        var felmeddelanden = []; var fel_hittad = false;
        if( $("#driversname").val().length < 1 || isNaN($("#driversname").val()) ){

                // Lagra felmeddelande
                felmeddelanden.push("Förarens namn måste anges med bokstäver ");

                // Markera detta fält för korrigering hos klienten
                $("#driversname").css("box-shadow", "0px 0px 5px #ff9900");

                fel_hittad = true; // Fel upptäckt
        }
        // Jänför värden i andra fält med deras respektive krav

        // Kontrollera om fel hittats
        if( fel_hittad == true ){
                e.preventDefault();  Stoppa formuläret

                /* Aktivera notifikationsruta */
                 return false;
        }
    }
);
```

Om minst ett fel hittades så stoppades formuläret från att skicka klienten vidare till granskningssidan. Alla felmeddelanden som samlats in visades i en notifikationsruta.



Figur 21. "Modal"-komponenten i Bootstrap, modifierad med egen CSS- och JavaScript-kod.

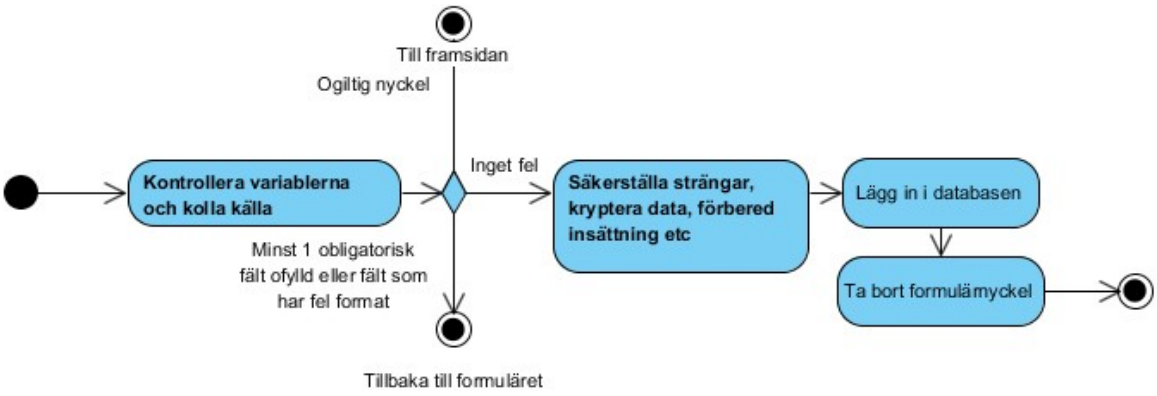
”Notifikationsrutan” är en av komponenterna i Bootstrap[29] som, tack vare sin förmåga att ”glida in” i applikationen, lätt kunde få klientens uppmärksamhet. Alla felmeddelanden skrevs ut i denna komponent med hjälp av en iterationssats. Se kodexempel 21.

Kodexempel 21. JavaScript-kod som visar "modal"-notifikationen om fel hittats i kodexempel 20.

```
$("#errorModal").modal({ show: true } ).on("shown.bs.modal", function(e){
        var error_string = "<ul>";

        for(i = 0; i <= felmeddelanden.length-1; i++){
            error_string += "<li>" + felmeddelanden[i] + "</li>";
        }
        error_string += "</ul>";

        $(".modal-body", this).html(error_string);
    }
);
```

Om det inte fanns några fel hos anmälningsformuläret så skickades klienten till granskningssidan. Där kopierades alla $_POST-värden till en sessionsarray, för att kunna återställa fälten i formuläret om klienten hade velat ändra något. Samma array användes även för att sätta in uppgifterna i databasen.

Serverns felhantering skedde efter att klienten granskat sina uppgifter och sedan valt att lagra dessa i databasen. Felhanteringen var beroende av sessionsarrayn och av formulärets unika nyckel.



Figur 22. Aktivitetsdiagram över fel- och säkerhetshantering innan insändning på serversidan.

Kontrollen utfördes enligt figur 22. Om ett externt system (se kapitel 2.5.2) skulle ha skickat in data så skulle insändningen inte ha godkänts på grund av sessionsnyckeln, och om ett ogiltigt värde hittades så skickades klienten tillbaka till formuläret. Huruvida ett värde var giltigt eller inte, avgjordes av samma villkorssatser som användes i kodexempel 20.

4.3.2 Hantering av injektioner

Systemet, därmed också anmälningsformuläret, kontrollerade alltid värden som skrivits in av klienten. Värden som skulle användas för utskrift i en ”view” kontrollerades alltid med PHP-funktionen htmlentities(). Denna funktion ersatte alla tecken som reserverats av märkspråket HTML till deras entitetsmotsvarigheter.

Kodexempel 22.htmlentities() används på JavaScript-kod för att förhindra körning vid utskrift.

```
<?php htmlentities("<script type='script/javascript'>document.write()</script>", ENT_QUOTES, ”UTF-8”); ?>
```

Kodexempel 22 skulle resultera i följande sträng, som inte skulle kunna utföra något.

<script type='script/javascript'>document.write()</script>,

Denna åtgärd vidtogs bl.a. på granskningssidan för anmälningsformuläret. Gällande databasen så kommunicerade systemet med MySQL via PDO-klassen, vars prepare()-funktion användes för att undvika SQL-injektioner.

Kodexempel 23. PDO:s prepare()-funktion rensar SQL-sats från skadliga tecken

```
<?php
$query = "insert into users(name, password)
values(:name_input, :password_input)";
$statement = $db->prepare($query);
$statement->execute(array
        (
                "name_input" => "<a href='test.html>test</a>",
                "password_input" => "<script src='http://localhost.com'></script>"
        )
);
?>
```

Anledningen är att htmlentities() enbart konverterar tecken som reserverats av HTML och hindrar nödvändigtvis inte databasen från att tolka SQL-satser på ”fel sätt” om medföljande variabler innehåller tecken som inte är tänkta att vara där.

Kodexempel 23 använder därför ”prepared statements” som separerar en SQL-sats från ”platshållare”. Dessa platshållare börjar alltid med kolontecken och kan knytas till vilka värden som helst. I detta fall knyts dessa till HTML-kod, varav en av dem har ett oavslutat HTML-attribut. Se figur 23.

Figur 23. HTML-kod skickas till databasen och läggs in i en tabell utan modifikationer.

I figur 23 används ”:name_input” och ”:password_input” som platshållare. Båda separeras från resten av SQL-satsen med hjälp av prepare()-funktionen, och skickas sedan till databasen. Värdena i platshållarna behandlas därefter skilt från själva SQL-satsen, därför sker inte några injektioner.

Databasens arbete avbryts inte heller av att det kan finnas enskilda apostrofer och citationstecken i insättningsvärdena, som i vanliga fall resulterar i syntaxfel om man sätter in dem direkt i SQL-satsen. Se figur 23 och jämför med följande exempel:

```
Insert into users(name, password) values('<a href='test.html>test</a>', '<script
src='http://localhost.com'></script>');
```

4.3.3 Kryptering av personuppgifter

Klientens personliga uppgifter (t.ex. namn, adress, bilens registreringsnummer och telefonnummer m.fl.) lagrades aldrig i databasen som ren text. Av ren princip krypterades dessa uppgifter innan lagring.

För detta ändamål användes funktionen openssl_encrypt() för kryptering och openssl_decrypt() för att återskapa de krypterade värdena vid användning i administrationspanelen. Icke-personliga uppgifter som t.ex. antalet passagerare krypterades inte.

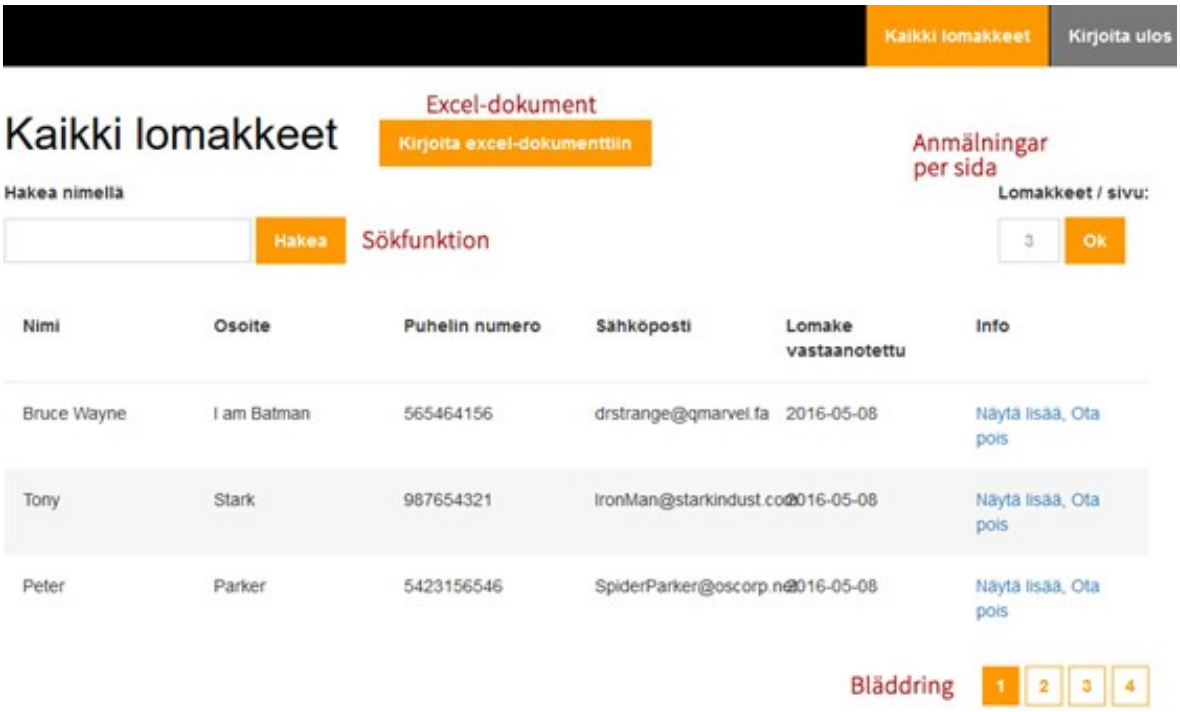
Kodexempel 24. Kryptering med AES-256-CBC-algoritmen.

```php
<?php
$iv = substr("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec imperdiet lacinia pellentesque. Ut vestibulum sapien eget dolor hendrerit, id porttitor enim lacinia.", 0, 16);
$key = "This is a key";
$driver_name = "Thai Nguyen";

// Kryptera förarens namn
$encrypted_name = openssl_encrypt($driver_name, "AES-256-CBC", $key, 0, $iv);
// $encrypted_name är 4Ph112bXP8ErVmX7MQBh0Q==

// Konvertera den krypterade datan till en textsträng som kan lagras
//felfritt I databasen med hänsyn till PDO:s prepare()-funktion
$encrypted_name = base64_encode($encrypted_name);

// Konvertera tillbaka textsträngen till det krypterade värdet
$decrypted_name = base64_decode($encrypted_name);

// Återskapa förarens namn
$decrypted_name = openssl_decrypt($decrypted_name, "AES-256-CBC", $key, 0, $iv);?>
```

4.4 Administrationspanelen

Panelens främsta uppgift var att lista alla inskickade anmälningar och att ge klienten möjlighet att spara anmälningarna i Excel-format. Gränssnittet hos den här panelen visades enbart på finska, eftersom personen som skulle använda har det som sitt modersmål enligt uppdragsgivaren.

Listan över alla anmälningar var det första som visades efter inloggning. Anmälningarna listades med deltagarens namn, adress, telefonnummer, epost och anmälningsdatum. Listan sorterades enligt den tidsordning som anmälningarna skickades in.



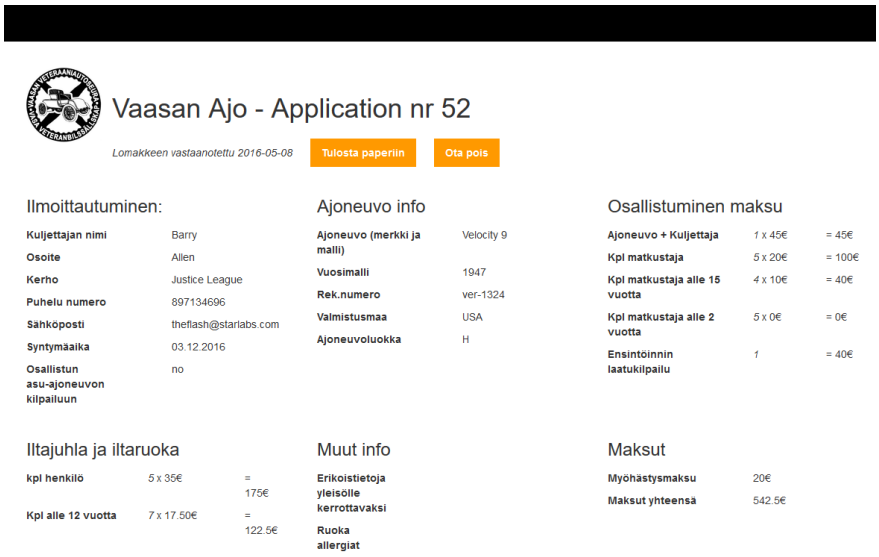
Figur 24. Lista över anmälda deltagare.

Användargränssnittet kan ses i figur 24. Det programmerades för att förenkla arbetet för klienten. Antalet sidor i bläddringsfunktionen bygger på en SQL-sats, som beror helt på antalet anmälningar som klienten vill visa och på det totala antalet anmälningar som finns i databasen. Sökfunktionen baserade sig på kodexempel 21, som söker i databasen efter namn som liknar sökordet.

Kodexempel 21. SQL-satsen som för sökfunktionen

```
<?php $sql = "select driver_name, application_id, address, phone_number, email, received from applications
where driver_name like '%".$term."%'"; ?>
```

När man klickade på "Näytä lisää" i listan så visades alla uppgifter som den valda deltagaren skickat från anmälningsformuläret. Se figur 25.



Figur 25. Profilsida för en deltagare.

Från profilsidan var det möjligt att skriva ut uppgifterna. I det här fallet triggades webbläsarens inbyggda "skriv ut"-funktion genom att klicka på "Tulosta paperiin", där JavaScript-funktionen window.print() aktiverades vid klick.

```
<a href="javascript:;" class="print-button" onclick="window.print()">Tulosta paperiin</a>
```

Utskriftens utseende bestämdes av CSS-kod, som hämtades med attributet "media" för att förklara för webbläsaren att CSS-koden skulle gälla för utskrift.

```
<link rel="stylesheet" href="css/style-application-print.css" media="print" />
```

Klienten kunde ta bort en anmälning genom att klicka på länken ”Ota pois”, och sedan bekräfta det.

Det här var emellertid en säkerhetsrisk ifall klienten hade valt att göra något annat på internet, t.ex. gått in på en extern sida som automatiskt får webbläsaren att ladda in ”bekräfta”-länken, t.ex. via HTML-taggar (iframe, img, link, script osv.).

För att stoppa denna möjlighet så tillämpades samma ”sessionsnyckel”-metod som användes för att identifiera anmälningsformuläret.

Kodexempel. Slumpvald nyckel tillagd på ”delete”-länken.

```
<?php
        $_SESSION["auth_del"] = bin2hex( openssl_random_pseudo_bytes(32) );
?>
<a href="delete.php?application=[anmälningens id]&choice=yes&auth_del=<?php echo
$_SESSION["auth_del"]; ?>">Poista</a>
```

Varje gång klienten valde att ta bort något så genererades en nyckel som tillämpades på bekräftelselänken. Om nyckeln inte motsvarade värdet som genererats på servern så utfördes ingen borttagning vid anrop.

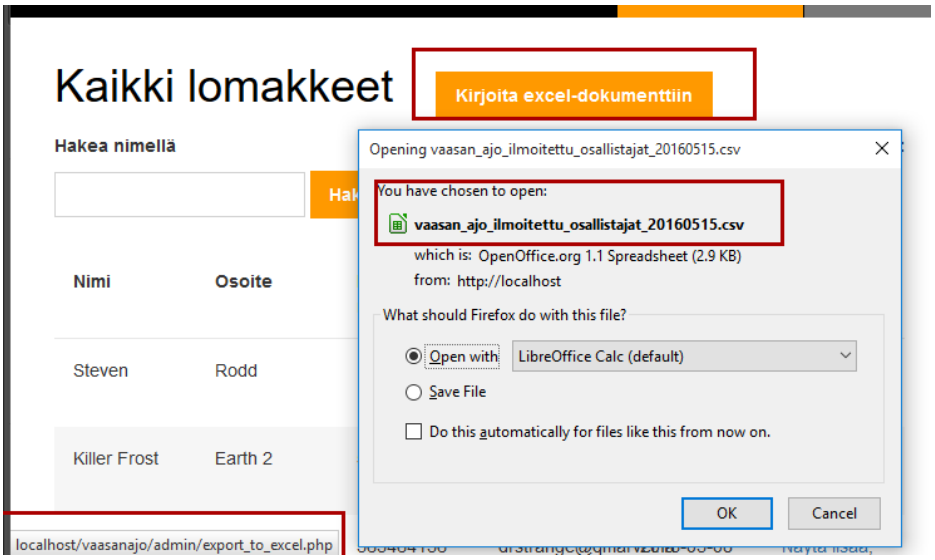
4.4.1 Export av anmälningar till Excel

Tanken med den här metoden var att skapa ett CSV-dokument, som kunde laddas ned via administrationspanelen. Det här gjordes i ett PHP-dokument, där en http-header för CSV-filen ställdes in enligt kodexempel 22.[30]

Kodexempel 22. Ändring av filnamn, teckenkodning och filtyp för CSV-filen

```
<?php
	$filename = "vaasan_ajo_ilmoitettu_osallistajat_" . date('Ymd') . ".csv";
	header("Content-Encoding: UTF-8");
	header("Content-Disposition: attachment; filename=" . $filename);
	header("Content-Type: text/csv;");
?>
```

CSV-filens teckenkodning ställdes in till UTF-8 för att stödja de nordiska bokstäverna ÅÄÖ. Dess filnamn döptes enligt $filename-variabeln, som alltid inkluderade datum enligt den dag man laddade ned filen. Filtypen ställdes in till "text/csv".



Figur 26. "Kirjoita excel-dokumenttiin" leder till Excel-dokumentet via PHP.

För att lägga in innehållet från databasen så användes en så kallad ”filström” för att låta PHP skriva till CSV-filen. Denna skapades enligt kodexempel 23.

Kodexempel 23. ”Filström” öppnas och används för att lägga in data i CSV-filen

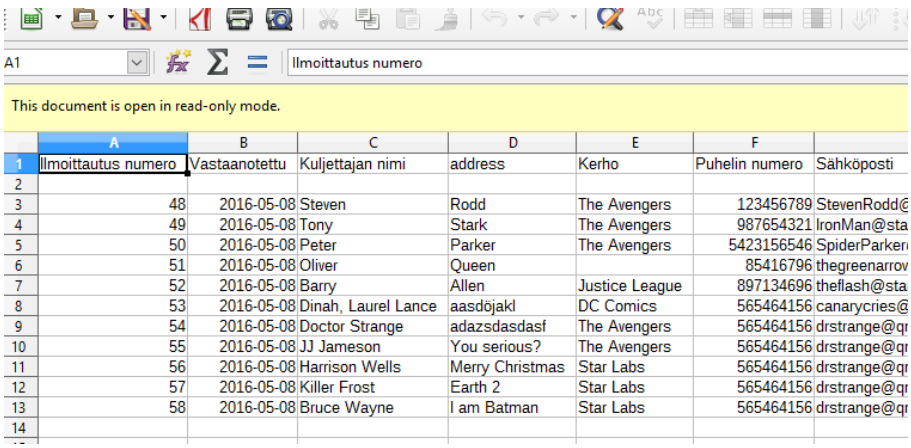
```
<?php
$output = fopen("http://output", 'w');
fputcsv($out, $colnames, ',', '"');
fputcsv($out, array_values( array() ), ',', '"');

while(false !== ($row = $action->fetch(PDO::FETCH_ASSOC)){
        $row["driver_name"];

        /* Behandla varje rad skilt för sig */

        /* Lägg till rad i CSV-filen */
        fputcsv($out, array_values($row), ',', '"');
}
fclose($output);
?>
```

Exemplet lägger in data rad för rad i CSV-filen med funktionen fputcsv($filestream, $column_name) efter att filströmmen öppnats. I ett textdokument skulle en inskriven rad kunna se ut som följande:

"Ilmoittautus numero",Vastaanotettu,"Kuljettajan nimi",address,Kerho,...

Kolumnernas namn lades in först, följt av en tom rad. SQL-satsen för datahämtningen bestämdes på förhand, för att sedan läggas in i ett databasanrop.

Iterationssatsen i kodexempel 23 skriver in data i filen vartefter dataraderna hämtas från databasen. Filströmmen stängdes efter att all data skrivits in.



A1 Ilmoittautus numero

This document is open in read-only mode.

	A	B	C	D	E	F	
1	Ilmoittautus numero	Vastaanotettu	Kuljettajan nimi	address	Kerho	Puhelin numero	Sähköposti
2							
3	48	2016-05-08	Steven	Rodd	The Avengers	123456789	StevenRodd@
4	49	2016-05-08	Tony	Stark	The Avengers	987654321	IronMan@sta
5	50	2016-05-08	Peter	Parker	The Avengers	5423156546	SpiderParker
6	51	2016-05-08	Oliver	Queen		85416796	thegreenarrov
7	52	2016-05-08	Barry	Allen	Justice League	897134696	theflash@sta
8	53	2016-05-08	Dinah, Laurel Lance	aasdöjakl	DC Comics	565464156	canarycries@
9	54	2016-05-08	Doctor Strange	adazsdasdasf	The Avengers	565464156	drstrange@q
10	55	2016-05-08	JJ Jameson	You serious?	The Avengers	565464156	drstrange@q
11	56	2016-05-08	Harrison Wells	Merry Christmas	Star Labs	565464156	drstrange@q
12	57	2016-05-08	Killer Frost	Earth 2	Star Labs	565464156	drstrange@q
13	58	2016-05-08	Bruce Wayne	I am Batman	Star Labs	565464156	drstrange@q
14							

Figur 27. CSV-tabellen med efterfrågade uppgifter.

4.4.2 Autentisering och hantering av cookie-manipulation

För varje administrationskonto i databasen lagrades ett ”hashat” lösenord och ett unikt ”salt”-värde. Själva lösenordet finns inte någonstans i systemet, varken i databasen eller i själva systemkoden.

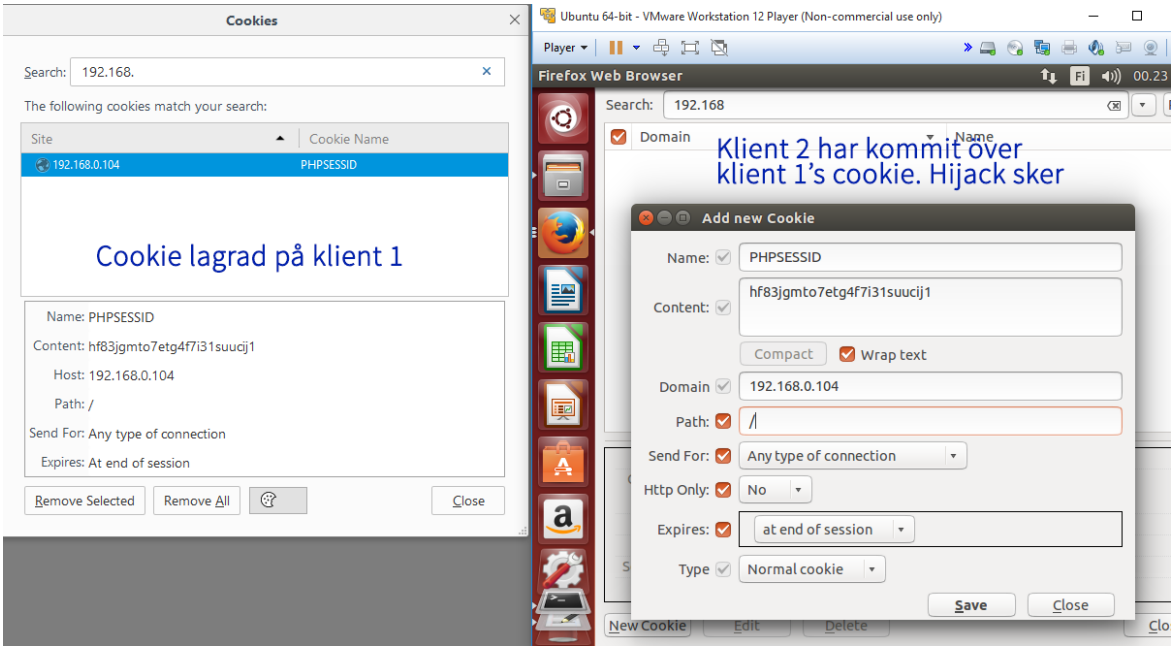
Vid varje inloggningsförsök skapades ett tillfälligt ”hash”-värde med hjälp av det inskrivna lösenordet och det lagrade saltet, som jämfördes med det hashade lösenordet för användarkontot som berörs. Om ”hash”-värderna inte stämde överens med varandra så nekades inloggning. Se kapitel 2.5.4.

Kodexempel 24. Jämförelse av två ”hash”-värden för att avgöra om ett lösenord är giltig.

```
$stored_salt = "En slumpad värde baserat på något"; //Salt hämtad från databasen
$stored_hash = "c07d6c79b0456782c55c32280d3c5f301f7efeca"; /* Hash hämtad från databasen */

function check($password, $salt, $hash){
	$login_hash = sha1($password, $salt); /* Skapa hash från ett inputlösenord och det lagrade saltet */
	if($login_hash === $hash){ /* Jämför det lagrade saltet med inloggningens salt */
		return true; /* Lösenordet korrekt */
	}
	return false; /* Lösenord inkorrekt */
}

/* Inloggningsförsök */
if( check($password, $stored_salt, $stored_hash) == true){
	login();
} else {
	die("Fel lösenord!");
}
```

Saltet tilldelades ett slumpvalt värde vid skapandet av varje användarkonto.

Om ett inloggningsförsök lyckades så tilldelades klienten en sessionsvariabel som gav åtkomst till administrationspanelen. I detta skede vidtogs åtgärder för att förhindra stöld av samma session som klienten loggat in med.



Figur 28. En virtualmaskin som använder samma sessionsuppgifter som tilldelats arbetsdatorn.

Fokus lades på att försvåra för en avlyssnare att få tillgång till giltiga sessionsidentifikationer (SID). Varför just SID prioriterades kan man se i figur 28, eftersom administrationspanelen gav tillgång till deltagarnas personliga uppgifter.

Figur 29. Aktivitetsdiagram över administrationspanelens session.

I figur 29 tilldelas arrangören en ny SID vid både inloggning och utloggning, medan den som tidigare använts blev ogiltig. Det här är ett sätt att förhindra Session Fixation ifall någon skulle ha lyckats skapa en manipulerad cookie i klientens webbläsare, exempelvis genom skadliga tilläggsprogram.

Kodexempel 25. Generera ny SID

```
<?php
session_start();
if(!isset($_SESSION["session_generated"])){
        session_regenerate_id(true);
        $_SESSION["session_generated"] = 1;
}
?>
```

En annan åtgärd som vidtogs var att automatiskt logga ut klienten om denne inte gjort något under en viss tid, för att förhindra att samma SID skulle vara aktiv för länge i onödan.

4.5 Testanvändning och korrigering

Systemet testkördes en sista gång innan lansering. Under den här fasen upptäcktes visningsfel som berörde Bootstrap, som snabbt korrigerades. Sessionerna kontrollerades för att garantera att de fungerade, och att de hade någorlunda skydd.

Vidare testades felhanteringen i anmälningsformuläret och dess granskningssida, för att säkerställa att dessa kunde hantera specialtecken oavsett om dessa skrivits in av misstag eller med flit av klienten. I de grafiska felmarkeringarna hittades ett par variabelfel som rättades.

Under arbetet hade inte uppdragsgivaren kommit med varken nya eller specifika önskemål som innebar drastiska förändringar, så inget speciellt behövdes när det praktiska utförandet närmade sig sitt slut.

Emellertid slogs databastabellerna för anmälningsformuläret ihop till en enda tabell, eftersom det inte längre fanns något som motiverade fem tabeller när arbetet nådde stadiet med insättning av data i Excel-tabellen.

5 Resultat

Arbetet resulterade i ett heltäckande system som uppfyllde uppdragsgivarens önskemål. Ett system med en publik del, där informationssidan, anmälningsformuläret och språkbyten ingick, och en administrationspanel varifrån Excel-dokumentet kunde laddas ned.

Utöver det så tillfördes ett par funktioner till anmälningsformuläret för att göra ifyllningen lättare. Ett bläddringssystem och sökfunktion lades till i administrationspanelen för att förenkla arbetet för de personer som skulle hantera de inskickade uppgifterna. Detta system fungerade även på mobila enheter.

Systemet lades upp i sin helhet på internet den 1 april 2015 och stängdes ned den 31 maj 2015 efter att anmälningstiden löpt ut och efter att Excel-dokumentet hade hämtats. Vaasan Ajo 2015 ägde rum ett par veckor därefter.

5.1 Diskussion och kritisk granskning

Det här var i princip ett arbete där jag fick fria händer, så länge som det byggda systemet uppfyllde uppdragsgivarens krav.

Under arbetet dök det ständigt upp småfrågor i mina tankar. Hur hanterade man evenemang på internet? Vad betydde säkerhet och vilka skulle kunna skada evenemanget? Vilka tekniker var lämpligast? På vilket sätt kan man göra systemet mera användbar, och slutligen, vad måste prioriteras?

Ibland gäckades jag även av frågan om jag verkligen gjorde fel då jag valde att inte arbeta med CodeIgniter. Men som nykomling och obekant inom det ramverket så drog jag slutsatsen av att det inte gick att lära sig att arbeta med det med hjälp av detta examensarbete. Uppdragsgivaren ville ha ett säkert resultat, och jag ville inte riskera att inte uppnå det under den korta tid som det praktiska utförandet kunde pågå.

Istället lades vikt på systemets användbarhet ur det grafiska och ur det funktionella perspektivet för att uppfylla uppdragsgivarens önskemål. Sedan tillades grundläggande säkerhetsskydd på grund av personuppgifterna som systemet behandlade.

Anmälningsformuläret var funktionell för ifyllaren, administrationspanelen hade ett par praktiska funktioner och Excel-dokumentet såg fräscht ut. Systemet anpassade sig också klanderfritt till de enheter som klienten använde.

Trots att det gjordes en handfull onödiga saker under examensarbetet, så är jag i slutändan nöjd med resultatet. Uppdragsgivaren var också nöjd, vilket var det allra viktigaste och även ett tecken på att arbetet verkligen slutförts.

5.2 Möjlighet till vidareutveckling

Vaasan Ajo 2015 var en engångshändelse, och arbetet gjordes efter detta faktum. Dock skulle det inte vara omöjligt att återanvända samma system i framtiden om föreningen skulle behöva en ny nätbaserad blankett eftersom nya fält kan läggas till om det skulle behövas.

MVC-modellen gör det också lätt att lägga till nya sidor i den publika delen av systemet och även inne i själva administrationspanelen, vilket skapar utrymme och möjlighet att utvidga med ytterligare funktionalitet - antingen för att skapa ett ansikte utåt på nätet, eller för att få större kontroll inne i administrationspanelen.

Källförteckning

[1] *What is Waterfall model- advantages, disadvantages and when to use it?*. [Online] http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/ [hämtat: 4.4.2016]

[2] Taylor, J. *Using Both The Waterfall And Agile Methods* [Online] http://www.statslice.com/using-both-the-waterfall-and-agile-methods [hämtat: 4.4.2016]

[3] Taylor, J., 2013. *Using Both the Waterfall and Agile Methods*. [Online] http://www.dataversity.net/using-both-the-waterfall-and-agile-methods/ [hämtat: 4.4.2016]

[4] Agile Aliance. *12 Principles Behind the Agile Manifesto*. [Online] https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/ [hämtat: 4.4.2016]

[5] Pastor, P., 2010. *MVC for Noobs*. [Online] http://code.tutsplus.com/tutorials/mvc-for-noobs--net-10488 [hämtat: 4.4.2016]

[6] Rouse, M., 2011. *model-View-controller (MVC)*. [Online] http://whatis.techtarget.com/definition/model-view-controller-MVC hämtat [hämtat: 4.4.2016]

[7] Oracle. *A Relational Database Overview*. [Online] https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html [hämtat: 10.5.2016]

[8] W3Schools. *PHP 5 Sessions*. [Online] http://www.w3schools.com/php/php_sessions.asp [hämtat: 10.5.2016]

[9] Owasp, 2014. *Session hijacking attack*. [Online] https://www.owasp.org/index.php/Session_hijacking_attack hämtat [hämtat: 10.5.2016]

[10] Owasp, 2014. *Session fixation*. [Online] https://www.owasp.org/index.php/Session_fixation [hämtat: 10.5.2016]

[11] DuPaul, N. *Cross-Site Scripting (XSS) Tutorial: Learn About XSS Vulnerabilities, Injections and How to Prevent Attacks*. [Online] http://www.veracode.com/security/xss [hämtat: 10.5.2016]

[12] PHP.NET. *SQL Injection*. [Online] http://php.net/manual/en/security.database.sql-injection.php [hämtat: 10.5.2016]

[13] Irizarry, A., 2014. *Cross-Site Request Forgery (CSRF) in Plain English*. [Online] https://www.tinfoilsecurity.com/blog/what-is-cross-site-request-forgery-csrf [hämtat:10.5.2016]

[14] Conviso. 2014. *Worst and best practices for secure password storage*. [Online] http://blog.conviso.com.br/worst-and-best-practices-for-secure-password-storage/ [hämtat: 10.5.2016]

[15] Holbreich.org, A. 2016. *Symetric-key cryptography*. [Online] http://alexander.holbreich.org/symmetric-key-cryptography/ [hämtat: 11.5.2016]

[16] Wikipedia. *LAMP (software bundle).* [Online] https://en.wikipedia.org/wiki/LAMP_%28software_bundle%29#WAMP [hämtat: 12.5.2016]

[17] W3Techs. *Usage statistics and market share of Apache for websites*. [Online] http://w3techs.com/technologies/details/ws-apache/all/all [hämtat: 12.5.2016]

[18] Polepeddi, L., 2013. *Relational Databases for Dummies*. [Online] http://code.tutsplus.com/tutorials/relational-databases-for-dummies--net-30244 [hämtat: 12.5.2016]

[19] W3C. *HTML & CSS*. [Online] https://www.w3.org/standards/webdesign/htmlcss [hämtat: 12.5.2016]

[20] Mozilla Developer Network. *JavaScript basics*. [Online] https://developer.mozilla.org/en-US/Learn/Getting_started_with_the_web/JavaScript_basics [hämtat: 12.5.2016]

[21] Wikipedia. *JSON*. [Online] https://en.wikipedia.org/wiki/JSON [hämtat: 12.5.2016]

[22] PHP.NET. *What is PHP?*. [Online] http://php.net/manual/en/intro-whatis.php [hämtat: 12.5.2016]

[23] Padron-McCarthy, T., 2005. *Introduktion till frågespråket SQL*. [Online] http://www.databasteknik.se/webbkursen/sql/ [hämtat: 13.5.2016]

[24] Getbootstrap.com. *Designed for everyone, everywhere*. [Online] http://getbootstrap.com/

[25] jQuery, 2015. *About jQuery*. [Online] https://learn.jquery.com/about-jquery/

[26] jQuery user interface. *Datepicker Widget*. [Online] http://api.jqueryui.com/datepicker/#option-dateFormat [hämtad: 14.5.2016]

[27] CodeIgniter. *Welcome to CodeIgniter.* [Online] http://www.codeigniter.com/user_guide/general/welcome.html

[28] W3C. *Markup Validation Service*. [Online] https://validator.w3.org/ [hämtad: 14.5.2016]

[29] Getbootstrap.com. *Modals.* [Online] http://getbootstrap.com/javascript/#modals [hämtad: 14.5.2016]

[30] The Art of Web. *PHP: Exporting Data to Excel*. [Online] http://www.the-art-of-web.com/php/dataexport/ [hämtad: 14.5.2016]

Vaasa-ajo lauantaina 13.6.2015 Ilmoittaut[illegible]islomake

Osallistuva ajoneuvo: henkilöauto, hyötyajoneuvo, moottoripyörä, mopedi, polkupyörä, traktori tai muu hidas ajoneuvo. Museokatsastetut tai yli 30 vuotta vanhat ajoneuvot voivat osallistua.

Kuljettajan nimi: Click here to enter text.

Osoite: Click here to enter text.

Kerho: Click here to enter text.

Puh: Click here to enter text. Sähköposti: Click here to enter text.

Syntymäaika: Click here to enter text. Kyydissä henkilöä: Choose an item.

Ajoneuvo (merkki ja malli): Click here to enter text.

Vuosimalli: Click here to enter text. Rek.numero: Click here to enter text. Valmistusmaa: Click here to enter text.

Ajoneuvoluokka: ABC ☐, DE ☐, F ☐, M ☐, U ☐, N ☐, H ☐

Osallistun asu-ajoneuvo kilpailuun ☐ En osallistu ☐

Erikoistietoja yleisölle kerrottavaksi: Click here to enter text.

Ruoka-allergiat: Click here to enter text.

Osallistumismaksut: Ajoneuvo + Kuljettaja .....= 45 €

Click here to enter text. kpl matkustaja a' 20 € .....= Click here to enter text. €

Click here to enter text. kpl lapsi alle 12 vuotta a' 10 € .....= Click here to enter text. €

Click here to enter text. kpl lapsi alle 2 vuotta ilmaiseksi

Entisöinnin laatukilpailu: ☐ 40 €/ajoneuvo.....= Click here to enter text. €

Iltajuhla ja iltaruoka a' 35 €/henkilö, lapset alle 12 v. 17,50 €.

Click here to enter text. kpl henkilö a' 35 € .....= Click here to enter text. €

Click here to enter text. kpl alle 12 vuotta a' 17,50 € .....= Click here to enter text. €

Maksut yhteensä = Click here to enter text. €

Osallistumismaksuun kuuluvat: Ajomerkki, ajomateriaali, aamukahvi, lounas ja iltapäiväkahvi.

Maksu tilille: IBAN FI 66 5672 6020 0169 71 Eräpäivä: 11.05.2015

Ilmoittautuminen, viimeistään 11.5.2015. Jälki-ilmoittautuminen: 31.5.2015 + 20 €/ajoneuvo.

Täytä tämä lomake ja lähetä se:
vaasaajot@gmail.com, **tai osoitteeseen Vaasan Veteraaniautoseura ry. PL 127, 65101 VAASA**