VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Daniel Westerlund

Ärendehanterare för intranät

Case Creamarketing

Företagsekonomi och turism
2014

VASA YRKESHÖGSKOLA

Utbildningsprogrammet för informationsbehandling

ABSTRAKT

Författare	Daniel Westerlund
Lärdomsprovets titel	Ärendehanterare för intranät – Case Creamarketing
År	2014
Språk	svenska
Sidantal	56 + 13
Handledare	Kenneth Norrgård

Avsikten med detta lärdomsprov är att utveckla ett användbart och funktionellt system för uppdragsgivaren där kunder kan skicka in ärenden. Ärendehanteraren delas upp i tre delar: felrapportering, feedback och feature request. Lärdomsprovet bygger på en layout som planerats i ett tidigare projektarbete som inkluderade forskning över olika användargränssnitt.

Detta lärdomsprov kommer att förklara olika metoder, programmeringsspråk och verktyg som har använts vid utvecklingen. Lärdomsprovet skall ge en bra översikt om funktionaliteten och idén bakom olika lösningar i ärendehanteraren. Ärendehanteraren är utvecklad med HTML, CSS, PHP, JavaScript och MySQL.

Lärdomsprovets resultat är en produkt som uppdragsgivaren kan använda i sitt interna nätverk för att underlätta kommunikationen mellan kunderna och uppdragsgivaren.

Ämnesord Ärendehanterare, HTML, CSS, PHP, JavaScript

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Utbildningsprogrammet för informationsbehandling

ABSTRACT

Author	Daniel Westerlund
Title	Errand manager for intranet
Year	2014
Language	Swedish
Pages	56 + 13
Name of Supervisor	Kenneth Norrgård

The purpose of this thesis is the development of a functional and user friendly errand manager for the company Creamarketing. The errand manager is divided in three parts: error reporting, feedback and feature request. The thesis is built on a template developed in a former project researching graphical user interfaces.

This thesis will explain different methods, programming languages and tools used in the development. The thesis will give a comprehensive overview of the functions and ideas used as solutions in the errand manager. The errand manager is developed by using HTML, CSS, PHP, JavaScript and MySQL.

The result of this thesis is a product that can be used in the company's intranet to ease the communication between the company and clients.

Keywords Errand manager, HTML, CSS, PHP, JavaScript

INNEHÅLL

FÖRTECKNING ÖVER FIGURER OCH TABELLER

FÖRTECKNING ÖVER BILAGOR

ORDLISTA OCH FÖRKLARNINGAR

GUI – Graphical User Interface – Grafiskt användargränssnitt

Ett grafiskt användargränssnitt är ett hjälpmedel för att användare skall kunna kommunicera med olika program. GUI är ofta mycket simpelt designat så att det skall vara lätt för användaren att använda gränssnittet. Windows operativsystem är ett bra exempel på ett grafiskt användargränssnitt. Det användaren ser på skärmen är grafiska användargränssnittet som hjälper användaren att använda datorn.

Pixel

En pixel är en del av grafiken som visas av skärmar. Det är den minsta möjliga delen som går att visas. Inom webbdesign används pixel som ett mått för att berätta t.ex. om höjd och bredd.

CMS

Content Management System – ett verktyg för att hantera webbsidor.

Webbläsare

En webbläsare är ett program som används för att komma åt HTML kodade sidor. Några exempel på webbläsare: Chrome, Firefox, Safari, Internet Explorer och Opera.

Källkod(öppen)

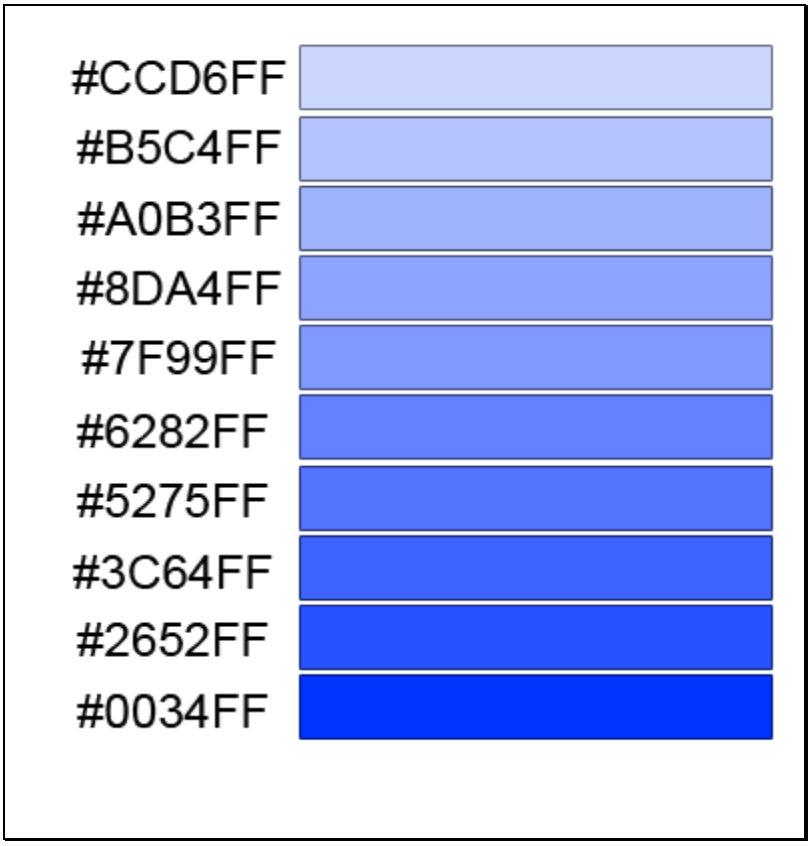
Källkoden är det som programmerare skriver då de skapar IT-produkter. En källkod kan skrivas på flera olika programmeringsspråk. En öppen källkod ger rättigheter för utomstående personer att se in i produkten och göra egna ändringar samt kopiera koden och sprida det vidare.

Användbarhet

Med användbarhet menar man användarvänligheten i en produkt. För att uppnå hög användbarhet kan man ställa sig frågor som t.ex. Hur lätt har användaren att använda produkten samt vilken nytta har användaren av produkten?

Hex färger

Alla färger har en motsvarande färg i en hex kod som används speciellt vid programmering. Hex färger börjar alltid med ”#” och innehåller sex tecken för att identifiera en specifik färg. Det finns flera nyanser av färger och de kan även hittas i hextabellen som presenteras i figur T1.



Figur T1. Olika nyanser av färger kan även hittas i en hextabell.

W3C-standard

Allmänna standarder för att skapa webbsidor av hög kvalité.

IF-sats

En if-sats är en fråga ställt till funktionen som kräver att funktionen skall fylla ett visst kriterium för att en handling skall ske. Ett exempel på detta presenteras i figur T2.

```
<% if Children %>
        Hello Children
<% end_if %>
```

Figur T2. Handlingen sker då ett element har child-element, d.v.s. underliggande element.

Förälderelement

Då ett element har underliggande element, blir det ett ”förälderelement”. Se exemplet i IF-sats.

Case

Ordet Case kan betyda flera saker och är i detta sammanhang olika fall som kunde ske angående ärendehanteraren. Ett fall kunde vara t.ex. en kund med olika krav för ärendehanteraren.

Front-end och Back-end

Front-end är den synliga sidan av en webbsida då back-end igen är ”motorn” bakom det. Back-end refereras ofta med CMS-verktyget.

1 INLEDNING

1.1 Bakgrundsinformation

Uppdragsgivaren i detta lärdomsprov är det företaget som jag utförde min grund- och yrkesinriktade praktik för. Jag hade en del planeringar angående lärdomsprovet, men beslöt mig sedan att göra ett arbete som är till nytta för någon. Under praktiken diskuterade jag med verkställande direktören i företaget om lärdomsprovet och fick veta att företaget har ett projekt som kunde intressera mig. Efter att vi gått igenom innehållet för projektet, visste jag att detta var något jag absolut kunde välja som ämne till lärdomsprovet. Vi diskuterade med uppdragsgivaren vid flera tillfällen och beslöt oss för att jag skall utföra lärdomsprovet för företaget.

1.2 Uppdragsgivare

Företaget utvecklar webbsidor och andra produkter inom området. Creamarketing har grundats år 1998 i Vasa. Detta ämne är viktigt för uppdragsgivaren eftersom det inte finns i dagens läge ett system som kunde sköta om ärenden. Företaget skall även utveckla ett internt nätverk i samband med ärendehanteraren.

1.3 Innehåll

Innehållet är en redogörelse över hur ärendehanteraren utvecklats och vilka lösningar och metoder använts. Lärdomsprovet är indelat i ordningsföljd i olika faser. De olika programmeringsspråken är individuellt presenterade och metoden för programmeringen har även presenterats. Det finns en lista över terminologin som kan användas för att underlätta förståelsen av detta lärdomsprov.

1.4 Syfte

Syftet med detta lärdomsprov är att utveckla en funktionell och användarbar lösning för att företaget ifråga lätt skall kunna kommunicera med kunder via ett internt nät. Ärendehanteraren skall vara planerad så att det prioriterar kunden och dess förmåga att så lätt som möjligt kunna använda systemet.

1.5 Mål

Målet är att utveckla en färdig produkt som kan användas av uppdragsgivaren. Resultatet skall vara felfritt, av hög kvalitet och så användbart som möjligt. Ärendehanteraren kommer att fungera som ett verktyg mellan uppdragivaren och kunderna för att underlätta kommunikationen mellan parterna, med detta som målsättning måste ärendehanteraren även vara säker och ha en bra och funktionell design.

2 PROGRAMMERINGSTEKNIK

Detta kapitel beskriver noggrant de programmeringsspråk, program och metodiker som använts vid utvecklingen av ärendehanteraren. Kapitlet beskriver en del terminologi och övriga förklarningar som kommer att användas senare och finns som ett hjälpmedel för att underlätta läsningen.

2.1 Programmeringsspråk

2.1.1 HTML

HTML står för HyperText Markup Language och är standarden av märkspråk vid utvecklingen av webbsidor. Sedan webben har utvecklats har det publicerats fem versioner av HTML, då den nyaste är HTML5. World Wide Web Consortium (W3C) övervakar utvecklingen av HTML. HTML är ett märkspråk, d.v.s. strukturen av HTML bygger sig på taggar eller element. Dessa märken eller ”taggar” identifierar ett specifikt element som används för en viss del av webbsidan. Några exempel på dessa taggar är <head>, <body> och <footer> (Duckett, Larsen 2013, 42-43). Vid utvecklingen av HTML baserade sidor måste man först deklarera vilken version av HTML man använder, med den nyaste HTML5 versionen har deklarationen gjorts först på dokumentet med taggen: <!doctype html>. I figur 1 visas ett exempel på en grundstruktur i ett HTML dokument. Resultatet av exemplet blir en fungerande webbsida som kan visas med en webbläsare. Resultatet visas i figur 2.

```
<!doctype html>
<head> <!-- Head taggen används för att koppla olika dokument och script till sidan,
           innehållet i head syns inte på sidan -->
    <link rel="stylesheet" href="styles.css" type="text/css" />
    <!-- Denna rad kopplar ett CSS dokument i detta HTML dokumentet -->
</head>
<!-- Body är strukturen som används för att visa material på webbsidan. -->
<body>
    <!-- h1 deklarerar att texten inom taggen skall användas som rubrik på sidan -->
    <h1> Hello World! </h1>
    <!-- p deklarerar att texten inom taggen är en paragraf -->
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
    <!-- Alla taggar skall stängas efter de har förfyllt sin uppgift med /tag -->
</body>
<!-- Body stängs här -->
</html>
<!-- HTML stängs här - sidan slutar här -->
```

Figur 1. Denna figur visar grundstrukturen i ett HTML dokument. Raderna med grön text är kommentarer och visas inte på själv webbsidan. För att kommentera ut text i ett HTML dokument används <!-- --> märken med texten innanför. Bilden i figuren är en fullständigt fungerande webbsida och kan öppnas med en webbläsare.

Figur 2. Detta är resultatet för koden i figur 1. Alla taggarna och kommentarerna är gömda från denna sida.

2.1.2 CSS

CSS står för Cascading Style Sheets och används för att sätta in olika stilar på HTML-element. Ett CSS dokument är en skild fil som läses in av HTML dokumentet (se figur 1) för att lätt och smidigt kunna editera utseendet av webbsidan. Då CSS-dokumentet är kopplat till HTML-dokumentet kan man lägga

in stilar på olika element som används på webbsidan med hjälp av t.ex. klasser och id:n.

Skillnaden mellan klasser och id-värden är att klasser kan användas av flera element flera gånger. ID-värdet är menat för att bara användas en gång. Vid figur 3 har elementet <h1> fått klassen HelloText, denna klass kunde man använda på nytt t.ex. vid elementet <p>.

Ett CSS-dokument kan modifiera element på t.ex. följande sätt: ID med markeringen #, klass med ett ”.” (punkt) eller direkt modifiering av ett element. Figur 3 visar exempel på hur man ger klasser och ID värden till olika element i ett HTML-dokument för att sedan kunna modifiera dem via CSS filen.

För att få CSS att fungera måste man sedan ge olika stilar till elementet i CSS filen. Figur 4 visar hur elementet <body> har fått en bakgrundsfärg via id anknytning, elementet <h1> har fått en färg på texten (röd) och alla paragrafer i dokumentet får en blå färg eftersom elementet <p> knyts till blå färg.

Resultatet av HTML-dokumentet och-CSS dokumentet visas i figur 5.

```
<!doctype html>
<head> <!-- Head taggen används för att koppla olika dokument och script till sidan,
           innehållet i head syns inte på sidan -->
    <link rel="stylesheet" href="styles.css" type="text/css" />
    <!-- Denna rad kopplar ett CSS dokument i detta HTML dokumentet -->
</head>
<!-- Body är strukturen som används för att visa material på webbsidan. -->
<body id="HelloBody">
    <!-- h1 deklarerar att texten inom taggen skall användas som rubrik på sidan -->
    <h1 class="HelloText"> Hello World! </h1>
    <!-- p deklarerar att texten inom taggen är en paragraf -->
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
    <!-- Alla taggar skall stängas efter de har förfyllt sin uppgift med /tag -->
</body>
<!-- Body stängs här -->
</html>
<!-- HTML stängs här - sidan slutar här -->
```

Figur 3. I denna figur har elementet <body> fått en id Hellobody och elementet <h1> har fått en klass HelloText.

```
/*ID*/
#HelloBody {background:#CCC}
/*Denna rad ger stil åt ID:n HelloBody med en grå bakgrund*/
/*CLASS*/
.HelloText {color: red}
/*Denna rad ger stil åt alla element som använder klassen HelloText*/
/*ELEMENT*/
p {color: blue}
/*Denna rad ger stil åt alla paragrafer i HTML dokumentet*/
```

Figur 4. CSS dokumentet har stilar för id värdet HelloBody, klassen HelloText och elementet <p>.

Figur 5. Slutliga resultatet efter de olika elementen har fått stilarna från CSS dokumentet.

2.1.3 PHP

PHP är ett programmeringsspråk som används främst i programmering av dynamiska webbplatser. PHP var ursprungligen en förkortning av Personal Home Page, vilket utvecklades av Rasmus Lerdorf år 1995. År 1997 gjordes stora ändringar i källkoden och namnet byttes samtidigt till Hypertext Preprocessor. År 1998 räknades att över 50 000 webbsidor använde PHP. Sedan början har det kommit ut flera versioner varav den nyaste är PHP 5. (Gilmore, 2005, 1-4)

PHP kräver en installation av källkoden till servern för att fungera. Då man har installerat PHP kan man köra det i HTML-koden. En mycket simpel kod rad för att visa text på webbsidan:

<?php echo "Lärdomsprov";?> Denna kod rad visar på webbläsaren texten: Lärdomsprov.

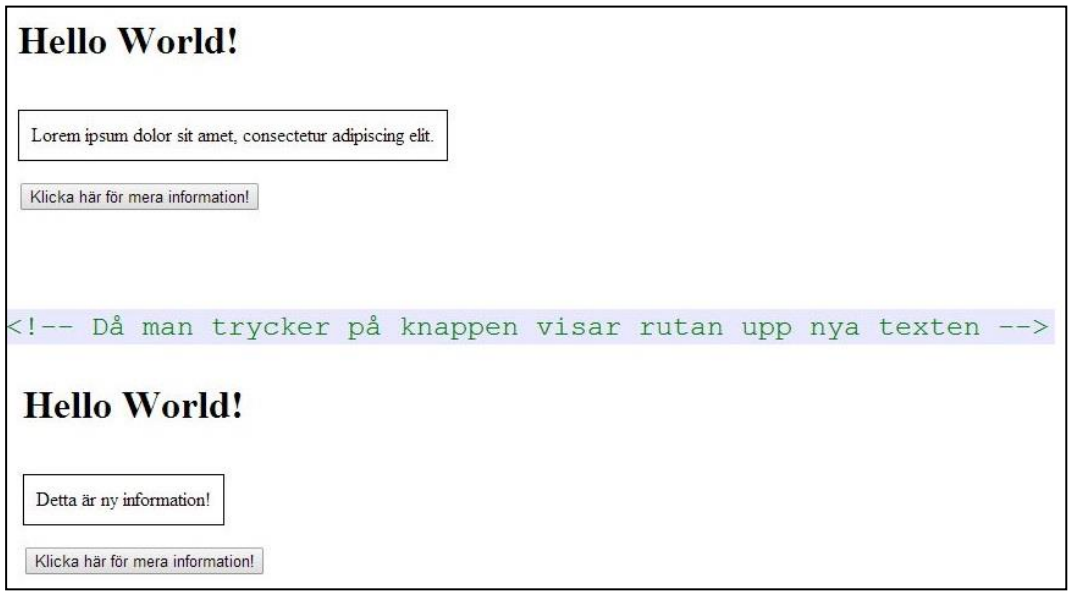
I detta lärdomsprov används PHP för att hantera alla relationer och databaskopplingar.

2.1.4 JavaScript

JavaScript är ett programmeringsspråk som berättar till datorn att den skall göra något. Några exempel på vad man kan använda JavaScript för: byta storlek på ett element, visa eller editera text, byta bakgrunder eller färger, olika slags händelser relaterat till tiden m.m. Datorn sköter alltså den uppgift som står i koden då något händer (t.ex. En knapptryckning) (Wilton, McPeak 2010, 33-34). JavaScript används för att lösa problem som kan uppstå med t.ex. En design eller layout som inte går att bygga upp med bara HTML och CSS. Figur 6 presenterar ett simpelt JavaScript som byter texten i en ruta. Resultatet kan öppnas med en webbläsare och figur 7 visar vad det händer då man trycker på knappen "nyText".

```
<!doctype html>
<head>
    <link rel="stylesheet" href="styles.css" type="text/css" />
    <script>
        function nyText()
        {
            document.getElementById("textbox").innerHTML = "Detta är ny information!";
        }
    </script>
</head>
<body>
    <h1> Hello World! </h1>
    <p id="textbox"> Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
    <button type="button" onclick="nyText()"> Klicka här för mera information!</button>
</body>
</html>

<!-- Då man trycker på knappen visar rutan upp nya texten -->
```

Figur 6. Denna figur visar ett simpelt JavaScript som körs då användaren trycker på knappen ”Klicka här för mera information!”. Resultatet av denna sida visas i figur 7.



Figur 7. Resultatet då användaren trycker på knappen, innehållet i textrutan byter. Detta är ett mycket simpelt script.

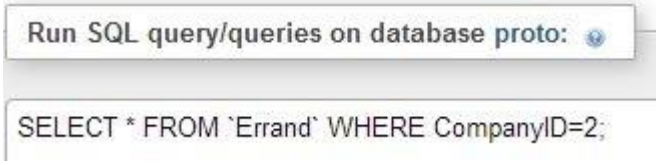
2.2 Program och övriga metoder samt resurser

2.2.1 SQL, MySQL

SQL står för Structured Query Language och används för kommunikation med databaser. SQL använder sig av frågor för att söka information från databasen. (Forta, 2004, 11-16) En simpel fråga kan se ut enligt följande:

SELECT * FROM 'Errand' WHERE CompanyID=2;

Denna fråga söker information från tabellen Errand med data som har CompanyID värde två (2). En tabell är ett register där data sparas strukturerat. (Forta, 2004, 16-17) I figur 8 presenteras en exempel fråga i MySQL och resultatet det ger visas i figur 9.



Figur 8. Detta exempel filtrerar data ur tabellen ”Errand” med CompanyID = 2.

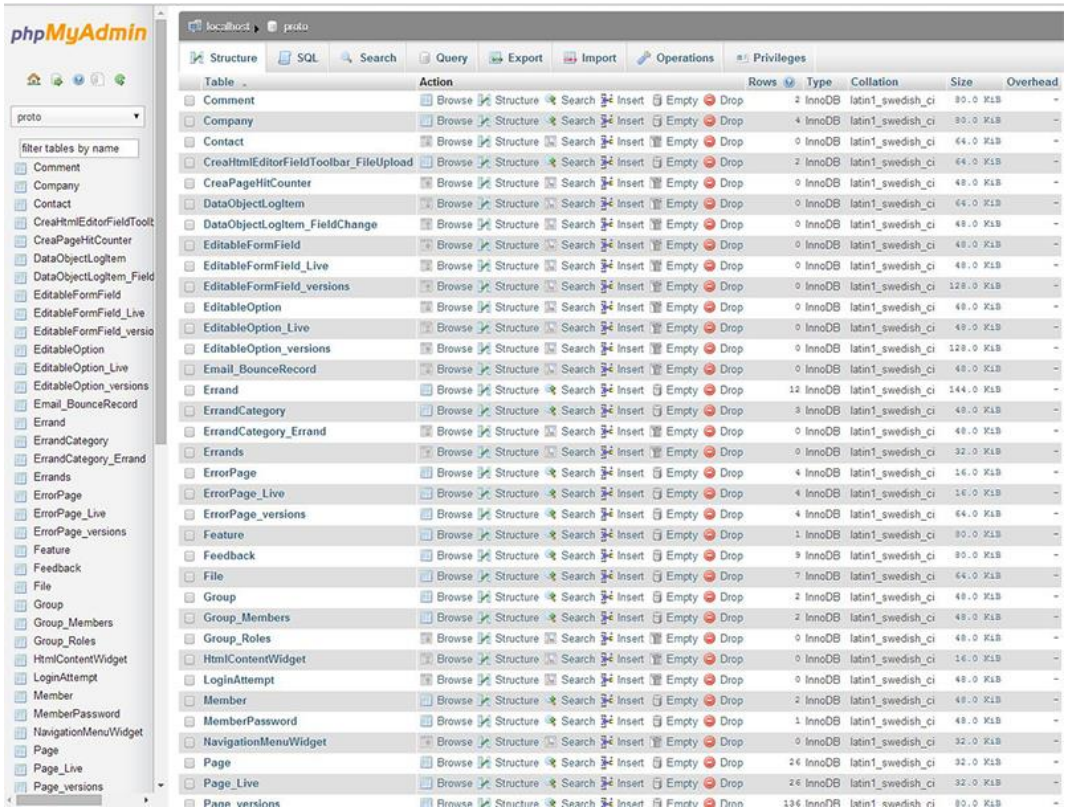
Company	Contactperson	Phone	Type	Website	Description	ErrandPageID	Email	CompanyID
NULL	NULL	NULL	WebAdmin5	NULL	NULL	29	NULL	2
NULL	NULL	NULL	Module	NULL	NULL	29	NULL	2
NULL	NULL	NULL	NULL	NULL	NULL	29	NULL	2
NULL	NULL	NULL	NULL	NULL	NULL	29	NULL	2

Figur 9. Objekt från tabellen ”Errand” med CompanyID värde två.

MySQL är en relationsdatabas som utvecklades av Michael ”Monty” Widenius och David Axmark. I dagens läge är MySQL den populäraste databas som har öppen källkod. Några stora organisationer som använder MySQL: Yahoo!, NASA, Google, Nortel Networks och Cisco. (Gilmore, 2005, 511) MySQL används med andra ord för att spara data på en server. Det finns flera olika databas lösningar men idag är då MySQL ett av de populäraste och används även i detta lärdomsprov. I detta lärdomsprov används phpMyAdmin för att hantera MySQL och mera om phpMyAdmin berättas i nästa stycke.

2.2.2 phpMyAdmin

phpMyAdmin är en webbapplikation som är skriven med PHP. Programmet använder sig även av HTML, CSS och JavaScript. Användargränssnittet är gjort så lätt som möjligt för att kunna användas av personer som eventuellt inte har mycket kunskap inom området. Programmet är planerat för att hantera MySQL databaser och är det populäraste hanteringsverktyget (Delisle, 2008, 8-9). Användargränssnittet är mycket simpelt men väl fungerande och det presenteras i figur 10.



Figur 10. Vy från phpMyAdmin gränssnittet.

2.2.3 Apache

Apache är en webbserver utvecklad i mitten av 90-talet av NCSA (National Center for Supercomputer Applications) och är en av världens populäraste webbservrar (Chopra, 2004, 2-3). Apache används för att uppehålla t.ex. Webbsidor och är gratis att använda. En webbserver hanterar alla de filer som krävs för att webbsidor skall fungera.

2.2.4 SCRUM

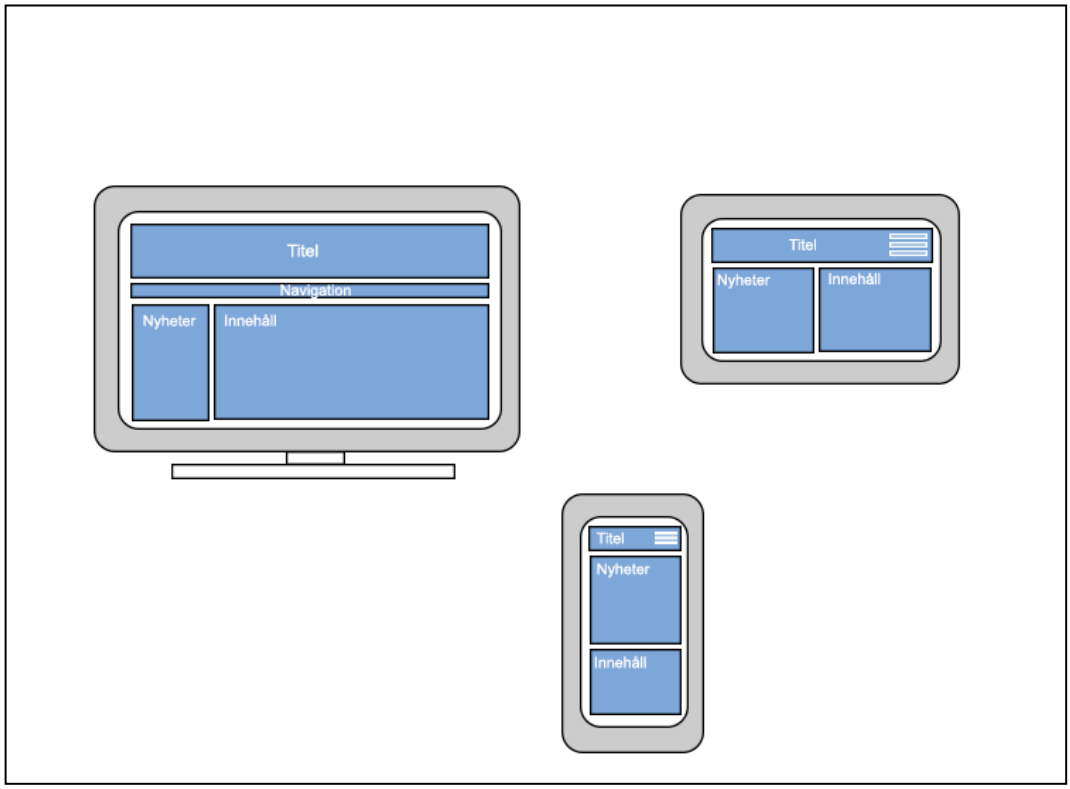
Scrum används för att utveckla och underhålla produkter. Metoden är egentligen ett ramverk med hjälpmedel för produktutveckling. Scrum har använts redan i över 20 år. För att använda sig av Scrum skall det finnas ett scrumteam som har olika aktiviteter, artefakter, regler och roller. Programmeringen utförs i ”Sprintar” d.v.s. Korta sträckor på två veckor då programmeringen sker för att sedan ha ett möte och gå igenom resultaten. (Schwaber & Sutherland 2013) I detta lärdomsprov användes Scrum som en metod för utvecklingsprocessen.

2.2.5 Objekt orienterad programmering (OOP)

PHP är ett objektorienterat programmeringsspråk vilket innebär att det finns objekt som innehåller attribut vilka beskriver objektet. Processer som sköts i OOP kallas för metoder. Ett objekt kan vara t.ex. ”produkt” som innehåller attributen 'pris', 'material' och 'färg'.

2.2.6 Responsiv webbdesign

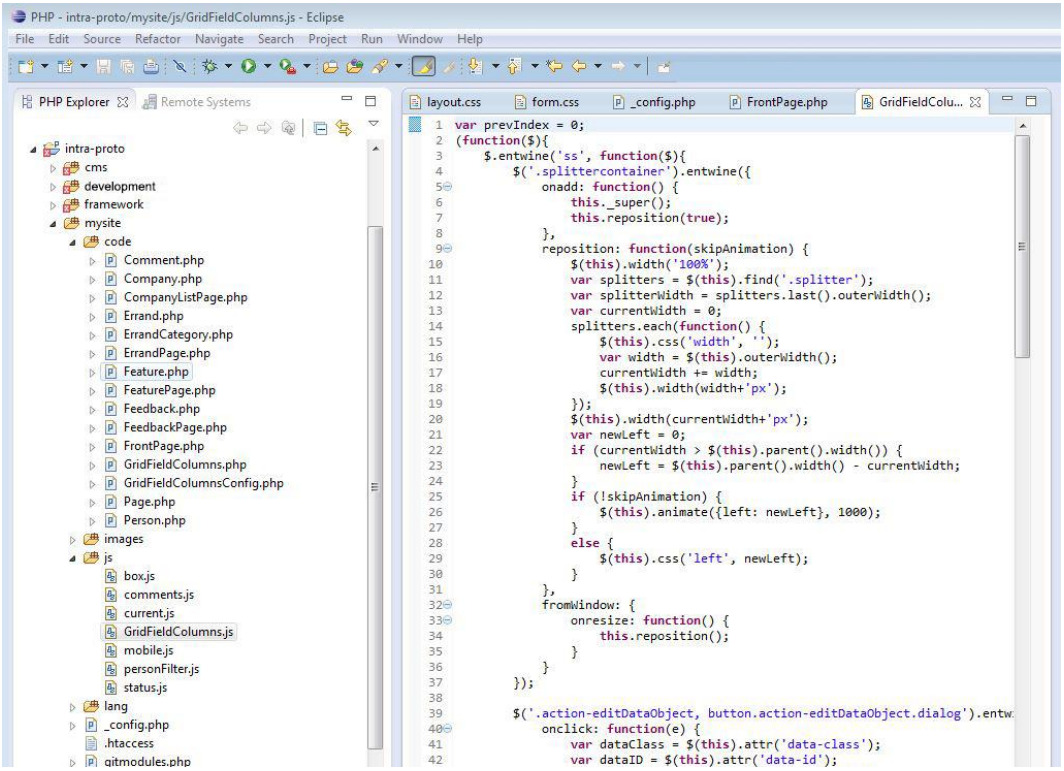
Responsiv webbdesign är en utvecklingsteknik för att anpassa webbsidor att läsa in pixelbrädd på olika apparater för att sedan anpassa webbsidan för skärmen. Responsiv design är inte samma som mobilanpassning, vilket fokuserar på att skapa ett egen mobilanpassad sida (Marcotte, 2011). Figur 11 presenterar hur de olika ”klossarna” d.v.s. materialet på webbsidan rör på sig beroende på hur mycket utrymme de har att använda.



Figur 11. Ett exempel på hur responsiv design kunde se ut. Ifall man skall använda sig av ikoner skall de vara väl kände (t.ex. Navigationen i detta exempel)

2.2.7 Eclipse

Eclipse är en utvecklingsmiljö som kan användas för flera olika programmeringsspråk. Gränssnittet är gjort mycket enkelt att förstå och underlättar utvecklingsprocessen av ett program eller en webbsida. Eclipse är en av de mest kända utvecklingsmiljö programmen, ett annat värt att nämna är NetBeans. Gränssnittet av Eclipse visas i figur 12.



Figur 12. Grafiska gränssnittet för programmering i Eclipse.

2.2.8 WampServer

WampServer är en utvecklingsmiljö för Windows operativ systemet för att ta kontakt med en Apache Webbserver.

2.2.9 Pageant

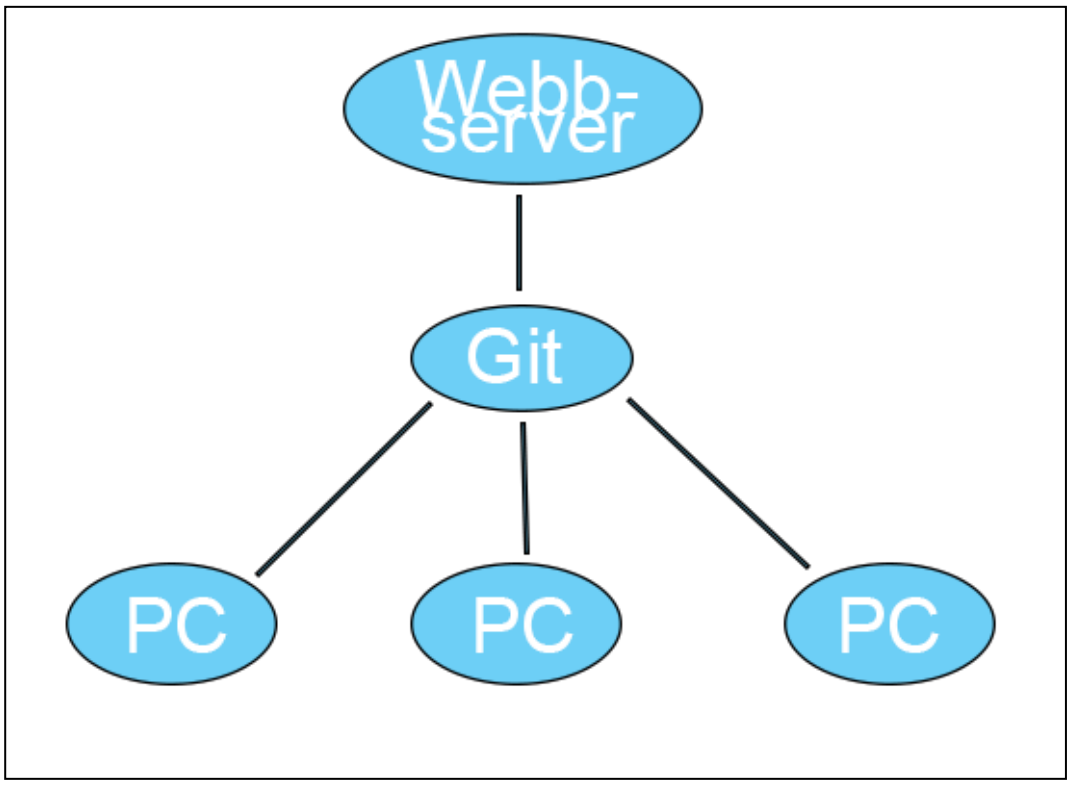
Pageant är ett säkerhetsprogram för att autentisera anknytningsförsök till webbservern.

2.2.10 Silverstripe

Silverstripe är ett CMS-verktyg vilket använder sig av öppen källkod. Silverstripe är alltså gratis att använda för vem som helst och kan modifieras för eget behov. Verktyget är långt utvecklat och vid installeringen ger det en god bas till att bygga webbsidan på verktyget.

2.2.11 Git

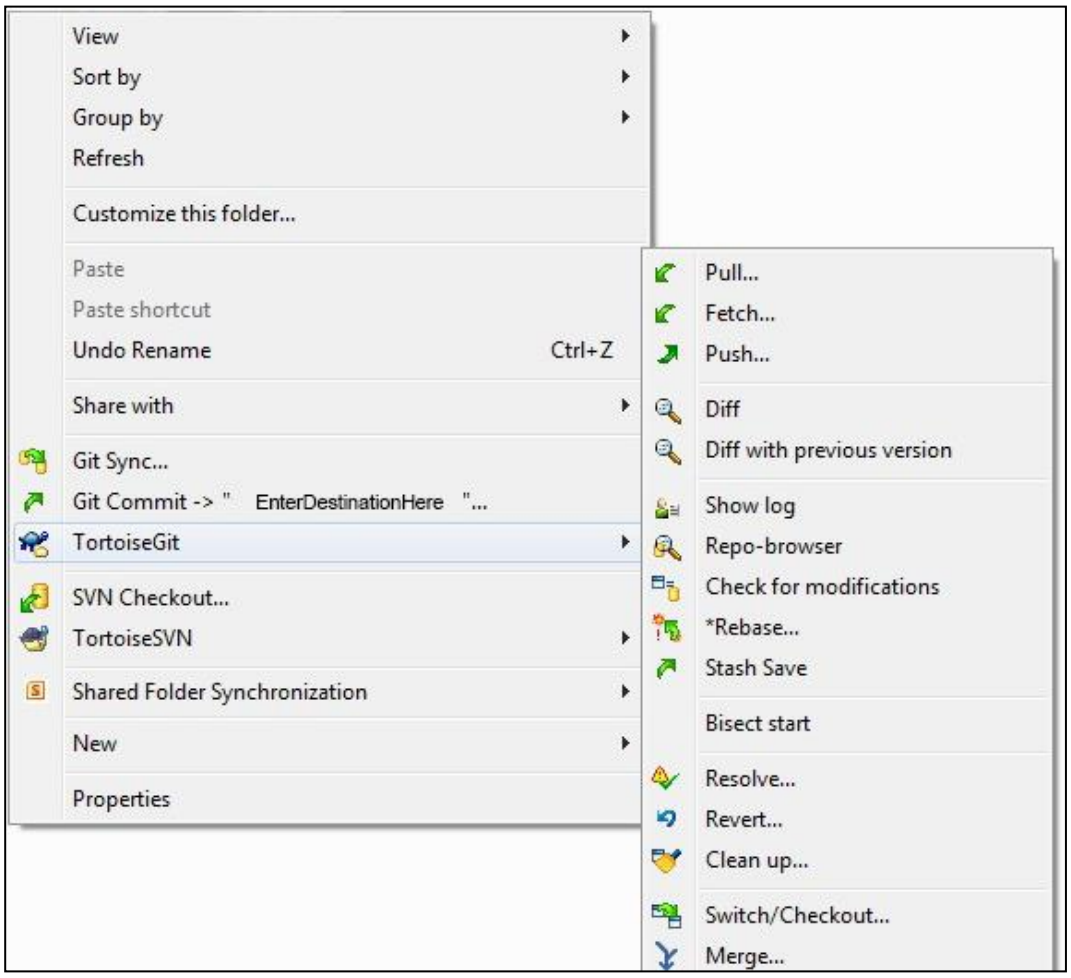
Git utvecklades år 2005 av Linus Torvalds som även ligger bakom Linux. Git är ett versionshanteringssystem (DRCS, Distributed Revision Control System) och används för projekthantering. Då ett team av utvecklare jobbar med ett större projekt är det svårt att hålla reda på olika versionerna. En programmerare skapar en ny version (då programmeraren utvecklar programmet blir det automatiskt nya versioner av produkten) av projektet och laddar upp det till Git. Andra programmerare kan sedan efter ladda ner den nyaste versionen. På detta sätt är det möjligt att se efter vilken version projektet har för tillfället och se till att alla medarbetare är på samma nivå. Git kunde lätt beskrivas som en ”parkering” mellan en webbserver och lokala datorer. I figur 13 visas ett exempel på hur miljön kunde se ut.



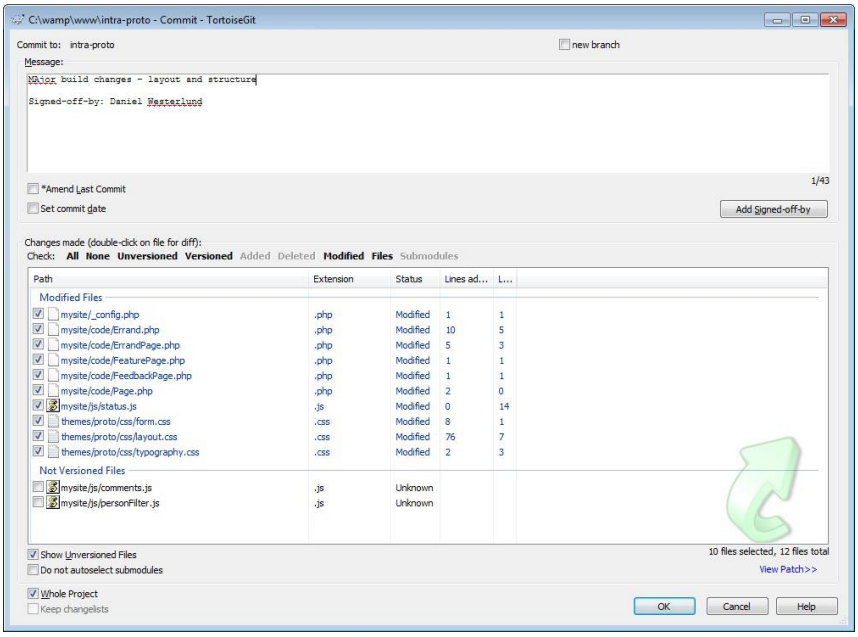
Figur 13. Ett exempel på hur ett projekts utvecklingsmiljö kunde se ut. Datorerna skickar de nyaste versioner till Git, varav andra datorer kan ladda upp de. Sedan kan man skicka projektet vidare till webbservern från Git.

2.2.12 TortoiseGit

TortoiseGit är ett grafiskt användargränssnitt som används för att hantera Git. Med programmet kan man utföra flera olika funktioner som t.ex. Ladda upp filer till Git eller ta ner nyaste versionerna. I figur 14 visas TortoiseGit menyn och i figur 15 görs en ”commit” som sedan följs av en ”push”. Dessa kommandon sparar ändringarna i Git.



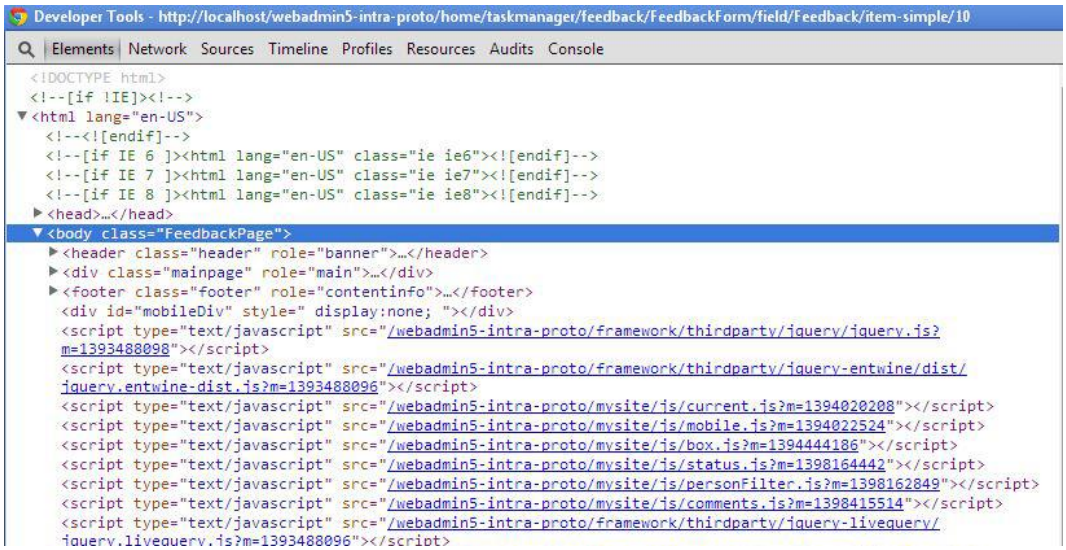
Figur 14. TortoiseGit innehåller flera olika funktioner för att hantera Git, som t.ex. Olika loggar, uppdatering av Git eller uppdatering av egna lokala filer till den nyaste versionen.



Figur 15. Då projektet skall uppdateras still nyaste version skall man köra in funktioner som ”Commit” (i denna figur) och ”Push”.

2.2.13 Chrome Developer

Webbläsaren Chrome innehåller ett inlägg som kallas för 'Developer Mode' då utvecklare kan se in i koden på webbsidorna och övervaka händelser då något händer på sidan. I figur 16 presenteras Chrome Developer vyn**.**



Figur 16. Chrome Developer Mode används för att utvecklare skall kunna studera en webbsida noggrannare.

3 PLANERING AV ÄRENDEHANTERAREN

3.1 Forskning och värdering av resultaten

Planeringsskedet krävde en stor insats i början för att få med alla olika delar som behövdes för ärendehanteraren. Planeringen började med forskning över liknande lösningar som krävdes för detta lärdomsprov. Forskningen drevs av nyckelorden ”issue”, ”errand”, ”bug”, ”tracker”, ”report”, ”feedback”, ”feature” och ”request”. Det fanns många olika lösningar till liknande system, dock inget av dessa var det som strävades efter i detta projekt. Forskningen gav en del bra idéer hur man kunde lösa eventuella problem, men i varje fall i sån mån att allt måste lösas med eget initiativ.

3.2 Kravspecifikation

Kravspecifikationen för ärendehanteraren:

- Innehåller felrapportering, feedback och feature request
- Kunden skall kunna logga in till intranätet för att skicka ett meddelande
- Automatiserat system för att notifiera företaget om ett nytt meddelande
- Funktion för att programmerare kan kommentera till kunderna
- En specifik modul för CMS
- Filhantering
- Realtid status på ärendet
- Responsiv design

3.3 Layout och användargränssnitt

Användargränssnittet och layouten för lärdomsprovet blev utvecklat i ett tidigare projekt för samma uppdragsgivare. Projektet gjordes med hänsyn till detta lärdomsprov och planerades då på ett sådant sätt att ärendehanteraren passade in till användargränssnittet. Syftet med projektet var att skapa en prototyp till ett intranät med olika case.

Ärendehanteraren var redan vid prototypen som högsta prioritet. Projektet krävde mycket forskning om olika lösningar kring GUI som fungerade även responsivt. Under projektets lopp blev det klart vad det var som uppdragsgivaren egentligen krävde, och vi kom till en slutsats som båda parterna var nöjda med. Användargränssnittet är planerat att fungera på olika sätt beroende på skärmens storlek. Forskningen ledde till ett resultat, som blev att användargränssnittet skall byggas upp på tre kolumner då skärmbredden är större än 768 pixel. Apparater med mindre skärmar (som t.ex. Telefon) använder sig av en kolumn. Efter utvecklingen av tre kolumners användargränssnitt måste det lösas att hur eventuella andra dialoger skall visas. Lösningen till detta blev en funktion som flyttar på kolumnerna åt vänster för att ge utrymme till en ny kolumn.

3.4 Planering av funktionaliteten

Funktionaliteten planerades långt på forskningsresultaten, layouten och kravspecifikationen.

3.4.1 Krav och lösningar

3.4.1.1 Felrapportering, feedback och feature request

Kraven ställda för detta lärdomsprov var att bygga ett system då kunderna kan skicka in följande meddelanden: felrapportering, feedback och feature request. Planeringen av detta var långt kundbaserad med tanke på att kunderna skall ha nytta av produkten och den skall vara så användbar som möjligt var det bäst att hitta den lösning som är så simpel som möjligt. Lösningen för detta blev att skapa olika sidtyper (ErrandPage, FeaturePage och FeedbackPage) för varje meddelandetyp. Då varje meddelande har en egen sid typ blir det lättare och

klarare att utveckla projektet vidare. Vad detta konkret innebär är att visiteraren i intranätet ser en meny med navigationsoptioner för varje meddelandetyp. Vid planeringen av layouten togs detta i hänsyn och sidorna med de olika alternativen anpassades även för mindre skärmar.

3.4.1.2 Inloggning

Basen till intranätet ligger på CMS-verktyget Silverstripe som har ett inbyggt säkerhetssystem. Användarkonton sparas in till en databas och användaren kommer till inloggningssidan med att skriva ”/admin” efter webbadressen.

3.4.1.3 Meddelande om ett nytt ärende

Lösningen till denna funktion blev ett e-postmeddelande som skickas ut till en vald person, eller enligt inställningar till en skild e-mail adress enbart för meddelanden för ärendehanteraren. Vid testningen användes e-postadressen arendehanterare@creamarketing.com vilket kan även ses i figur 2.20. Då ett meddelande skickas in via intranätet skickas ett automatiserat e-mail till vald adress som innehåller följande information: Namn på företaget/personen som skickat e-post, information om ärendet som t.ex. Ärendetyp (felmeddelande, feedback, feature request), titel, datum och själva meddelandet. När e-postet kommer fram till arendehanterare@creamarketing.com skall en medarbetare sköta om styrningen till rätt programmerare. I detta skede kunde det vara fråga om projektledare som ser vem som gjort projektet som får in ett meddelande för att sedan kunna skicka vidare e-mailet till valda utvecklaren. E-posten kommer fram till utvecklaren som kan se rakt från meddelandet vad ärendet handlar om. I meddelandet finns även bifogat en länk som styr utvecklaren till rätt problem i intranätet.

3.4.1.4 Funktion för att kommentera på ärenden

Ett av kraven för ärendehanteraren är att det skall vara möjligt för kunderna och utvecklaren att hålla kontakt via ett automatiserat system. Lösningen för detta blev en kommenterings system då båda parterna kan kommentera varandra angående ärendet.

3.4.1.5 Modul för CMS-verktyget

Ärendehanteraren har grundstenen i CMS-verktyget Silverstripe. Ett av kraven för detta lärdomsprovs arbete var att koppla intranätet och CMS-verktyget ihop med en modul. En modul är en extradel i verktyget som hanterar en viss funktion. I detta fall sköter modulen om de samma funktionerna som man kommer åt från intranätet, men det skall finnas eftersom en kund skall ha det så lätt som möjligt att skicka in meddelanden. För att skapa modulen måste relationerna i projektet studeras (mera om detta i nästa avsnitt). Resultatet av modulen skall ha alla de samma funktionerna som intranätet.

3.4.1.6 Filhantering

Silverstripe har en inbyggt funktion för att ladda upp filer som fungerar som grund till filhantering. Lösningen i detta problem är att skapa en långt liknande funktion som finns färdigt med några justeringar. Filhanteringen skall vara så lätt som möjligt att utföra och skall göra det möjligt för en kund att lägga in bilder och dokument bifogat i ärendet för att kunna ge mera information angående problemet eller feedback.

3.4.1.7 Realtid status

Meddelanden som skickas in till ärendehanteraren skall ha en statusindikator som berättar i vilket skede ärendet befinner sig i. Det möjliga värdet skall vara ”New”, ”In progress” eller ”Closed”. Kunden skall kunna följa med i realtid hur kundens ärende framskrider utvecklingsmässigt. Lösning för detta blev en mycket simpel funktion som visar upp en lista på dessa värden som utvecklaren har access att byta på. Då ett nytt meddelande skickas får det automatiskt värdet ”New” och då en utvecklare tar itu med problemet byter man manuellt till ”In progress”. Orsaken varför lösningen inte innehåller en fullt automatiserad funktion är p.g.a. Att i flera fall kan det vara att en utvecklare får meddelandet men hinner inte påbörja hanteringen av problemet. Ifall statuset byter automatiskt värde kan det ske missförstånd mellan kunden och utvecklaren.

3.4.2 Problemområden

3.4.2.1 Automatisering

Automatiseringen av systemet kräver mycket planering eftersom det kan lätt uppstå fel (se exempel ovan med automatisering av status). Vid planeringen måste det även tas i hänsyn att utvecklare inte alltid finns på plats vilket leder till att man inte automatisk kan styra problemen till en viss utvecklare. Lösningen för eventuella problem med automatisering löstes med förflyttning till manuala versioner, men allt som gick att automatisera skall automatiseras.

3.4.2.2 Användbarhet

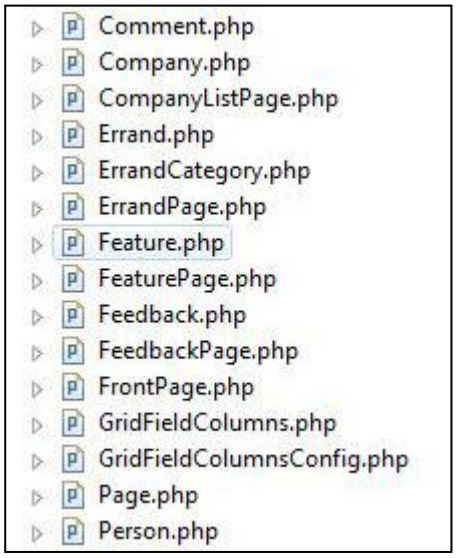
Resultatet skall vara så användbart som möjligt vilket inkluderar att den även skall vara till stor nytta för kunden. Forskningar och första planeringarna måste ändras då det fanns mycket med onödiga funktioner som inte hade någon speciell nytta utan var mera som extra funktioner.

3.4.2.3 Responsiv design

Planen för responsiva designen är att skapa en layout som bygger på tre kolumner som sedan skjuts upp åt vänster när en ny flik öppnas. Problemet med detta är att när man förflyttar sig till mindre skärmar (speciellt Tablet i detta fall) måste övriga lösningar hittas på då det inte är möjligt att rakt förflytta sig till den minsta upplösningen (telefon). Tablet versionen som responsiv är ett problemområde då det inte kan använda sig av den ursprungliga planeringen. Lösningen till detta problem presenteras i senare avsnitt.

3.5 Planering av relationer

En bra planering av relationer mellan olika objekt är en av de viktigaste aspekterna för att ärendehanteraren skall fungera enligt förväntningar. Planeringen innebär en översikt om de olika delarna av systemet och hur de länkas med varandra. Redan vid tidig planering gällde det att ta detta i hänsyn för att kunna bättre inse hur programmet kommer och fungera och hurdana funktioner krävs. I figur 17 presenteras en del av objekten som använts för att utveckla hanteraren.



Figur 17. En listning på en del av de olika objekten som ärendehanteraren bygger sig på.

Vid planeringen av olika relationer måste man fundera på flera aspekter. Speciell eftertanke skall läggas i kopplingarna. För att konkretisera detta avsnitt finns det ett exempel nedan. Ärendehanteraren är till största del byggt med PHP som är ett objekt orienterat programmeringsspråk. Objekt orienterad programmering använder sig av objekt, relationer och metoder mm. Relationerna är en av de viktigaste delarna eftersom de skapar länkar mellan olika objekt som sedan påverkar direkt till funktionaliteten i programmet.

Några exempel på relationer: has_one, has_many, many_many.

De följande relationerna skapades för att bygga upp funktionaliteten.

Company: Company står för företagen i systemet och är alltså ett objekt. För att skapa de relationer som behövs för att Company skall fungera som planerat, måste följande aspekter tas i beaktande:

> Vilka andra objekt skall vara länkade till Company och i hurdana relationer?
> Vad är syftet med att ska en relation mellan objekten?
>
> Company – has_one – CompanyListPage | Detta betyder att ett Company

objekt har en CompanyListPage, d.v.s. en sida för att presentera sig själv.

Company – has_many – Person | Detta betyder att företaget kan ha manga personer, d.v.s. arbetare eller kontakts personer.

CompanyListPage: Detta objekt är skapad för att företagen skall ha en sida där de presenteras. CompanyListPage har bara en relation och den är:

CompanyListPage – has_many – Company | Detta betyder att CompanyListPage har flera företag som den skall presentera.

Person: Person i detta syfte är en medarbetare i ett företag och kallas då för employee. En person är länkad till ett företag.

Person – has_one – Employer | Detta betyder att en person kan ha en arbetsgivare.

Errand: Detta objekt står för ärenden som skickas in via ärendehanteraren och har flera attribut än de andra. Objektet innehåller även flera relationer än de andra.

Errand – has_one – ErrandPage | Detta betyder att ärendet har en sida där det presenteras.

Errand – has_one – Company | Ärendet har ett företag som skickat in meddelandet.

Errand – has_one – Person | Det finns en kontakts person som tar hand om ärendet.

Errand – has_one – Category | En tidig utvecklingspunkt då felrapportering, feature request och feedback gick via att välja en kategori. Detta är inte mera i användning men lämnades in eftersom det var ett steg av utvecklingen.

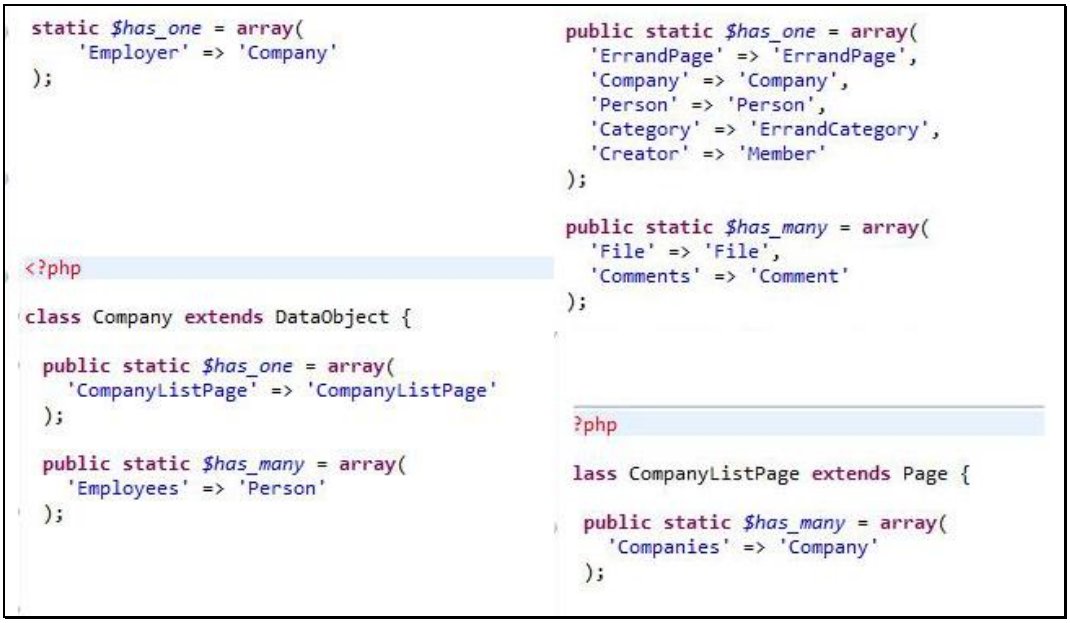
Errand – has_one – Creator | Detta används för konto hantering, ett konto har skickat in ärendet och detta objekt sköter om dess rättigheter att editera ärendet.

Objektet Errand har även relationer som kräver flera insatser och använder sig av has_many.

Errand – has_many – File | Detta betyder att ett ärende kan innehålla flera bifogade filer.

Errand – has_many – Comments | Ett ärende kan ha flera kommentarer som skickas mellan kund och utvecklare.

Relationerna presenteras i kod form i figur 18.



```
static $has_one = array(
    'Employer' => 'Company'
);
```

```
<?php

class Company extends DataObject {

  public static $has_one = array(
    'CompanyListPage' => 'CompanyListPage'
  );

  public static $has_many = array(
    'Employees' => 'Person'
  );
```

```
public static $has_one = array(
  'ErrandPage' => 'ErrandPage',
  'Company' => 'Company',
  'Person' => 'Person',
  'Category' => 'ErrandCategory',
  'Creator' => 'Member'
);

public static $has_many = array(
  'File' => 'File',
  'Comments' => 'Comment'
);
```

```
?php

lass CompanyListPage extends Page {

 public static $has_many = array(
   'Companies' => 'Company'
 );
```

Figur 18. Kod version av relationerna mellan objekten.

4 UTVECKLINGSPROCESSEN

4.1 Om utvecklingsprocessen

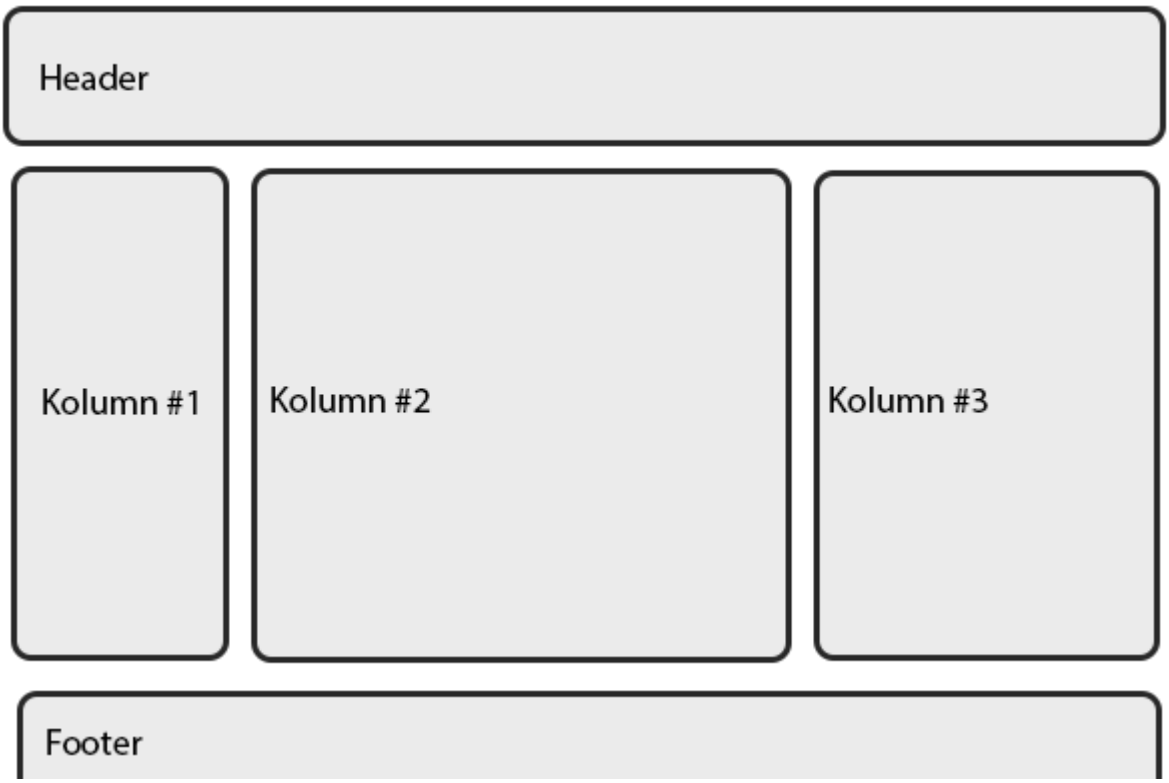
Utvecklingsprocessen av ärendehanteraren delades i följande fyra delar som framskred enligt upplistningen. Planeringsskedet blev förklarat i tidigare kapitel och beskrivs inte i detta kapitel.

4.1.1 Planering

Denna del är beskriven i tidigare kapitel.

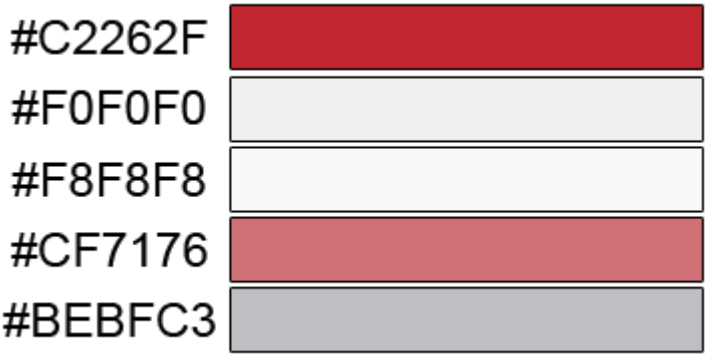
4.1.2 Utveckling av grafik och layout, en grund till ärendehanteraren

Grunden för grafiken och layouten fick sin början av ett parallellt projekt. Syftet med projektet var att hitta lösningar för olika case angående ett intranät. Planen för layouten gick ut på att ha en tre delad sida i form av kolumner. I figur 19 presenteras ursprungliga planeringen för layouten av ärendehanteraren. Utvecklingen mötte layouten mycket långt med några små justeringar.



Figur 19. Den ursprungliga planen för layouten i ärendehanteraren. Header är översta delen på sidan och innehåller oftast element som logo för företaget, språkval, sökfält och navigationen. I detta fall blev navigationen insatt till Kolumn #1. Footer är den nedersta delen på en sida och innehåller oftast element som kontakts uppgifter och ikoner för social media.

Färgskalan för ärendehanteraren härstammar från uppdragsgivarens logo vilken har på sig rött och grått. Utgångsfärgerna i hex kod är följande: #C2262F, #F0F0F0, #F8F8F8, # CF7176 och #BEBFC3. I figur 20 presenteras färgerna med hex koden och färgen som koden resulterar i.



Figur 20. Hex-koder och motsvarande färg

Ett av kraven för ärendehanteraren var responsiv design. Responsiv design måste planeras redan i ett tidigt skede och vid utvecklingen måste det användas speciella metoder för att uppnå en fungerande responsiv lösning.

4.1.2.1 Första steget i layout utvecklingen

Utvecklingen av ärendehanteraren använder sig av flera olika metoder och program, vilka finns utlistat i tidigare kapitel. Första steget för utvecklingen kräver programmen WampServer, Pageant, TortoiseGit, Eclipse samt en installation av Silverstripe. Efter installationen av programmen och Silverstripe är utvecklingsmiljön färdig.

Layouten programmeras med följande programmeringsspråk: HTML5, CSS och JavaScript. Silverstripe CMS-verktyget har använder sig mycket av PHP, men detta blir nämnt först i nästa kapitel eftersom layouten inte använder sig av PHP, istället används PHP för att binda ihop de olika objekten som t.ex. header, footer, navigation osv.

Först programmeringssteget skapade ett simpelt HTML dokument enligt W3C standarder och uppbygger sig enligt följande:

4.1.2.2 Elementet <header>

Headern är den översta synliga delen i ett HTML dokument och innehåller oftast logo samt navigation. I detta fall är navigeringen placerad på en annan plats men

”Logout” funktionen finns inbäddad i en navigation som finns i headern. En sida kan byggas med flera olika navigations balkar. Logon i vänstra uppe hörnet innehåller en länk som omstyr användaren till första sidan. Logout knappen loggar ut användaren från intranätet.

4.1.2.3 Elementet <body>

Body är själva materialdelen på en webbsida och är oftast placerad i mitten av HTML dokumentet efter <header>. Elementet <body> i ärendehanteraren innehåller en undernavigation som används för att navigera på sidan samt de olika kolumnerna för materialet i intranätet. Nederst i elementet finns även en informationsruta som berättar namnet på inloggade användaren samt en kort fras med välkomst hälsning.

4.1.2.4 Elementet <footer>

Footer elementet används ofta för att ha in information av olika sorter och används i ärendehanteraren för att berätta vad produkten är och med hjälp av vad den är uppbyggd.

4.1.2.5 Andra steget i layout utvecklingen

Efter en grund har utvecklats till produkten är det dags att fylla i olika stilar och design för att få ett bättre utseende till produkten. HTML använder sig av Cascading Style Sheets (CSS) som binder sig i HTML dokumentet för att editera klasser och id:n som elementet fått under utvecklingen av HTML dokumentet.

Ärendehanteraren använder sig av flera olika nyanser av grått och rött. Resten av delarna har basfärgen vit. För att knyta en färg eller stil till ett element används CSS med rader som t.ex.

body {background:#FFF;}. Denna rad ger hela body elementet vit färg (#FFF motsvarar vit färg).

Ärendehanteraren har fått sitt utseende till elementen med bland annat följande stilar:

- Background-color: Bakgrundsfärg
- Background-image: Bakgrundsbild
- Background-position: Position för bakgrunds bild
- Border-radius: Runda kanter
- -Webkit-border-radius: Runda kanter för t.ex. Safari webbläsare
- -Moz-border-radius: Runda kanter för Mozilla webbläsare
- Font-size: Storlek på texten
- Color: Färg på texten
- Position: Positioneringen av ett element
- Display: Visa/gömma ett element
- Float: Elementen flyter åt ett visst håll
- Margin: Mellanrummet ett element skapar till nästa element
- Padding: Mellanrummet innanför ett element
- Border: Linjer runtomkring elementet
- Width: Bredd på ett element

En normal design använder sig av pixlar för att berätta om t.ex. bredden på ett element. Vid utvecklingen av en responsiv design skall pixlar ersättas med procent. För att räkna ut procentuella bredden måste det finnas kunskap om följande egenskaper: Hur bred är elementet i pixel som skall omvandlas till procent? Hur bred är elementet som det skall jämföras med, d.v.s. förälder-elementet? Nedan ett exempel för att förklara denna uträkning bättre.

Exempel:

Body är bakgrunden på hela sidan och har en inmatad bredd på 1000px. Innanför body finns ett element som har döpts till Inner. Inner är 900px bred och är en del

av body eftersom den befinner sig innanför förälder elementet body. För att räkna ut procentuella bredden till Inner finns det nu två värden:

Body:1000px
Inner:900px
Inner / Body = X

900 / 1000 = 0.9 vilket motsvarar 0.9/1 = **90%**

En annan aspekt för att skapa en fungerande responsiv sida är att se till att fonterna är läsbara på en mindre apparat än datorn. Fon-size är en stilinställning som används för att berätta hur stora fonterna är på sidan. Vid en normal design inmatas värden med pixlar, men då det handlar om responsiv design skall det använda sig av em. Stilen kan inställas via CSS och fungerar som de andra stilarna. Måttet em fungerar liknande som procent. Till skillnad från procent måste em ha ett värde som det kan relatera sig med, oftast fylls detta värde in i body elementet med raden: body {font-size: 100%;} Font-size: 100% berättar webbläsaren att texten skall ha vara av full storlek vilket motsvarar pixelstorlek av 16. Vid uträckningen av em dividerar man den font storlek som man vill använda sig med 16. Ett exempel på detta: en text skall ha font storleken 12, vi dividerar 12 med 16 och får em värdet 0.75. Värdet matas alltid in i form av 0.75em och kommatecken skall inte användas.

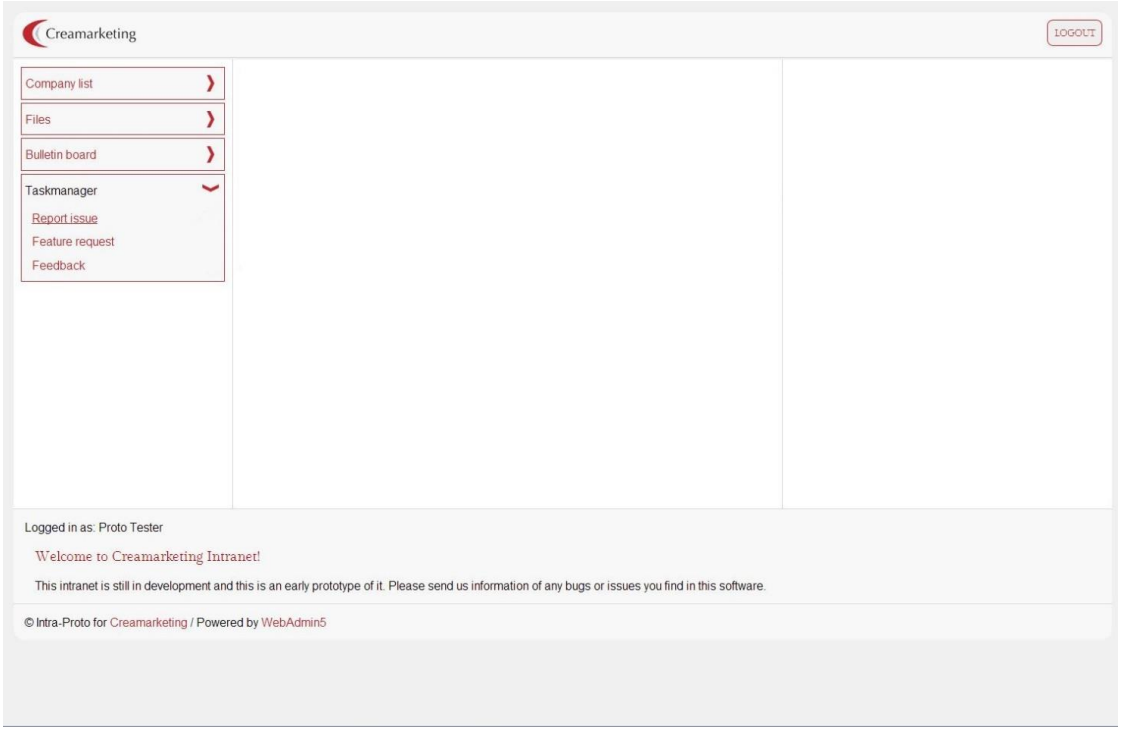
4.1.2.6 Tredje steget i layout utvecklingen

Ett HTML dokument med knyten CSS fil kan ofta vara tillräckligt att skaffa en modern och grafiskt fin webbsida. Emellanåt behövs det dock en del justeringar som måste utföras med JavaScript. Ärendehanteraren använder sig av en del skript för att lägga till klasser och id:n samt för att förflytta element. JavaScript används även för att göra korrigeringar i responsiva designet. I figur 21 visas hur JavaScript används för att lägga till klasser till element.

```
(function($){

    $(document).ready( function(){

        $('.thirdnavigation li').click( function(){

            $('.thirdnavigation li').removeClass("current");
            $(this).addClass("current");

        })

    });

})(jQuery);
```

Figur 21. Ett JavaScript för att sätta till klass ”current” till ett navigationsalternativ då man klickar på det.

Slutresultatet för layouten och grafiska gränssnittet kan ses i figur 22.

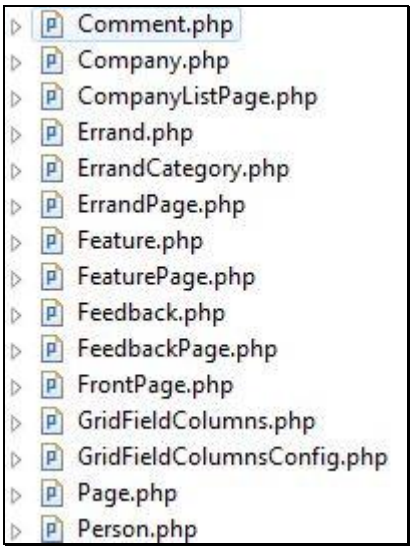


Figur 22. Slutresultatet för användargränssnittet.

4.1.3 Utveckling av funktionalitet och databas

Ärendehanteraren är långt uppbyggt med PHP eftersom grunden bakom projektet ligger i Silverstripe. Silverstripe är långt PHP baserat och för funktionalitetens skull är det viktigt att följa samma linje.

Layouten förklarades i föregående kapitel och detta kapitel berättar om ”motorn” bakom ärendehanteraren. Ärendehanteraren använder sig av objekt orienterad programmering, d.v.s. det finns olika objekt med attribut och de använder sig av metoder (funktioner). Planeringsskedet gav en överblick till vilka objekt projektet kräver. I figur 23 visas alla de objekt som används av ärendehanteraren. Det finns flera objekt som är inbäddade i Silverstripe men de skall inte nämnas i detta dokument eftersom de inte är relevanta för programmet.



Figur 23. Listning på objekt som används i ärendehanteraren.

4.1.3.1 Skapandet av objekt

Detta kapitel innehåller även kod som inte jag personligen skapat. Det inkluderar förflyttningen av kolumner, editerade Silverstripe funktioner samt vissa JavaScript. Under ett möte med uppdragsgivaren beslöts det att en del funktioner överskrider nivån på detta lärdomsprov och kan anses för svåra till en person som inte jobbat med Silverstripe i en längre tid.

Objekten planerades redan i ett tidigt skede och var klara vit utvecklingsskedet.

Figur 25 visade de objekten som är relevanta för detta lärdomsprov och de följer långt samma struktur och uppbyggnad eftersom de representerar tre milstolparna med lärdomsprovet: felrapportering, feedback och feature request. I figur 24 presenteras hur ett objekt är uppbyggt.

```
public function getCMSFields() {

  $types = new FieldList(array('Module'  => 'Module', 'Layout' => 'Layout', 'WebAdmin5' => 'WebAdmin5', 'Other' => 'Other'));
  $status = new FieldList(array('New'  => 'New', 'In progress' => 'In progress', 'Closed' => 'Closed'));
  $priority = new FieldList(array('High' => 'High', 'Normal' => 'Normal', 'Low' => 'Low'));



  $config = GridFieldColumnsConfig::create();
  $dataColumns = $config->getComponentByType('GridFieldAdvancedDataColumns');
  $dataColumns->setIsNested(true);
  $config->removeComponentsByType('GridFieldTabs');
  $config->removeComponentsByType('GridFieldSortableHeader');
  $config->removeComponentsByType('GridFieldFilterHeader');
  $dataColumns->setExpandNested(true);

  $fields = new FieldList(
    new TabSet('Root',
      new Tab('Edit', 'Edit',
        $companyField = new DataObjectListboxField('CompanyID', _t('Errand.COMPANY', 'Company'), DataList::create('Company')),
        $personField = new DataObjectListboxField('PersonID', _t('Errand.CONTACT', 'Contact person'), DataList::create('Person')),
        $typeField = new DataObjectListboxField('CategoryID', _t('Errand.TYPE', 'Errand type'), DataList::create('ErrandCategory')),
        new TextField('Website', _t('Errand.WEBSITE', 'Website URL (including http://)')),
        new TextField('Title', _t('Errand.TITLE', 'Errand title')),
        new DropdownField('Priority', _t('Errand.PRIORITY', 'Priority'), $priority),
        new DropdownField('Status', _t('Errand.STATUS', 'Status'), $status),
        new DateField('Date', _t('Errand.DATE', 'Date')),
        new TextareaField('Description', _t('Errand.DESCRIPTION', 'Errand description')),
        $uploadField = new UploadField('File', _t('Errand.FILES', 'Files')),
        new DropdownField('Programmer', _t('Errand.PROGRAMMER', 'Programmer'), $programmer)
      ),
      new Tab('Comments', 'Comments',
        new GridField('Comments', _t('Errand.COMMENTS', 'Comments'), $this->Comments(), $config)
      )
    )

  );
  $typeField->setMultiple(false);
  $companyField->setMultiple(false);
  $companyField->setShowEditButton(true);
  $personField->setMultiple(false);
  $personField->setExtraProperties(array('EmployerID'));
  return $fields;
```

Figur 24. Kodexempel från objektet Errand. I figuren visas sammankopplingen mellan Errand objektet och Silverstripe. Funktionen getCMSFields() är en inbyggd funktion av Silverstripe som läser in delar till CMS:et. För att ärendehanteraren skall fungera som den gör blev lösningen att hämta fram ett formulär från CMS.

Största delen av objekten har liknande uppbyggnad som presenteras i figuren 26, och innehåller även relationer, extrafunktioner (som automatiserat e-post då ett objekt sparas), fält som skall vara sökbara, förvalda värden samt olika inställningar för sparandet och att ta bort.

4.1.3.2 Former

Lösningen bakom att visa ärenden i interna nätet grundar sig på former och GridFieldColumnsConfig. Idén är att skapa en lista av objekten som visas i en GridField, d.v.s. i ett rutnät. Silverstripe har flera färdiga inbyggda funktioner som måste editeras en del för att få det resultatet som önskades. I figur 25 visas hur rutnätet skapas för FeedBack objektet.

```
52      public function FeedbackForm() {
53          $fields = new FieldList ( $gridField = new GridField
54          ( "Feedback", _t ( "FeedbackForm.LIST", "Feedback list:" ),
55          $this->Feedback (), GridFieldColumnsConfig::create () ) );
56          $actions = new FieldList ();
57          $form = new Form ( $this, 'FeedbackForm', $fields, $actions );
58          $form->setAttribute ( 'data-url', $this->Link () );
59          return $form;
60      }
61  }
```

Figur 25. Lösning för att skapa rutnät där FeedbackForm presenteras.

4.1.3.3 Implementering av modulen till CMS

Ett av kraven för ärendehanteraren var att systemet skall fungera både på front-end och back-end. För att implementera ärendehanteraren till back-end CMS krävdes hantering av ModelManager som sköter om uppgifterna back-end delen hanterar. Genom att tillägga objektet till $managed_models läser Silverstripe in det för att visas i back-end. Detta presenteras i figur 26.

```
<?php

class ErrandAdmin extends ModelManager
{
    public static $managed_models = array(
        'Errand'

    );
```

Figur 26. Implementering av ärendehanteraren till back-end.

4.1.3.4 Mailfunktion

Lösningen för hanteringen av de olika ärendena blev ett e-postutsläpp. Då en kund sparar ett meddelande, skickas det ett mail till uppdragsgivaren som sedan väljer en utvecklare för att ta ärendet till hantering. I figur 27 presenteras lösningen för att automatisera ett e-post då ett objekt sparas.

```
public function onAfterWrite() {
      if ($this->isChanged('ID')) {
          $companyField = $this->From();

          $sendmail = new Email();
          $sendmail->To = "tester_1@creamarketing.com";
          $sendmail->From = "arendehanterare@creamarketing.com";
          $sendmail->Subject = "Ärandehanterare - Nytt ärende av kund $companyField";
          $sendmail->Body = "Ett nytt ärende av kunden $companyField,
          var vanlig och kvitera genom att visitera länken nedan.<br /><a href='{$this->Link()}'>länk</a>";
          $sendmail->send();
      }
}
```

Figur 27. Lösningen för automatiserat mail. Funktionen följer onAfterWrite() vilket betyder att vad händer efter objektet har sparas. Efter det hittar funktionen en IF sats som frågar ifall något har ändrats och ifall det har, skickar funktionen ett mail.

4.1.4 Testning

Ärendehanteraren kunde testas med flera olika apparater för att försäkra funktionaliteten oavsett apparaten.

4.1.4.1 Responsiv design

Den responsiva designen och dess lösningar kunde testas med att använda sig av olika apparater med olika stora skärmupplösningar samt Google Developer Mode med optionen ”Emulate”. Emulate byter vyn i webbläsaren att fungera som en annan apparat, t.ex. en mobiltelefon.

4.1.4.2 Belastning

Ärendehanteraren klarar av sig en hög grad av belastning men detta har egentligen inget med själv programmet utan är direkt relaterat till servern som filerna finns i. För att testa ärendehanteraren i de förhållanden som är tillgängliga ingicks några

simpla test som t.ex. att logga in med flera användare, skicka in flera ärenden samtidigt och spara större filer.

4.1.4.3 Säkerhet

Säkerhetsaspekten är i dagens läge en av de viktigaste saker att ta hänsyn till. Projektet använder sig av MySQL för att spara data i en databas. Databasen befinner sig på samma webbserver som resten av filerna och är då skyddat med lokala lösenord samt begränsningar direkt från webbservern.

4.1.4.4 Funktioner och relationer

Den viktigaste delen i hela projektet är testning av funktionaliteten eftersom testerna berättar samtidigt om eventuella andra problem som har uppkommit under utvecklingen. För att testa funktionaliteten av ärendehanteraren gjordes en del liknande uppgifter som användartestet i kapitel 5 innebär. Egentligen testas funktionaliteten under hela utvecklingstiden eftersom allt som framkommer vid utförandet av uppgifter visar eventuella fel i funktionaliteten.

5 ANVÄNDARTEST FÖR ÄRENDEHANTERAREN

Denna empiriska del av detta lärdomsprov beskriver det användartest som gjorts med ärendehanteraren. En färdig produkt kräver alltid mycket testande och detta test är ett av dem för att försäkra höga kvaliteten på produkten.

5.1 Metod

Ett användartest med fem testpersoner kan redan ge ett pålitligt resultat. Det är dock rekommenderat att ha flera subgrupper för att få ett noggrannare resultat. Barnum beskriver en lyckad testgrupp med tre subgrupper och sammanlagt tio personer. I detta lärdomsprov följer vi dessa riktlinjer, vilket är mycket passande eftersom det behövs testanvändare av olika apparater. (Barnum, 2010, 113-122)

5.2 Syfte

Syftet med detta test är att få resultat om användbarheten i ärendehanteraren samt få reflektioner om produkten. Användartestet skall även ge resultat om de responsiva lösningar i ärendehanteraren.

5.3 Grundstenar i testet

Användartestet grundar sig på att användaren skall följa instruktioner för att självständigt kunna avklara en mängd med uppgifter i ärendehanteraren. Användare tilldelas ett lösenord för att komma in till intranätet som befinner sig på uppdragsgivarens webbserver. Alla användare får ett anonymt ID för att kunna spåras efteråt. Användarna får instruktioner för att utföra följande uppgifter:

- Logga in
- Navigera till Feature request och lägga in ett meddelande
- Navigera till Feedback och lägga in ett meddelande
- Navigera till Taskmanager (Felrapportering)
    - Skapa ett nytt ärende
    - Välj företaget som motsvarar användarens ID nummer
    - Välj kontakts person som motsvarar användarens ID nummer

- Fyll i titel med användarnamnet (användare 1-10)
- Fyll i resten av fälten
- Bifoga en bild
- Fyll i beskrivningsfältet med förbättringsförslag samt eventuella fel användaren tycker stött sig på. Positiva sidor kan även fyllas i.
- Lägg till en ny kommentar ”Detta är ett test”
- Logga ut

5.4 Testgruppen

Testgruppen består av tio (10) personer vilka representerar olika arbetsbakgrunder. Valet av deltagare grundade sig enbart på yrket eftersom ärendehanteraren har i första hand utvecklats för att användas av uppdragsgivarens kunder som kan ha verksamhet inom olika slags områden. Testgruppen består av personer i en ålder från 20 till 30 år och av båda könen. I testgruppen finns det användare av t.ex. följande områden: försäljning, bokföring, säkerhetsindustrin, politik, musik och studeranden.För att undersöka responsiva lösningar har användarna delats upp i olika grupper med varierande apparater. Huvudgrupperna är dator, läsplatta och mobiltelefon. I figur 28 presenteras uppdelningen av användarna samt en beskrivning på apparaterna de använder.

ID		Apparat		Information
Användare #1		Dator		N/A
Användare #2		Dator		N/A
Användare #3		Dator		N/A
Användare #4		Dator		N/A
Användare #5		Läsplatta		Samsung Galaxy Tab 10,1"
Användare #6		Läsplatta		Samsung Galaxy Tab 2 10,1"
Användare #7		Läsplatta		Asus Nexus 7"
Användare #8		Mobiltelefon		HTC Mozart 7 - 3,7"
Användare #9		Mobiltelefon		LG G2 - 5,2"
Användare #10		Mobiltelefon		Apple iPhone 5 - 4"

Figur 28. Information om testgruppen.

5.5 Utvärdering av testresultat

Resultaten av användartestet gav en bra överblick över mindre och större problem som uppkom med testet. Största delen av problemen var apparat relaterade, speciellt med Internet Explorer och Safari. Datoranvändare rapporterade inga problem så felen finns på de responsiva lösningarna. Åtminstone följande funktioner måste korrigeras i vidareutvecklingen:

- Sparandet av ärenden misslyckades på vissa apparater.
- Små grafiska fel med olika apparater
- Äldre version av Windows Phone hade stora problem med grafiken samt funktionaliteten, detta kräver större undersökningar.
- Safari förhindra sparandet i vissa fall totalt
- Font storlekar och ”drop-down menyer” kräver finslipning
- Några användare på läsplatta och mobil rapporterade att de inte hittade knappen för att logga ut, det kunde fixas med att lämna ”Logga ut” texten kvar oavsett skärmstorleken.

Utgångspunkten för testet var att produkten var klar, men användartestet var lyckat eftersom det hittades områden som kräver förbättring. Alla användare var medvetna om att det långt handlar om en ”proto-typ” av intranätet och vissa funktioner kan ha små buggar. Målet med lärdomsprovet uppfylls trots dessa små fel eftersom det finns en bra grund till att utveckla ett större system i detta lärdomsprov. Största delen av användarna var nöjda med programmet samt tyckte att det fungerade mycket bra. Ur utvecklarens synvinkel kunde jag påstå som en egen kommentar att problem som uppstod vid testet var i stor utsträckning apparat- och designrelaterade, vilket är det resultat som var det bättre alternativet eftersom funktionaliteten bakom ärendehanteraren är i skick.

6 UTVÄRDERING AV RESULTAT

Målet med detta lärdomsprov var att utveckla ett fungerande system för uppdragsgivaren att använda i sitt interna nät. Målsättningarna för lärdomsprovet var att produkten skall vara av hög kvalitet, bra användbarhet, responsiv design, hög säkerhetsnivå samt en bra design.

6.1 Utvärdering av de olika målsättningarna

Målsättningarna för ärendehanteraren presenterades i början av detta lärdomsprov och skall genomgås nu då ärendehanteraren är färdigt utvecklad.

Kvalitet

Ärendehanteraren är av hög kvalitet och fyller standarderna för W3C. Kvaliteten höll sin höga nivå p.g.a. kommunikation och noggranna inspektioner av systemet.

Användbarhet

Användbarheten för produkten kan anses vara hög, vid användartestet uppkom några mindre problem som t.ex. att användare inte hittade ”logga ut” knappen. Dessa små problem sänker inte användbarhetsnivån i sig, och de är mycket lätta att korrigera.

Responsiv design

Den responsiva designen kunde utvecklas även bättre, nuvarande lösningen är bra men det kunde finns bättre lösningar till vissa problem. Kravet för att produkten skall vara responsiv fylls eftersom ärendehanteraren är optimerad till telefon och läsplatta.

Säkerhetsnivå

Säkerhetsnivån i produkten är mycket hög. Filerna och information sparas på uppdragsgivarens webb server och är lika skyddad som andra uppdragsgivarens produkter.

Design

Målet för designen var att skapa en vacker och funktionell layout med tre kolumner. Resultatet blev även bättre än planerat och det slutgiltiga resultatet med kolumnerna är: Dator tre kolumner, läsplatta två kolumner och telefon en kolumn.

Vidareutveckling

Uppdragsgivaren skall utveckla sitt interna nät med detta lärdomsprov som grund. Ärendehanteraren är i sig funktionell men vissa lösningar kunde förbättras. Problemen som uppkom med användartestet borde korrigeras.

Uppdragsgivarens kommentarer

Kommentarerna från uppdragsgivaren har varit mycket positiva. Under hela utvecklingsprocessen har det funnits kommunikation mellan båda parterna för att försäkra att utvecklingen går i rätt linje. Speciellt var uppdragsgivaren nöjd med självständigt arbete samt resultatet av lärdomsprovet. Den sista utvärderingskommentararen var mycket positiv:

”Daniel har utvecklat en prototyp som kommer att tas i användning av företaget. I det första skedet kommer företaget att internt ta systemet i produktion, för att i ett senare skede saluföra det till existerande och nya företagskunder. Den nya plattformen har utvecklats bra inom ramen för Daniels slutarbete. Daniel har jobbat självständigt med projektet och kontinuerligt lyssnat på feedback av företagsledningen. Samtliga involverade i projektet från Creamarketings sida är mycket nöjda med resultatet av Daniels slutarbete. Vi ser framemot att implementera lösningen och fortsätta utvecklingen av systemet.”

- *Kalle Smeds, VD, Creamarketing AB* -

6.2 Slutsatser

Detta lärdomsprov var mycket lärorikt och har gett mig en mycket bättre yrkeskunskap p.g.a. dess mångsidighet inom området. Lärdomsprovet lärde mig mycket om objekt orienterad programmering samt dess olika element. Kunskapen inom de olika programmeringsspråken som använts i detta lärdomsprov har blivit enormt bättre och det har skapat en bättre bild över att jobba som utvecklare.

Projektet var mycket intressant att genomföra eftersom det var så omfattande. Jag kommer att använda mig av dessa kunskaper långt i framtiden och är mycket glad att jag valt detta projekt som lärdomsprov. Jag tycker personligen att jag lyckats med ärendehanteraren mycket bra, ifall jag hade haft bättre kunskaper om vissa ämnen inom projektet från början kunde jag ha erhållit även bättre resultat men är även nu mycket nöjd med min prestation.

Det jag ansåg att var mycket svårt i detta lärdomsprov var dokumenteringen på en vetenskaplig nivå eftersom det är mycket svårt att förklara dessa saker i korta avsnitt utan att behöva förklara allt om programmering.

Jag är mycket tacksam för att få utföra detta lärdomsprov för uppdragsgivaren eftersom det kommer att hjälpa mig i framtiden. Jag vill speciellt tacka handledaren samt personalen hos uppdragsgivaren.

KÄLLOR

Böcker

Barnum, C. 2010. Usability Testing Essentials: Ready, Set... Test. Saint Louis. Elsevier Science & Technology.

Castro, E. 2007. HTML, XTHML & CSS, Sixth edition. Berkeley. Peachpit.

Cederholm, D. 2010. CSS3 for web designers. New York. A Book Apart.

Chopra, V., Bakore, Amit. & Eaves, J. 2004. Professional Apache TomCat 5. Hoboken. Wiley.

Delisle, M. 2008. Mastering phpMyAdmin 2.11 for Effective MySQL Management. Olton. Packt Publishing Ltd.

Duckett, J. 2013. Beginning HTML and CSS. Somerset. Wiley.

Forta, B. 2004. Lättpocket om SQL. Sundbyberg. Pagina Förlags Ab.

Gilmore, W.J. 2005. PHP & MySQL – Tehokas hallinta. Jyväskylä. Gymmerus Kirjapaino Oy.

Keith, J. 2010. HTML5 for web designers. New York. A Book Apart.

Marcotte, E. 2011. Responsive web design. New York. A Book Apart.

Wilson, P., McPeak, J. 2010. Beginning Javascript. Hoboken. Wrox.

Elektroniska publikationer

Schwaber, K., Sutherland, J. 2013. Den definitiva guiden till Scrum: Spelets regler.
https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-SE.pdf#zoom=100

Mötes- och konferensföredrag

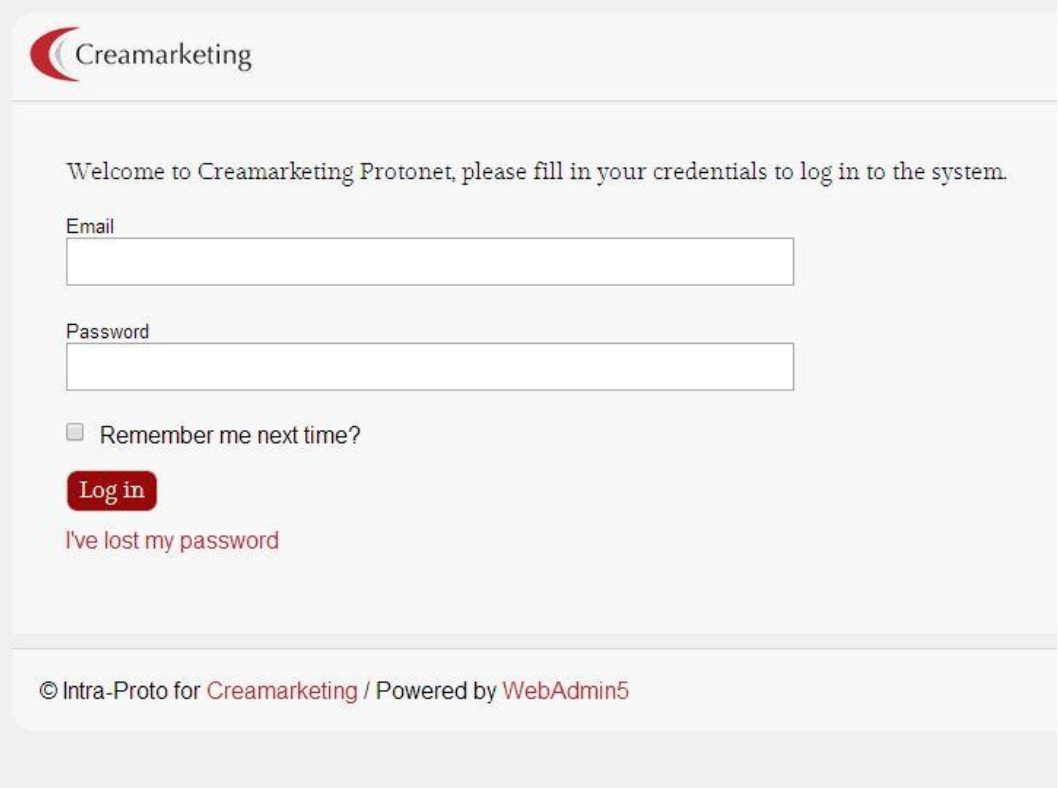
Forsdahl, N. 2014. Huvudansvarig systemutvecklare. 1.1.2014 - 15.5.2014. Vasa.

Smeds, K. 2014. Verkställande direktör. 1.1.2014 - 15.5.2014. Vasa.

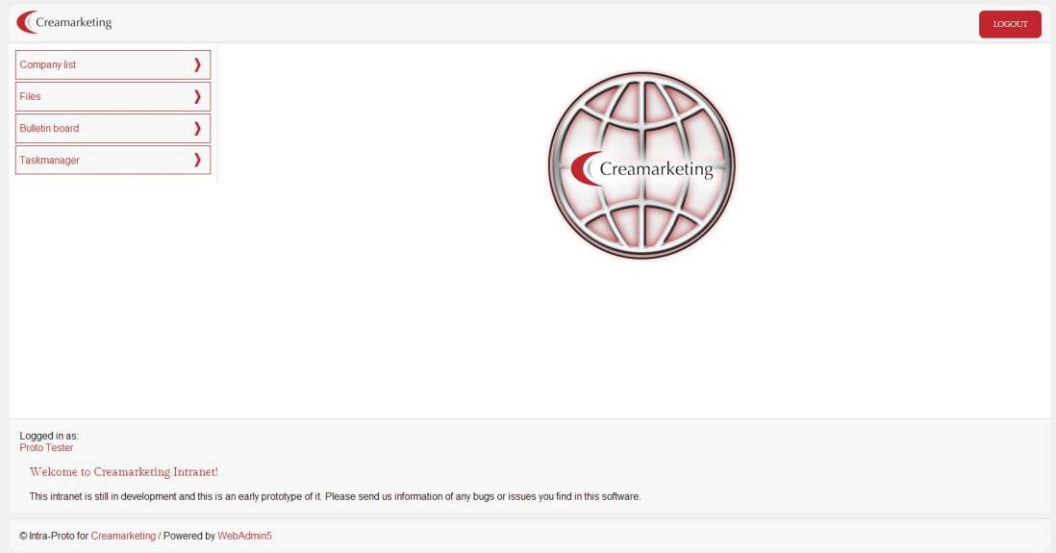
BILAGOR

Denna sista del i detta kapitel är en genomgång av bilagorna för detta lärdomsprov. Eftersom detta arbete var mycket praktiskt kunde det anses vara bäst att presentera resultatet i bilder för att ge en bättre överblick över hela produkten.

Slutresultaten för layouten presenteras i följande bilagor. En beskrivning av bilden finns i bildtexterna.

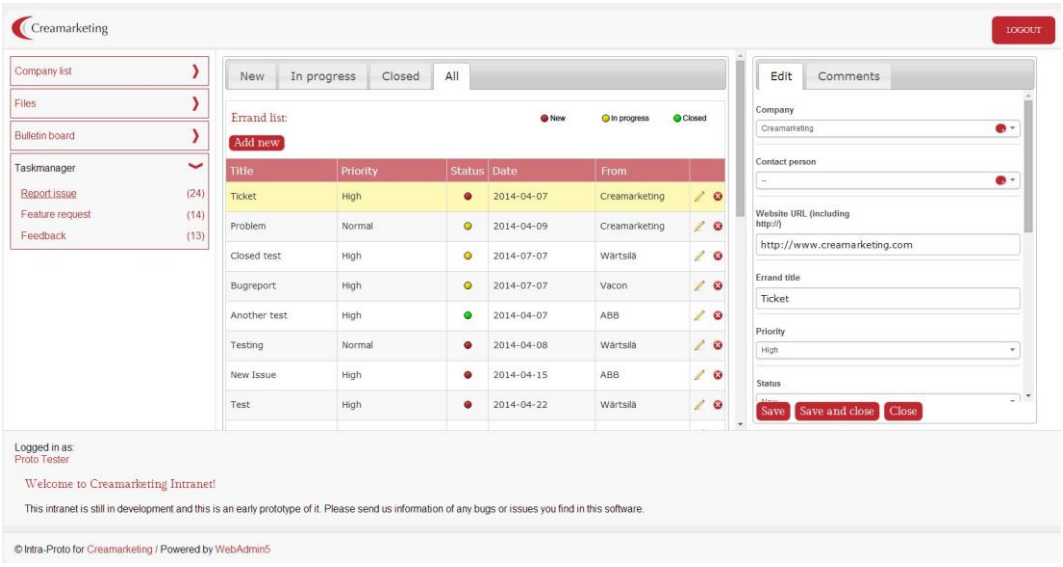


Bilaga 1. Inloggningsvyn till uppdragsgivarens interna nät (proto)

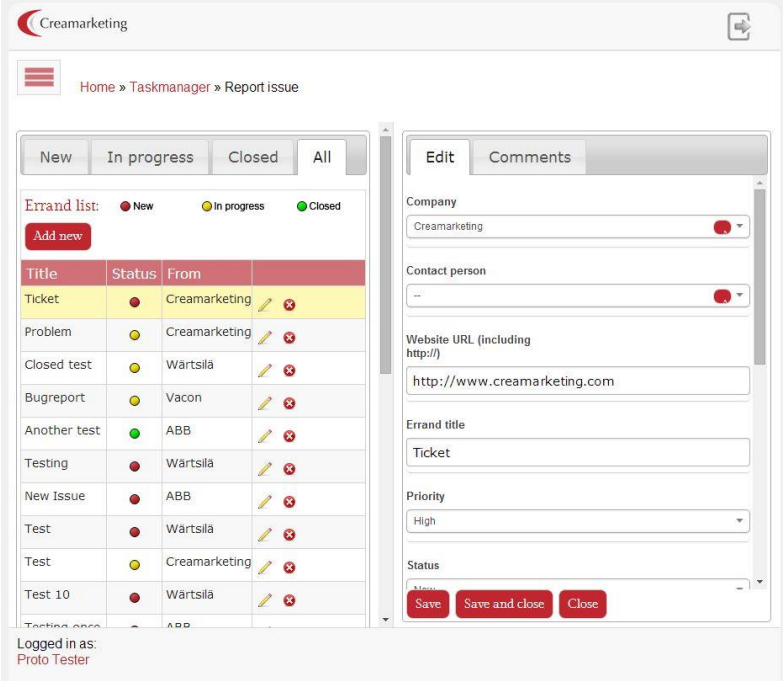


Bilaga 2. Vyn på första sidan med navigering åt vänster, beskrivning och information om användaren som är inloggad finns nederst på sidan. För att logga ut finns det en knapp i högra uppe hörnet.

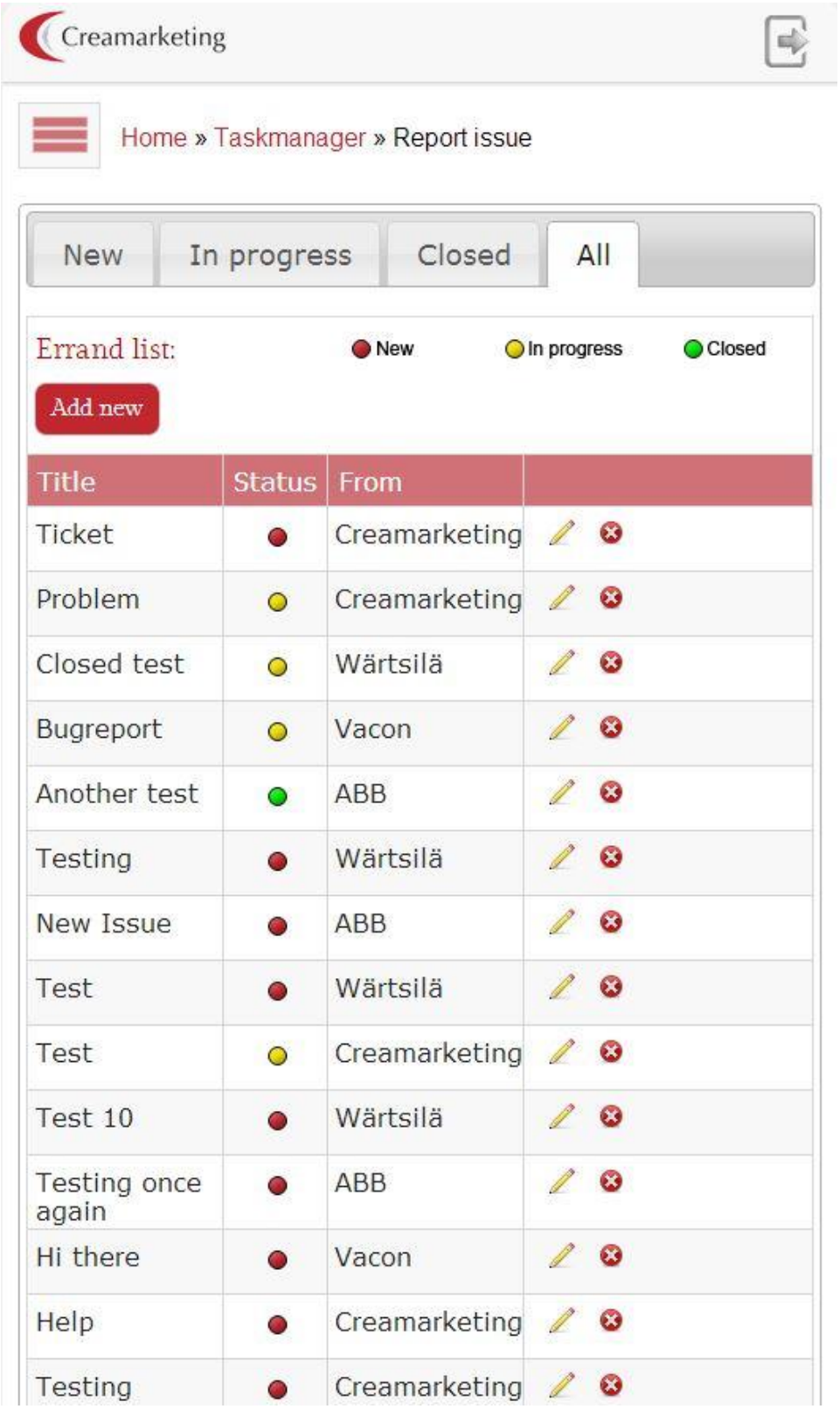
Bilaga 3. Första sidan då användaren är inloggad med mobiltelefon.



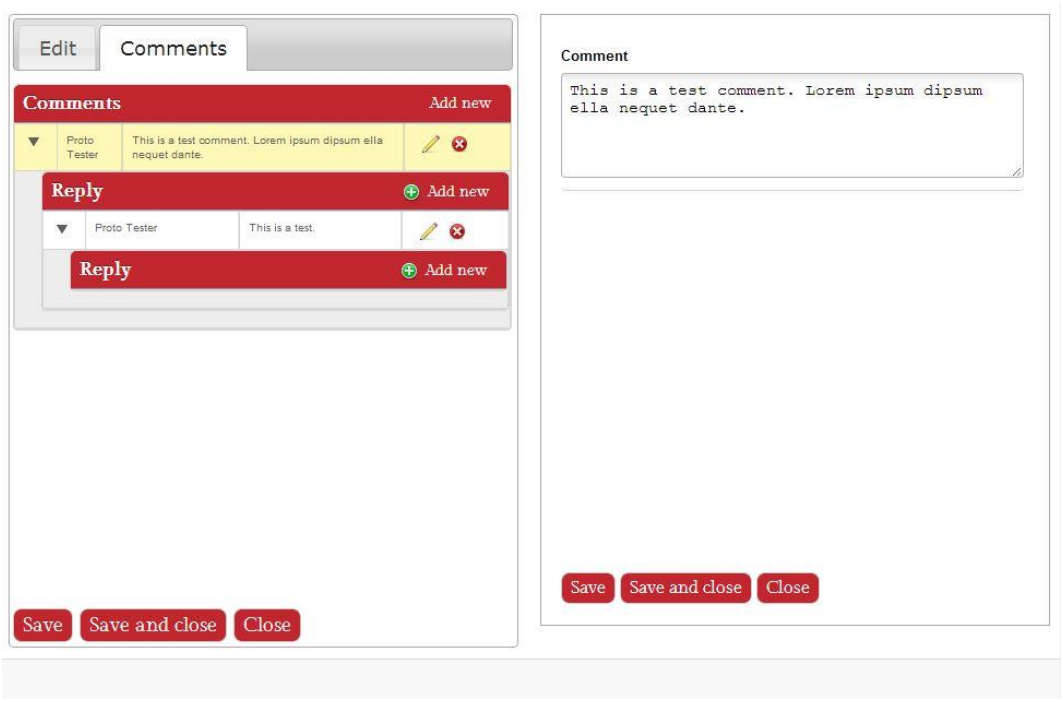
Bilaga 4. Vyn av ärendehanteraren, vänstra kolumnen är för navigation, mellersta kolumnen är för rutnätet som visar ärenden enligt filtrering (ny, utvecklas, färdig eller alla). Högra kolumnen är reserverad till ett öppnat ärende. Då ett ärende öppnas blir det markerat med gul bakgrundsfärg. I den högra spalten hittar man även en länk till att gå in och skriva kommentarer.



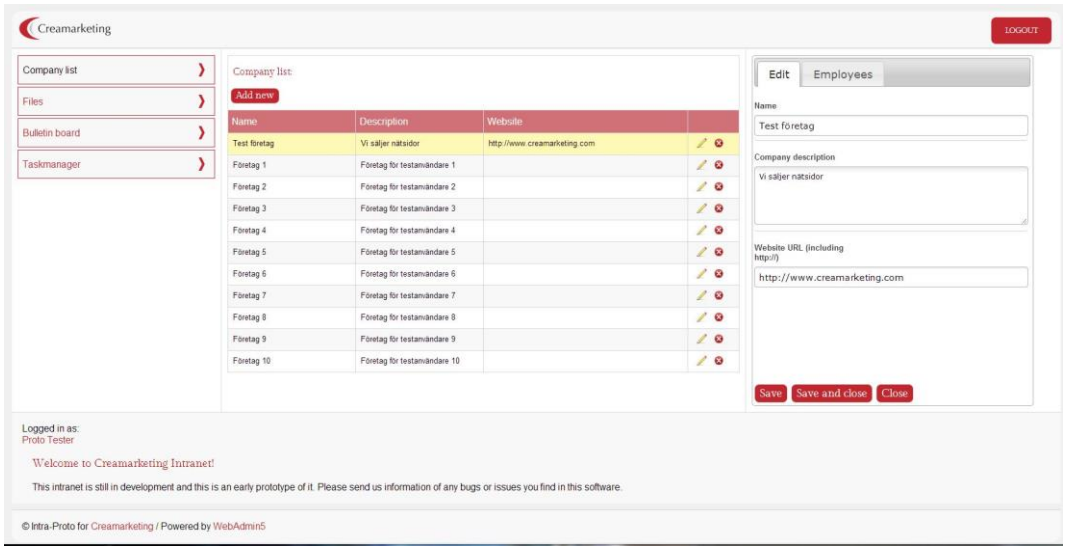
Bilaga 5. Samma vy som i figur 34 men användaren har valt att använda en läsplatta.



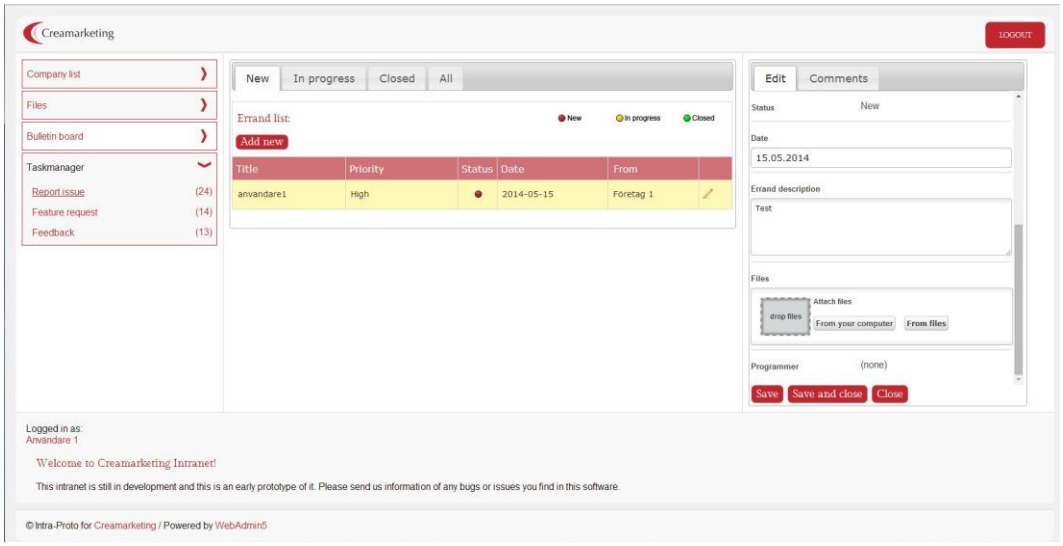
Bilaga 6. Ärendehanterarens responsiva utseende då användaren använder mobiltelefon.



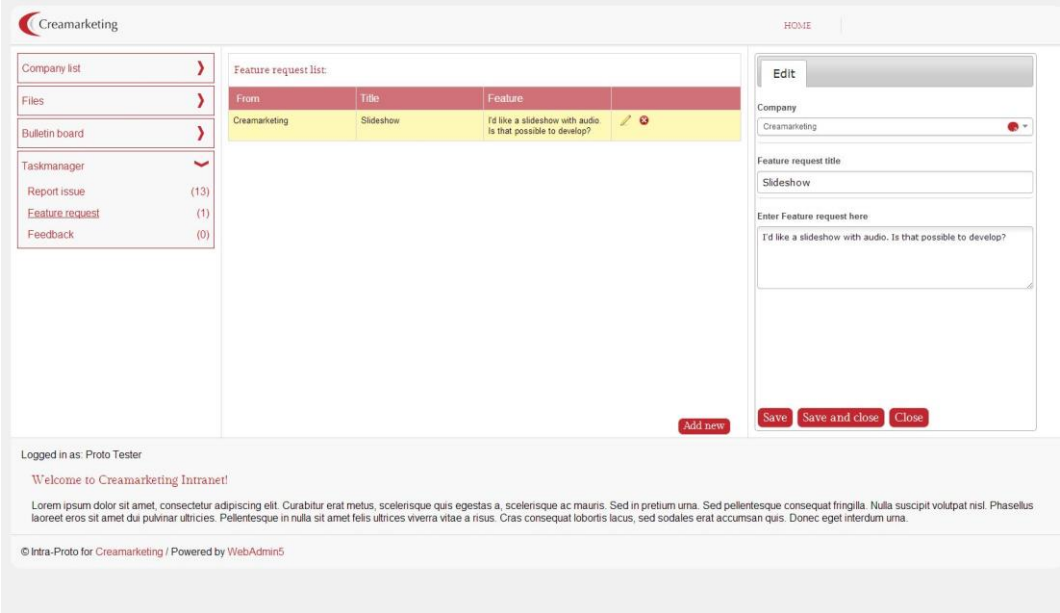
Bilaga 7. Ett ärende är öppnat och användaren har valt att skriva in en kommentar. Kommentardialogen möjliggör ett samtal mellan kunden och utvecklaren. Då användaren vill skapa ett nytt meddelande, skjuts layouten åt höger för att ge utrymme för den nya dialogen med kommenteringsmöjlighet.



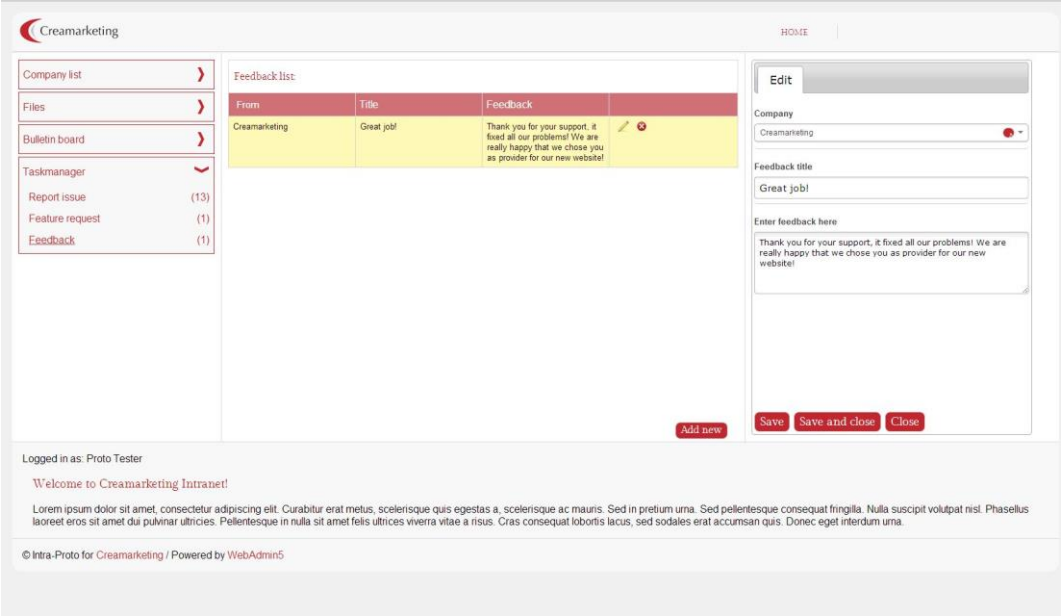
Bilaga 8. Ett kundregister skapades vid samband av ärendehanteraren för att hålla en databas över registrerade kunder. Denna vy är en grund för inmatningen och kommer att utvecklas vidare med mera fält och funktioner. Det är även möjligt att lägga till medarbetare under ”Employees” som blir kopplade till företaget.



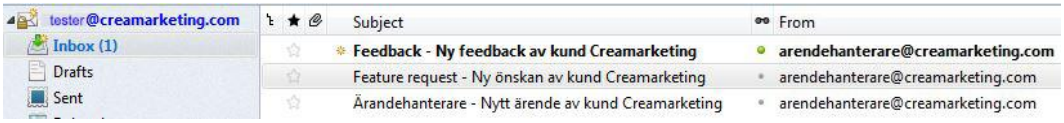
Bilaga 9. En vanlig användare utan administratörrättigheter ser ärendehanteraren på ett annat sätt. Det är inte möjligt för en vanlig användare att byta värden för ”Status”, ”Priority” och ”Programmer”. En vanlig användare ser inte de andra inläggen som gjorts, vilket eventuellt kan ändras vid en vidareutveckling.



Bilaga 10. Utseendet mellan de tre olika ärendetyper ser mycket lik ut. Denna vy är från Feature request sidan.

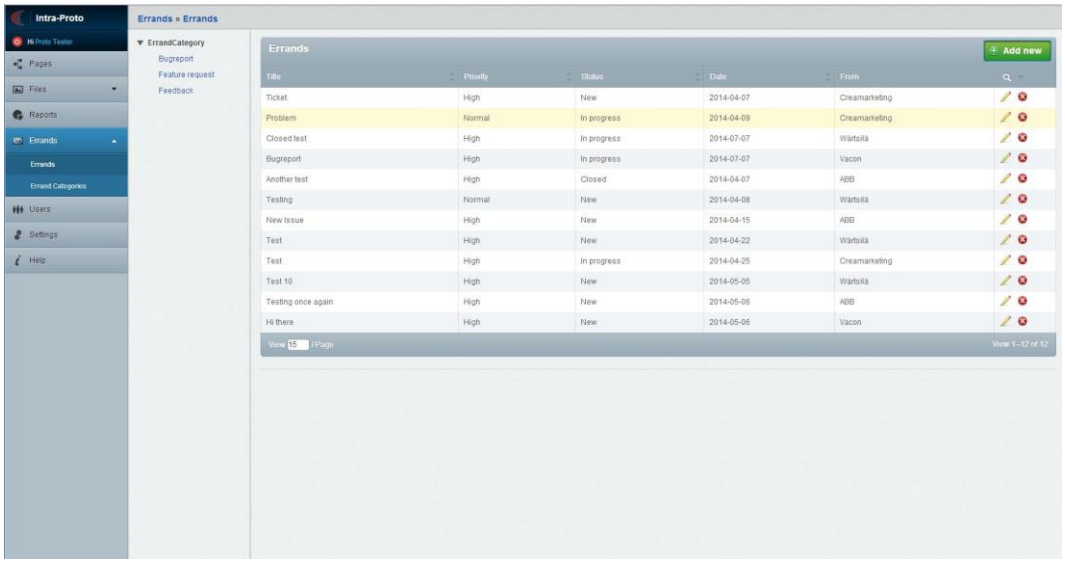


Bilaga 11. Den sista figuren av layouten presenterar vyn för Feedback sidan.



Bilaga 12. Det automatiserade mailing systemet skickar meddelande varje gång ett nytt meddelande sparas. Subjektet inkluderar även en automatiserad funktion som läser in namnet på företaget som har skickat in meddelandet.

Bilaga 13. Vid testskedet av ärendehanteraren var mailutsläppet i denna form. Denna text kommer att bytas ut då ärendehanteraren tas i bruk. Länken i figuren leder utvecklaren till det specifika ärendet i det interna nätet.



Bilaga 14. Ärendehanterares back-end del. CMS visar samma information som användaren kan se på interna nätet.