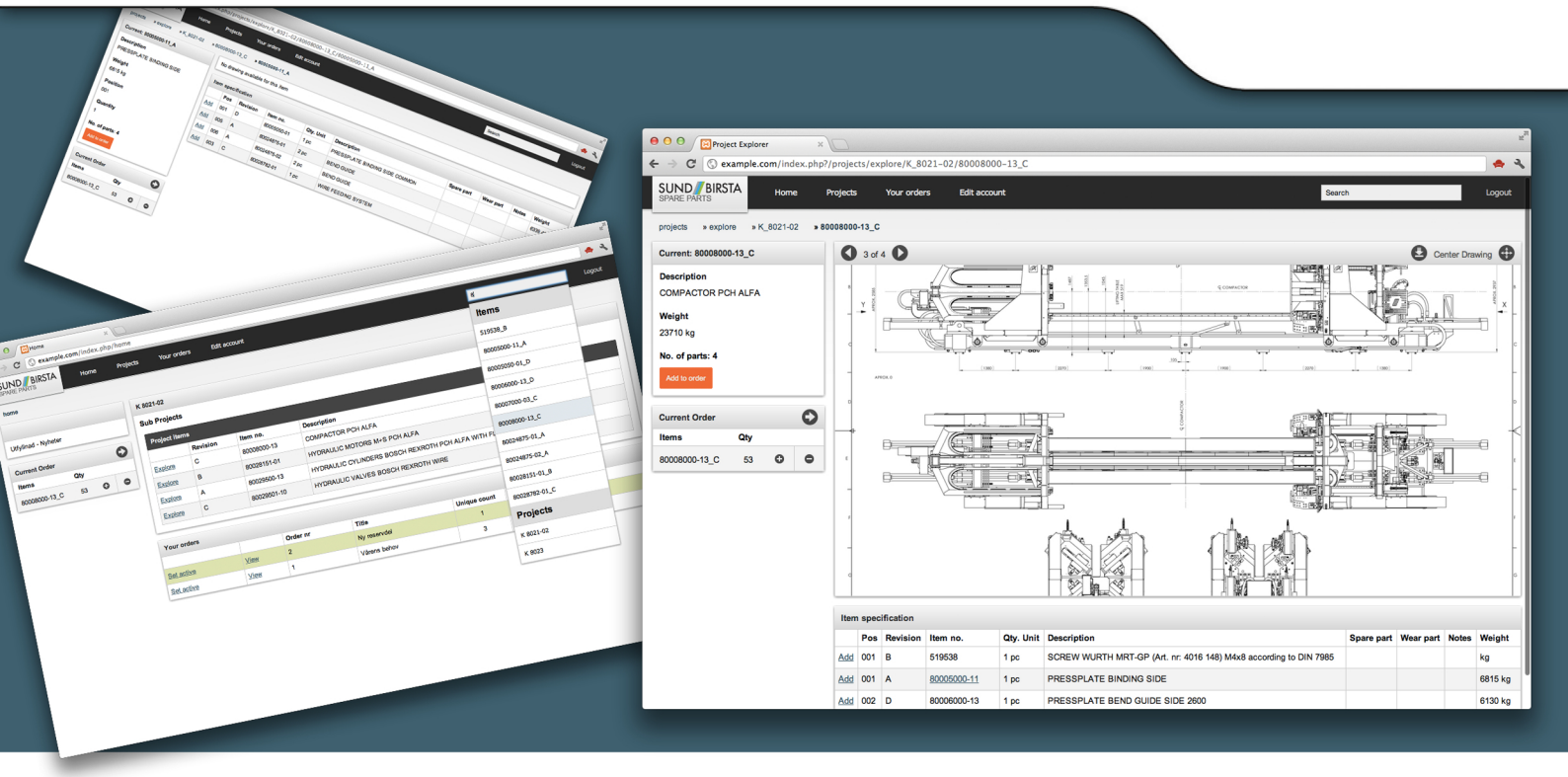
CHALMERS



Utveckling av webbaserad kundportal med hjälp av CodeIgniter och MSSQL

- Prototypframställning åt företaget Sund Birsta

Olof Bjerke, Niklas Doverbo, Herman Fransson, Tomas Odin

Institutionen för Data- och informationsteknik

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2012
Kandidatarbete/rapport nr 2012:06

Denna sida har med avsikt lämnats tom.

Abstract

This bachelor thesis has been commissioned by the company Sund Birsta and was carried out at Chalmers University of Technology. The company develops, manufactures and sells steel mill equipment to an international market. The purpose of the project has been to create a prototype of a user-friendly and easy to navigate customer portal based on Sund Birstas needs.

The work began with a pilot study consisting of a literature study and a visit to the company in Sundsvall to gain an understanding and a good basis for further work. A requirements specification was drawn up after the visit to get a good overview of what would be included in the prototype.

The implementation was performed according to the extreme programming methodology where pair programming was used. The graphical interface was developed using mock-ups and a color scheme was created, using different web tools.

The final outcome was a prototype implemented in the programming language PHP and the CodeIgniter framework. The prototype includes a project explorer with order management, protected by a login system. The prototype uses a database constructed in MSSQL.

The prototype is considered a good representation of the project objectives and provides a detailed picture of the system Sund Birsta is seeking. External development teams can continue to develop the prototype as it follows accepted conventions.

Sammanfattning

Detta kandidatprojekt har skett på uppdrag av företaget Sund Birsta och har genomförts på Chalmers tekniska högskola. Företaget utvecklar, tillverkar och säljer stålverksutrustning till en internationell marknad. Syftet med projektet har varit att skapa en prototyp av en användarvänlig och lättnavigerad kundportal baserad på Sund Birstas behov.

Arbetet inleddes med en förstudie bestående av en litteraturstudie och ett besök hos företaget i Sundsvall för att få en förståelse och en bra grund för det fortsatta arbetet. En kravspecifikation togs fram efter besöket för att få en bra överblick över vad som skulle ingå i prototypen.

Implementeringen utfördes enligt extremprogrammeringsmetoden där parprogrammering användes. Det grafiska gränssnittet togs fram med hjälp av mock-ups och ett färgschema skapades med hjälp av olika webbverktyg.

Slutresultatet blev en prototyp implementerad i programmeringsspråket PHP och ramverket CodeIgniter. Prototypen innehåller en projektutforskare med orderhantering skyddad av ett inloggningssystem. Prototypen använder en databas konstruerad i MSSQL.

Prototypen bedöms vara en bra representation av projektets mål och ger en utförlig bild av systemet Sund Birsta söker. Prototypen kan fortsätta att utvecklas av externa utvecklingsteam då den följer vedertagna konventioner.

Ordlista

AJAX (Asynchronous JavaScript and XML): AJAX är ett samlingsnamn för ett antal webbtekniker som hanterar sidhämtningar till och från en server utan att ladda om en Internetsida.

ASP (Active Server Pages): Ett ramverk för att utveckla dynamsika webbsidor där programmerarären oftast skriver i VBscript.

Back-end: Back-end innebär den delen av systemet som inte syns för användaren. Den består av logiken och datahanteringen.

Bread crumb-navigering: Ett navigationshjälpmedel som hjälper användare se var de befinner sig i en trädstruktur.

CakePHP:
Ett webbapplikationsramverk skrivet i PHP med öppen källkod.

CFML (ColdFusion Markup Language): Ett skriptspråk skapat för webbutveckling.

CodeIgniter: Webbapplikationsramverk med öppen källkod ämnat för att bygga dynamiska webbapplikationer med skriptspråket PHP.

Entitet: Ett väl definierbart objekt i en datastruktur.

ER-diagram (Entity-Relationship diagram): En grafisk representation till entiteter och deras relation till varandra.

Ganttschema: Ett flödesschema som används för att beskriva ett projekts olika faser.

HTML (HyperText Markup Language): Huvudmärkspråket som används av hemsidor.

Javascript: Ett skriptspråk som körs på klientsidan(webbläsaren)

Mjukvaruutvecklingsprocess: Det arbetssätt som används vid planering, skapande, testande och underhåll av mjukvara.

MSDN: Microsoft Developer Network, här samlar Microsoft alla information som rör utvecklare

MSSQL (Microsoft SQL Server): Microsofts SQL-databas

MVC (Model-View-Controller): En systemarkitektur som är vanlig i mjukvarusystem.

MySQL: En SQL-databas som idag ägs av företaget Oracle.

Märkspråk: Språk som används för att strukturera och presentera data.

PHP (PHP: Hypertext Preprocessor): Ett serverskriptspråk som skapat för webbutveckling för att visa dynamiska webbsidor.

PHPMyAdmin: Ett gratis verktyg med öppen källkod för att hantera administrationen av MySQL.

Plugin: Ett mindre program som är gjort för att erbjuda bättre eller extra funktionalitet till ett existerande program.

PNG (Portable Network Graphics): Är ett format som används för digitala bilder.

Python:Programmeringsspråk desginat av Guido van Rossum.

Query: Betyder *fråga* på svenska och är ett kommando skrivet i exempelvis SQL.

Ruby: Programmeringsspråk utvecklat av Yukihiro Matsumoto.

Serverskriptspråk: Ett serverskriptspråk är ett programmeringsspråk som exekveras på en server och behöver inte kompileras innan körning, exempelvis PHP.

SQL (Structured Query Language): Är ett frågespråk som används mot databaser.

Tabell (databas): Används i databaser för att lagra entiteter. Kolumner kan användas för att lagra attribut eller referenser.

Trigger: **F**unktion som utförs automatiskt i databasen vid olika händelser.

URI, URL, URN: En URI (Uniform Resource Identifier) är den kompletta adressen till en resurs på en webbserver. URI består av två delar, URL (Uniform Resource Locator) och URN (Uniform Recource Name). URL är adressen till servern, den består av en ip-adress eller ett domännamn. URN är adressen på själva servern till resursen som efterfrågas.

VBscript: Skriptspråk utvecklat av Microsoft som är baserat på Visual Basic.

Vy (databas): Ett sätt att paketera en komplicerad query och åskådliggöra resultatet som en virtuell tabell.

Zend framework: Ett webbapplikationsramverk uppbyggt i PHP med öppen källkod.

Innehållsförteckning

1 Inledning

Sund Birsta är ett företag som utvecklar, tillverkar och säljer maskiner till stålindustrin. Sund Birsta har en eftermarknadsavdelning som arbetar med att förse befintliga kunder med teknisk service och underhåll av maskiner.

Många av de maskiner som tillverkas har en lång livslängd och byts sällan ut. Däremot kan enskilda artiklar eller delar ha betydligt kortare livslängd. Detta innebär att det existerar ett behov av att förse maskinägare med reservdelar. Beställning av reservdelar sker i nuläget på olika vis, till exempel via e-post eller telefon. När en kund köper en maskin medföljer en pärm med ritningar och artiklarnas positionsnummer. Kunden får även en cd med en avancerad mappstruktur. Denna information skulle lättare kunna tillhandahållas via ett webbgränssnitt.

Sund Birsta vill ha möjlighet att genom en central kanal sälja uppgraderingskit till kunder. Då kan företaget med en egen reservdelsförsäljning erbjuda ett helhetspaket, där företaget både säljer maskinen och är mer involverade i underhållet. För att ge kunderna bättre service vill företaget satsa på en internetbaserad kundportal.

1.1 Syfte

Syftet med projektet är att förenkla Sund Birstas kundförehavanden och effektivisera företagets eftermarknadsförsäljning. Projektet kommer utforska möjligheten att förflytta eftermarknadsförsäljningen och service till en gemensam internetbaserad plattform.

1.2 Mål

Projektets mål är att ta fram en användarvänlig prototyp som ger en så komplett bild av systemet som möjligt. Denna ska utgå från Sund Birstas nuvarande system och företagets önskemål. Det ska även finnas möjlighet för ett annat team ska kunna ta över arbetet.

1.3 Problembeskrivning

En prototyp ska skapas som beskriver en kundportal enligt Sund Birstas önskemål. Projektet ställs inför utmaningen att skapa ett gränssnitt som ger ett branschenligt professionellt utseende. Fokus för prototypen ska ligga på användarvänlighet.

Varje kund ska ha en individuell inloggning för att se relevanta maskiner och ordrar. Den individuella inloggningen ska stänga ute obehöriga personer från att använda systemet. Systemets administratörer ska använda samma inloggningssystem men nå extra funktionalitet.

Sund Birsta vill under projektets gång få kontinuerlig information om hur projektet fortskrider. Företaget arbetar inte inom mjukvarubranschen och därför behöver informationen visas på ett enkelt och tydligt vis. En person som är oinsatt i projektet ska snabbt kunna tolka informationen om projektets fortskridande.

1.4 Avgränsningar

På grund av projektets omfattning kommer endast en del av portalen implementeras. Det exportverkyg som behövs mellan Sund Birstas nuvarande system och prototypen kommer inte skapas. Sund Birstas system använder en avancerad datastruktur och det skulle ta för mycket av projektets tid för att förstå dess uppbyggnad. Säkerhetsaspekten kommer endast behandlas i den del av prototypen som implementeras.

2 Teori

I detta kapitel redogörs relevant teori och centrala begrepp för projektet. Först förklaras programmeringsteknisk teori inom webbutveckling och därefter ges teoretisk bakgrund till grafiska gränssnitt.

2.1 Systemarkitektur

Systemarkitektur är den fundamentala organisationen av ett system gestaltat i dess komponenter, komponenternas relation till varandra och omgivningen. Dessutom beskrivs de principer som styr systemets design samt evolution av systemet (The Institute of Electrical and Electronics Engineers, Inc., 2000).

Att välja arkitektur till ett system får inte ses som en enkel uppgift. Valet av arkitektur påverkar hela systemets prestanda, robusthet och mängden underhållsarbete (Basch, 2000). När systemarkitektur väljs står valet mellan att antingen skapa en egen arkitektur från grunden, eller att använda ett känt designmönster. Fördelar som kan förväntas när designmönster används är minskad utvecklingstid och ökad kvalité hos den utvecklade mjukvaran (Morales-Chaparro et al, 2007). System byggda efter kända designmönster får dessutom en säkerhet inför framtiden då utvecklingsteam kommer känna igen arkitekturen.

Dock borde valet av struktur inte endast motiveras av lättsamt framtida underhåll. Valet av struktur påverkar, som skrivet ovan, hela systemets egenskaper. Att ändra ett systems arkitektur är ett tidskrävande och svårt jobb. Om andra strukturer passar bättre, ska dessa väljas (Sommerville, 2011).

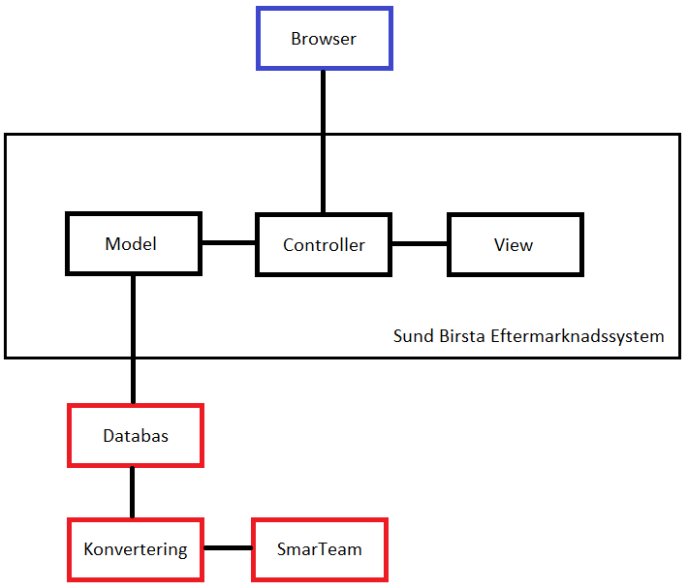
2.1.1 MVC

MVC-mönstret (Model-View-Controller) är en arkitektur som används som bas för interaktionshantering hos många webbaserade system och för de flesta ramverk inom webbutveckling (Sommerville, 2011; Sache, 2010). MVC-mönstret strukturerar upp systemet i tre logiska komponenter:

Model
Model-modulen hanterar systemets data och operationer som associeras med data (Sommerville, 2011).

View
View-modulen definierar och hanterar hur data presenteras för användaren (Sommerville, 2011).

Controller
Controller-modulen hanterar interaktionen med användaren (så som musklick och knapptryck) och skickar vidare dessa till model-modulen och/eller viewmodulen (Sommerville, 2011).



MVC har inte bara fördelen att det är en väl accepterad arkitektur, användning av MVC ger även följande fördelar (Sache, 2010; Morales-Chaparro et al; 2007; Sommerville, 2011):

- MVC-modellen möjliggör att utvecklingsteam kan jobba på olika platser vid olika tidpunkter när ett system utvecklas.
- Det blir lättare att skala upp/ner systemet när det är byggt runt MVC.
- Underhåll av systemet blir betydligt enklare.
- MVC stödjer att data presenteras på olika sätt, vilket underlättar användandet av många vyer i ett system.
- MVC-modellen möjliggör att presentation av data kan ändras utan att detta påverkar hanteringen av datan. Detsamma gäller även åt andra hållet.

2.2 HyperText Markup Language (HTML)

HTML är ett märkspråk (*eng. markup language*) för att strukturera och presentera data på internetsidor. Språket består av taggar som omgärdar data. HTML-koden skickas till en användare när en hemsida hämtas. Koden tolkas av en webbläsare och den strukturerade datan visas för användaren. HTML är en standard som utvecklas av World Wide Web Consortium(W3C). Den nuvarande standarden är HTML 4.01, men HTML 5 är under utveckling (Jackson J., 2011).

Syntaxen för språket består av taggar med möjlighet till attribut. Ett enkelt exempel på HTML är en -tagg:

Kod:

```
<p>Detta är ett stycke</p>
```

Resultat:

Detta är ett stycke

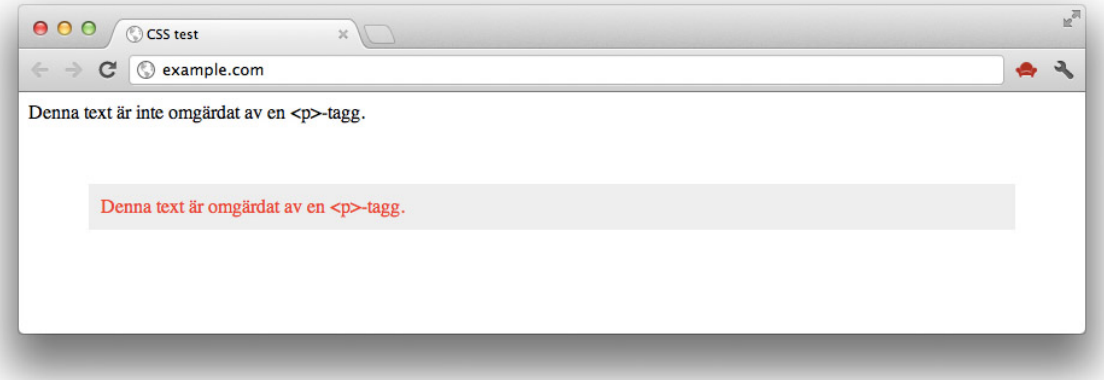
-taggen används för att skapa stycken. Varje tagg som sätts ut har en stängningstagg som innehåller ett snedstreck i början av taggen ( ). När webbläsaren tolkar koden döljer den HTML-koden för användaren. Koden går att komma åt genom att använda "visa källkod"-alternativet som webbläsare har.

2.3 Cascading Style Sheets (CSS)

CSS är ett presentationsspråk som används för att ändra utseendet hos HTML-dokument. Stilmallar ger möjlighet till separering mellan utseende och struktur. CSS är i nuläget i version 2 men utvecklas av W3C för att nå version 3 (W3C, 2012).

CSS används genom att skriva namnet på en tagg och därefter ett antal egenskaper. Nedan beskrivs ett exempel på syntaxen:

```
p {
      margin: 50px;
      padding: 10px;
      color: red;
      background: #eeeeee;
}
```



I fallet ovan används en <p>-tagg med 50 pixlar luft runt stycket och med 10 pixlar spaltfyllnad. Färgen på texten är röd och bakgrunden är svagt grå

2.4 PHP: Hypertext Preprocessor (PHP)

Inom webbutveckling finns en stor mängd programmeringsspråk att välja mellan. Några av de mest populära är PHP, ASP (är egentligen ett ramverk som använder sig av ett skriptspråk, oftast VBscript), Python, Ruby och CFML(ColdFusion Markup Language) (TIOBE, 2012). Valet av språk till ett projekt är viktigt då olika språk är inriktade mot olika områden. (Wall, Christiansen & Orwant, 2000).

PHP (PHP: Hypertext Preprocessor) är ett språk som har öppen källkod och stöder alla stora operativsystem. Dessutom stödjer det de vanligaste databaserna (Purer, 2009). Andra fördelar med att använda sig av PHP är som följer (Lerdorf, Tatroe & MacIntyre, 2006; Purer, 2009; Yu & Lu, 2008):

- PHP är världens mest använda serverskriptspråk som används på över 40% av alla webbapplikationer. Det finns över 2,5 miljoner utvecklare och en community runt språket som är en av de mest dynamiska som finns runt ett programmeringsspråk.
- PHP är enkelt att lära sig och syntaxen är lik C, vilket är något många programmerare känner sig hemma i. VBscript, som är standardspråk till ASP, är inte lika lätt att lära sig och syntaxen är generellt mindre känd än C:s.
- PHP är ett stilrent språk där mycket kan åstadkommas med lite kod. Detta minimerar utvecklingstiden och innebär dessutom att koden blir lättöverskådlig.
- PHP är plattformsoberoende till skillnad från till exempel ASP som kräver Windows.
- PHP är gratis att använda sig av till skillnad från till exempel ASP och CFML (ColdFusion Markup Language).
- Med PHP är det lätt att skapa sidor där utseendet skiljer sig för olika användare.

Utöver detta är PHP bra att använda vid uppbyggnad av e-handelssidor. Detta eftersom det är ett språk som är lätt att lära sig, går snabbt att starta upp och är väldigt bra att använda vid skapandet av personliga sidor (Purer, 2009).

2.5 Ramverk

Petrijevcanin Vuksanovic & Sudarevic (2011) beskriver ramverk som ett abstraktionslager ovanpå den vanliga koden. Lagret ger tillgång till diverse olika bibliotek som kan användas till att lösa de vanligaste problemen som kan uppstå vid programmering. Syftet med att använda ramverk är att undvika repetitiv kod.

Att använda sig av ramverk är dock inte självklart, särskilt inte vid utvecklandet av små applikationer, då har den accepterade konventionen varit att inte använda sig av ramverk alls. Det beror på att ramverk anses leda till ökad komplexitet i koden, ökad exekveringstid samt att inlärningströskeln blir betydligt högre vilket leder till att implementationstiden ökar (Petrijevcanin Vuksanovic & Sudarevic, 2011).

Petrijevcanin Vuksanovic och Sudarevic skriver att fördelarna med att använda ramverk, mer än väl väger upp för nackdelarna med ramverk, även vid utvecklandet av små projekt. Ramverk ger en välstrukturerad arkitektur som enkelt kan visualisera systemet och kod som följer ett känt mönster. Det ges även tillgång till väl testade kodblock. Dessutom blir mjukvaran mer säker då det är mindre risk att säkerhetsmissar uppkommer ifall inbyggda säkerhetsfunktioner används (Sache, 2010).

Möjligheten att använda väl testade kodblock samt återanvända kod leder till minskad implementeringstid. I ett test som gjordes av Petrijevcanin Vuksanovic och Sudarevic tog en applikation 11 timmar att skapa med hjälp av ramverk mot 19 timmar utan. Det minskar tiden till 58 % av ursprungstiden. Alexandru Sache beskriver ett liknande test för en betydligt större applikation. Det tog 46,5 dagar att utföra med hjälp av ett ramverk mot 95 dagar utan. Det minskar tiden till 49 % av ursprungstiden (Sache, 2010).

Det är viktigt att system är uppbyggda av enhetlig kod som följer vedertagna konventioner. Detta är något som underlättas av användandet av ett ramverk. Många mjukvaruutvecklare anser att mjukvaruunderhåll är ett komplext, tekniskt svårt och krävande jobb (Glass, 2006). Att använda sig av ett ramverk leder till att framtida mjukvaruunderhåll blir lättare. Detta har flera anledningar. Det blir lättare för en mjukvaruutvecklare att sätta sig in i den skrivna koden då mjukvaran följer en tydlig struktur. På grund av alla hjälpfunktioner och standardbibliotek hos ramverk blir det mindre kod att underhålla och kod som finns i ramverks bibliotek kan optimeras automatiskt vid en ramverksuppdatering (Petrijevcanin Vuksanovic & Sudarevic, 2011).

Valet av ramverk är en viktig del av planeringsfasen vid mjukvaruutveckling. Hela systemet bygger på ramverkets fundament. På många sätt är det likt valet av programmeringsspråk (Sache, 2010).

När ramverk ska väljas står valet mellan att antingen använda sig av ett ramverk med öppen källkod, eller ett proprietärt ramverk med stängd källkod. Fördelen med öppen källkod är den fria insynen i koden och att det finns en community som ständigt testar, uppdaterar och förbättrar mjukvaran. Dessutom kan kontroller av källkoden utföras för att få en bättre förståelse av underliggande mekanismer i koden. (Sache, 2010).

2.5.1 Ramverk för PHP

Tre av de största PHP-ramverken är Zend Framework, CakePHP och CodeIgniter. Nedan följer en jämförelse mellan de tre (Sache, 2010):

- CodeIgniter är det ramverk som är bäst dokumenterat, består av minst kod och är lättast att lära sig. CakePHP och Zend Framework är svårare att lära sig.
- CodeIgniter är snabbare än både Zend Framework och CakePHP. Det är dubbelt så snabbt som CakePHP och hela fem gånger snabbare än Zend Framework.
- Zend Framework erbjuder mest funktionalitet av de tre olika ramverken. CakePHP och CodeIgniter är likvärdiga när det gäller funktionalitet.

2.5.2 CodeIgniter

CodeIgniter är ett gratis, smidigt ramverk som används för att skriva bättre kod på mindre tid. Ramverket fokuserar på att göra grunderna inom webbutveckling så enkla som möjligt och fokuserar särskilt på att förenkla följande (Upton, 2007):

- Sessionshantering och hantering av cookies.
- Databashantering.
- Bygga HTML-dokument, så som websidor och formulär med validering från teckeninmatning.
- Testning.
- Kommunikation över Internet med FTP (File Transfer Protocol) eller XML-RPC (eXtensible Markup Language - Remote Procedure Call).

En funktion hos CodeIgniter är att systemet kopplar ihop controller-klasserna med URI:n. Nedan visas en exempel-URI.

```
example.com/projects/view_all/
```

Adressen ovanför tolkar systemet som att klassen “Projects” väljs och att metoden “view_all()” ska returneras.

Då MVC-metoden separerar logik från utseende behövs ett sätt att skapa modulära vyer för att visa upp information. I CodeIgniters bibliotek för att hantera vyer (views) går det att ladda in statiska sidor med tomma variabler som sedan ersätts med dynamisk information från modellen. Det går även att ladda in vyfiler i andra vyer vilket gör det enkelt att skapa moduler som återanvänds (EllisLab, 2012).

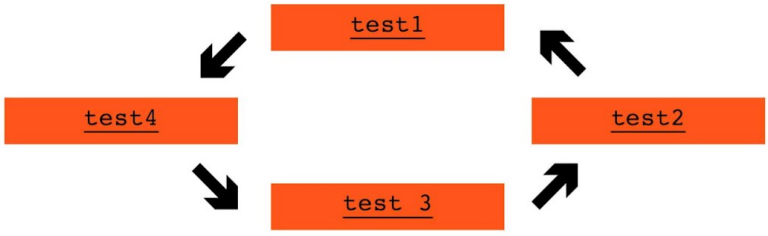
2.6 Databas

Det finns många olika databaser och även olika sorters databaser att välja mellan. Som exempel finns MySQL som utvecklas av företaget Oracle och är populärt bland privatpersoner då det är gratis för privat bruk och det finns en stor tillhörande community. En annan stor databas är MSSQL som utvecklas av Microsoft och fungerar enbart i windowsmiljön.

2.6.1 MSSQL

MSSQL används av många windowsanvändare. Tack vare en stor utvecklings-community finns mycket information att tillgå på Internet. En av nackdelarna med MSSQL är hur den hanterar relationer mellan olika tabeller. MSSQL har en begränsning angående tabeller som refererar till mer än en tabell. Då klarar den inte av att göra ON DELETE/UPDATE CASCADE, om det finns risk för så kallade cykliska vägar (Microsoft, 2012). Detta innebär att när det sker ändringar i den refererade tabellen ändras inte detta i tabellen som refererar. En annan nackdel med MSSQL är att det undermåliga stödet för bland annat PHP. För att jobba med PHP och MSSQL måste konfigurationsfilen för PHP ändras (PHP - MSSQL requirements, 2012).

item	parent
test1	test4
test2	test1
test3	test2
test4	test3



Illustrationen ovan beskriver ett exempel på en cyklisk relation

2.6.2 MySQL

En stor fördel med MySQL är att det är gratis för privatpersoner (Oracle, 2012). MySQL följer ofta med i färdigkonfigurerade webbmiljöer vilket medför enkel installation. Ett bra hjälpverktyg till MySQL är PHPMyAdmin. Verktyget hjälper till med att visualisera databasen och gör det enklare att utföra kommandon mot databasen istället för att gå via en terminal. PHPMyAdmin har några brister och då främst att det inte kan visualisera relationer mellan mer än två olika tabeller när nycklarna består av mer än en kolumn. MySQL fungerar bra ihop med PHP.

2.7 Grafiskt gränssnitt

Nedan följer en förklaring av riktlinjer som kan användas vid utveckling av grafiska gränssnitt. Kapitlet innehåller generella riktlinjer för gränssnittsutveckling, men även information om designmönster som kan användas i mer speciella fall.

2.7.1 Arkitektur

När en vy ses för första gången, fäster användaren blicken på olika ställen. Ett vanligt beteende är att användaren börjar med att titta i det övre vänstra hörnet och sedan låter blicken vandra ner mot det nedre högra hörnet. Därför är många hemsidor uppbyggda med menyer i övre och vänstra kanten. En ovan användare har ofta svårt att ta in mycket information på en gång. Det är viktigt för dessa användare att det inte presenteras för mycket information samtidigt. Enligt Tidwell (2005), bör den information som inte är viktig för just den vyn flyttas, tas bort, eller delas upp på flera vyer. Många designmönster bygger på att flytta användarens uppmärksamhet till det som är viktigt för stunden (Tidwell, 2005).

2.7.2 Navigering

På hemsidor och i andra applikationer är navigering mellan olika sidor en viktig del. Det är därför viktigt att en användare vet var i systemet den befinner sig. Det är också viktigt att användaren vet vart den ska ta sig därefter och hur den ska ta sig dit. Vana datoranvändare vill ofta ha många verktyg tillgängliga på samma gång. För mindre vana användare är det lämpligare att dela upp verktygen så att inte alla syns på en gång. Ibland behöver innehållet delas upp på flera sidor. Navigeringen på en sida bör alltid innehålla så få steg som möjligt (Tidwell, 2005).

2.7.3 Layout av sidoelement

För att en användare så snabbt som möjligt ska kunna hitta den information den är ute efter, kan en bra layout göra stor skillnad. Följande element kan integreras i sidan för att få en bra layout (Tidwell, 2005):

- **Visuell hierarki:** Den viktigaste informationen ska dra till sig användarens uppmärksamhet mest. Motsatsen gäller för den information som är minst viktigt.
- **Visuellt flöde:** Vägen användarens blick förflyttar sig genom gränssnittet. Bra gränssnitt har lokala fokuspunkter som först leder blicken till de element som är viktiga och därefter leder blicken till de element som är mindre viktiga. Fokuspunkter är punkter i gränssnittet som ögonen inte kan motstå att titta på. På så sätt vet användaren undermedvetet hela tiden vart den ska flytta fokus till.
- **Gruppering och placering:** Genom att gruppera element tolkar hjärnan det som att de är relaterade till varandra. Gruppering av element kan ske på följande vis:
    - Genom placera element nära varandra.
    - Använda samma form, storlek och färg.
    - Använda osynliga linjer som bildas när element läggs bredvid varandra.
    - Genom att forma geometriska figurer av mindre element, till exempel rektanglar.

  För att få bästa effekt bör dessa fyra metoder kombineras.

2.7.4 Inmatning från användare

När inmatning från användaren krävs är det viktigt att användaren förstår vilken typ av inmatning som förväntas och varför. Etiketter och annan text ska vara så pass tydlig att det inte kan misstolkas vad som menas (Tidwell, 2005).

Om det är möjligt bör det undvikas att fråga efter inmatning överhuvudtaget. Om det ändå behövs inmatning är det bra med förifyllda standardvärden eller information som systemet gissar med hänsyn till information användaren fyllt i tidigare. Detta kan minska den tid som behöver spenderas på att fylla i information (Tidwell, 2005).

Det kan vara svårt att komma ihåg alternativ som tillhör flervalsalternativ. Om användaren förväntas fylla i ett sådant alternativ är det bra att på något vis förmedla vilka alternativ som finns, till exempel genom en drop-down-lista. Om en användare förväntas skicka information enligt en viss formatering ska det framföras. Detta kan till exempel göras genom att ge bra standardvärden, dela upp inmatningen i flera rutor som motsvarar formatering på den förväntade inmatningen, eller ge en beskrivning på vad som förväntas (Tidwell, 2005).

2.7.5 Designmönster

Program är utvecklade för olika användarkategorier. Det finns applikationer som är utvecklade för avancerad användning som tillhandahåller mycket funktionalitet på samma gång. Detta kan vara förvirrande för en oerfaren användare, på grund av att programmet förutsätter att användaren redan har tillräckligt mycket erfarenhet för att använda det. Dessa program har ofta långa manualer och till och med kurser som lär ut hur programmet används. Avancerade användare känner oftast igen sig i gränssnittet och vill ha så få steg som möjligt för att nå önskad funktionalitet (Tidwell, 2005).

Enkla funktioner är mer användarvänliga och erbjuder färre valmöjligheter på samma gång. Användaren leds ofta genom uppgiften steg för steg. Det här innebär att användaren lär sig använda funktionen snabbt längs med vägen. De här användarna vill oftast spendera så lite tid som möjligt med att lära sig hur funktionen fungerar (Tidwell, 2005).

Oftast behöver program uppfylla kraven från båda användarkategorierna. Tidwell (2005) beskriver designmönster som kan användas för att möta båda användargruppernas behov. Designmönster används för att på ett strukturerat sätt designa ett gränssnitt på ett så bra sätt som möjligt ur användarens perspektiv. Nedan följer en lista på designmönster (Tidwell, 2005):

- **Global navigation:** En samling knappar eller länkar som återkommer på varje sida i systemet. Användaren kan med hjälp av dessa navigera sig till nyckeldelar av systemet från vilket sida som helst.
- **Breadcrumbs:** På varje sida i en hierarki visas vägen ner i hierarkin som leder ned till den sida användaren befinner sig på. Genom att klicka på de sidor som ligger ovanför i hiekarkin kan användaren ta sig tillbaka till den sidan.
- **Escape hatch:** På sidor med begränsade navigeringsmöjligheter finns en länk eller knapp som tar användaren tillbaka till en känd sida.
- **Visual framework:** Varje sida använder samma layout, färger och stil på element.
- **Center stage:** Det viktigaste innehållet på sidan ska använda den största delen av sidan.

- **Titled sections:** Innehållet på sidan separeras i olika sektioner med visuellt tydliga titlar.
- **Right/left alignment:** Vid design av formulär, eller tabeller, är etiketter till vänster högerjusterade och objekten till höger vänsterjusterade.
- **Liquid layout:** När fönstrets storlek ändras, ändras också innehållet på sidan så att sidan alltid är fylld med innehåll.
- **Button groups:** Händelser som är relaterade till varandra presenteras med hjälp av knappar som grupperade tillsammans.
- **Prominent "done" button:** En knapp som slutför en transaktion ska vara placerad i slutet av det visuella flödet på sidan. Knappen ska vara stor och ha en tydlig etikett.
- **Sortable table:** När data presenteras i en tabell ska det gå att sortera datan efter värdena i de olika kolumnerna.
- **Autocompletion:** När en användare skriver i en textruta ska möjliga svar visas i en lista och automatiskt slutföra inmatningen när det är möjligt.
- **Illustrated choices:** Bilder används istället för, eller som komplement till, text för att visa möjliga val.
- **List builder:** Två listor visas jämte varandra där objekt kan flyttas mellan listorna.
- **Good defaults:** Så fort det är möjligt är fälten i formulär förifyllda med värden system anser är bra gissningar till vad användaren vill fylla i.
- **Same-page error message:** När oönskade värden fylls i i ett formulär ska ett felmeddenade dyka upp på samma sida.
- **Few hues, many values:** Gränssnittet har få huvudfärger. Övriga färger väljs genom att ändra ljusstyrka på huvudfärgerna.

2.8 Färgschema

Inom färglära används färgscheman för att beskriva ett antal färger som passar ihop. Scheman består av neutrala färger och accentfärger. Svart, vitt, grått, brunt och beige är neutrala färger. Accentfärger är starkare färger med hög kulörthet som rött, blått, grönt, gult, lila.

För att en layout ska bli tydlig behövs accentfärger. De medför att viktiga element blir tydligare för användaren och skapar en känsla av energi och rörelse. En typisk accentfärg är rött som drar till sig en användares uppmärksamhet. Orange tar användarens uppmärksamhet utan att vara lika överväldigande som rött. Blå associeras med företag och ger en seriös känsla så länge den inte blir för ljus i kombination med hög färgmättnad (Chapman, 2010).

Av de neutrala färgerna är svart den starkaste. Den är bra i kombination med vitt för att få hög kontrast. Neutrala färgskalor används för att få ett sofistikerat färgschema. Grå färgscheman används på företagshemsidor för att ge en formell, professionell och sofistikerad känsla (Chapman, 2010).

3 Metod

I det här projektet har olika metoder använts inom områden som planering, utveckling och teori. Metoderna har givit arbetet ett mer effektivt och strukturerat arbetsflöde. Nedan beskrivs metoder som är relevanta för projektet på ett övergripande sätt.

3.1 Planering med hjälp av Ganttschema

Ett Ganttschema är ett stapeldiagram som används till att illustrera olika projektaktiviteter, relationerna mellan dem samt deras start- respektive slutdatum (Randolph & Posner, 1988). Detta är en av de äldsta och mest populära metoderna för schemaläggning av projekt (Ballakur & Steudel, 1984). Ett Ganttschema ger en projektledning kontroll över ett projekts faser och eventuella förseningar kan upptäckas på ett tidigt stadium (Wilson, 2003).

3.2 Utveckling

Utveckling av mjukvara innehåller många moment. För varje moment finns det olika metoder och tillvägagångssätt. Detta kapitel behandlar arbetssätt som använts i det här projektet.

3.2.1 Brainstorming

Brainstorming är en teknik som används för att få människor att våga komma på djärva idéer. Ett givet problem ges till en samlad grupp, därefter får alla i gruppmedlemmar lägga fram så många idéer de möjligen kan komma på. Att spinna vidare på andras idéer uppmuntras. Inga idéer får anses som konstiga, inga negativa kommentarer får ges och inga negativa tankar får uppstå. Varje idé antecknas av en sekreterare och efter en given tid passerat avslutas mötet. Nästa dag sorteras idéerna upp efter kategorier och sedan utvärderas idéerna. Då är det tillåtet att ge negativa kommentarer eller att avslå idéer, dock måste bra argument för detta kunna ges (Clark, 1958).

Fördelar som fås genom att använda brainstormingmöten jämfört med vanliga möten är följande (Clark, 1958):

- Ett vanligt möte har ofta en chef som gruppmedlemmar vill göra ett gott intryck på. Rädslan att nämna en djärv idé som kan få gruppmedlemmen att se dålig ut inför chefen kan ofta hindra denne från att säga något alls. Med ett brainstormingmöte uppmuntras även annorlunda idéer och tänk ”utanför lådan”.
- Vanliga möten är inte demokratiska då chefens åsikt väger tyngst. Vid ett brainstormingmöte avfärdas inte en idé bara för att ”fel” person inte tycker om den.
- Vanliga möten kan ha en negativ atmosfär som kväver nya idéer. När brainstorming används uppmuntras nya idéer vilket leder till att fler blir producerade.

Även om gruppbrainstorming har en mycket högre produktion av idéer än vanliga möten har, så är produktionen lägre än vad den skulle kunna vara. Om alla personer i en grupp kör brainstorming med sig själva så kan den totala mängden producerade idéer uppnå dubbelt av vad gruppbrainstorming kan (Paulus et al, 1993).

3.2.2 Mock-up

Mock-ups tillhandahåller ett effektivt sätt att skapa och visa upp idéer. Det är en metod för att skapa grova skisser som visar essensen i ett gränssnitt. De flesta projekt gagnas av att använda mock-ups tidigt i designprocessen (Mads Soegaard 2004).

3.2.3 User-centered design

User-centered design innebär att utvecklingen utgår från verkliga användares behov och mål, inte bara uppgiften som ska utföras. Det betyder inte att användaren är den som bestämmer över utvecklingen utan innebär att produkten anpassas efter en användargrupp allteftersom systemets byggs upp. Som följd blir systemet användarvänligt och funktionerna relevanta till uppgiften som ska utföras (Rogers, Sharp & Preece, 2011).

3.2.4 Plandriven och inkrementell process

En mjukvaruutvecklingsprocess är en utvecklingsmetod av mjukvara som involverar aktiviteter så som specifikation, implementation, validering och utveckling av mjukvara. Målet med att använda sig av mjukvaruutvecklingsprocesser är att få en högre kvalitet samt en snabbare produktion av sin mjukvara (Sommerville, 2011).

Ian Sommerville (2011) beskriver flera olika utvecklingsprocesser som går att använda vid planering av programvaruutveckling. Det går att kort sammanfatta dessa till två utvecklingsprocesser, den plandrivna samt den inkrementella. Den plandriva processen innebär att specificerade krav ställs i förväg hos ett system innan implementering påbörjas. Den inkrementella processen innebär att slutsystemets krav är kända i början och att flera versioner byggs av systemet för att nå dessa. Varje version innehåller en mindre mängd ny funktionalitet och inför varje version skapas ett kravdokument med de krav som ska ställas på nästa version av systemet.

3.2.5 Evolutionär process

Utöver den plandrivna och den inkrementella processen beskriver Per Zaring (2011) en tredje process i kursen “Grundläggande software engineering”, den evolutionära processen. Processen liknar den inkrementella processen, men skiljer sig genom att en del av systemets slutkrav är okända i början av projektet. Därför måste systemet analyseras vid varje iteration för att specificera nya krav och finna ny önskvärd funktionalitet. Eftersom en evolutionär process liknar en inkrementell process på alla punkter förutom att systemets krav inte är kända från början, kommer Sommervilles (2011) uttalanden om inkrementella processer i fortsättningen att appliceras även på evolutionära processer.

Sommerville (2011) nämner att det är det lättare att anpassa sig till kravförändringar hos ett system med en inkrementell/evolutionär process. Det beror på möjligheten att visa upp en prototyp för kunden och anpassa sig tidigt i utvecklingsfasen efter kundens behov. En prototyp är bättre än diagram och/eller textbeskrivningar vid analys av ett användargränssnitt, detta beror på användargränssnitts dynamiska natur. Inkrementella/evolutionära processer lämpar sig särskilt väl för system som till exempel e-handelssidor samt snabbutvecklade system.

Att använda en inkrementell/evolutionär process passar inte till alla projekt. Här nämns ett antal fall där en plandriven process passar bättre:

- Vid uppbyggnad av stora system med utvecklingsteam spridda över ett stort område.
- Inbyggda system där mjukvaran beror på hårdvarans prestanda.
- Kritiska system där krav måste analyseras för att inte störa säkerheten hos ett system.

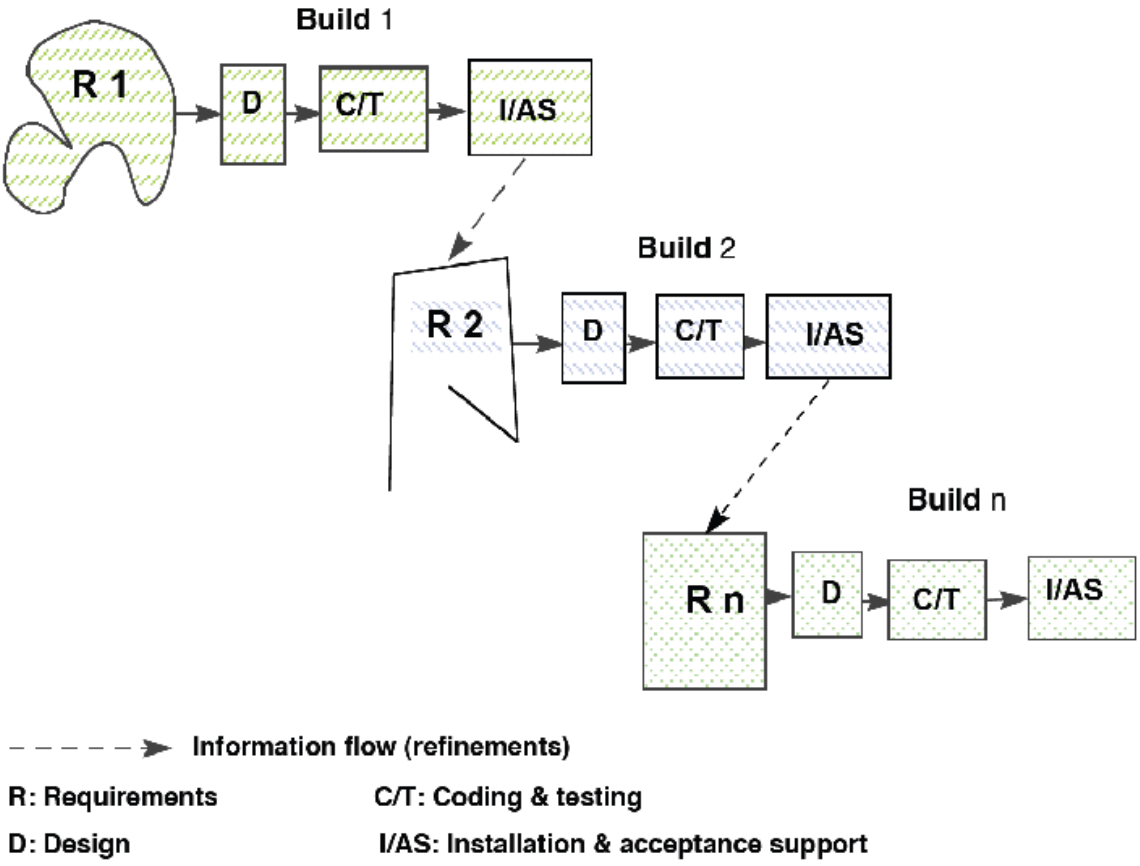


Fig. 1 Evolutionär process. Ovan illustreras den evolutionära utvecklingsprocessen. Varje "build" börjar med skapandet av en kravspecifikation på kommande version av mjukvaran. Därefter designas, kodas, testas och implementeras mjukvaran till systemet innan versionen visas för kunden (Zaring, 2011).

Det finns två olika tillvägagångssätt för inkrementella processer. Det första är att använda sig av ett plandrivet tillvägagångssätt där det från början bestäms precis vilka krav varje prototyp ska uppfylla. Det andra är att använda sig av ett mer agilt tillvägagångssätt där kraven bestäms för nästa prototyp efter att föregående prototyp utvecklats och kontrollerats. För den evolutionära processen kan endast agila tillvägagångssätt användas. Det beror på att alla krav inte är definierade från början vilket är ett krav vid ett plandrivet tillvägagångssätt (Sommerville, 2011).

3.2.6 Extremprogrammering

Extremprogrammering beskriver Sommerville (2011) som ett agilt tillvägagångssätt. Metoden använder exakta deadlines där prototyper och slutprodukt måste släppas på vissa datum. Om inte all funktionalitet hinner implementeras innan utsatt datum ska funktionalitet utelämnas. Utelämnad funktionalitet läggs tillbaka till listan över ej avklarade krav. Denna lista prioriteras sedan vid planeringen av nästa version. Extremprogrammering förespråkar parprogrammering och att kundens versionskrav bryts ner till delkrav under planeringsfasen. Dessa delkrav ska ta mellan 4 och 16 timmar att implementera och implementeras sedan iterativt.

3.2.7 Parprogrammering

Parprogrammering innebär att utvecklare jobbar i grupper om två när de utvecklar mjukvara. Grupperna bör inte vara fasta utan medlemmarna bör bytas ut med jämna mellanrum (Sommerville, 2011).

Fördelarna med att programmera i par är enligt Sommerville (2011):

- Kollektivet, snarare än individerna, äger koden. Eventuella fel i koden blir då gruppens ansvar vilket medför att hela gruppen tar ett större ansvar för koden.
- Skriven kod får en omedelbar granskning av den som för tillfället inte skriver. Tester har visat att granskningar av kod är mycket effektiva för att hitta fel, dock brukar de vara ganska dyra att genomföra. Parprogrammering eliminerar det problemet då granskningen sker parallellt när koden skrivs.
- Större chans att kod optimeras, då gruppen direkt använder sig av koden som skrivs är det större chans att stopp görs för att optimera kod.

Ett vanligt antagande angående parprogrammering är att eftersom bara en i ett par kan skriva åt gången minskar effektiviteten inom gruppen. Detta är visserligen sant för programmerare med stor erfarenhet (Arisholm et al, 2007; Parrish et al, 2004), men det gäller ej för oerfarna programmerare, till exempel studenter. Där är det visat att produktiviteten förblir densamma (Cockburn & Williams, 2001; Williams et al, 2000).

3.3 Litteraturstudie

En litteraturstudie utförs för att finna information som ligger utanför en persons kunskapsområde och för att finna djupare kunskap inom befintliga områden. Litteraturstudier kan hjälpa till med att besvara frågeställningar eller stödja misstankar inom ett ämne.

4 Genomförande

Projektet har bestått av olika områden som gränssnittsutveckling, back-end-programmering och databasdesign. Arbetet har även bestått av en planeringsfas där projektets ramar diskuterades fram. Det här kapitlet redogör den arbetsprocess som använts under projektets gång.

4.1 Förstudie

Projektet påbörjades med en studie för att ta reda på vilka krav som ställdes på prototypen, hitta ett gemensamt mål och för att förstå företaget. Studien började med att skapa de problembeskrivningar som behövdes. Brainstorming användes för att komma fram till vad som skulle behövas i systemet och vilka problem som skulle lösas. För att göra problemen mer exakta hölls kontinuerlig kontakt med företaget för att höra om deras syn av projektet. Problembeskrivningarna jämfördes och resultatet blev en gemensam syn av projektets mål.

Ett studiebesök hos Sund Birsta i Sundsvall genomfördes för att få en bättre förståelse om vad företaget gör och vilka problem som en kundportal skulle kunna lösa. Det gjordes ett studiebesök på en montageavdelning och ett urval av de maskiner företaget säljer introducerades. Därefter utfördes ett möte med projektansvariga från Sund Birsta där de fick förklara sina önskemål.

En stor del av mötet gick åt till att diskutera hur Sund Birstas databas skulle hanteras. Databasen innehåller all relevant information för projektet, men är svårtolkad och stor. Det beslutades att implementeringen av kopplingen mellan Sund Birstas databas och kundportalen skulle läggas till i listan med avgränsningar.

4.1.1 Arbetsplanering

Resultatet från förstudien användes när projektet skulle planeras. Planeringen ledde till en plan som följdes under projektets gång. Denna plan gestaltas som ett Ganttschema som bifogas i bilaga B.

Mjukvaruutvecklingsprocess
Olika utvecklingsprocess passar ett projekt, beroende på faktorer som systemets storlek, utvecklingsgruppens storlek och kontakten mellan utvecklarna. Enligt förstudien stod det klart att systemet skulle bli litet men komplext, innehållande likheter med en e-handelssida. Det bestämdes att projektgruppen skulle hålla kontakten på regelbunden basis. Evolutionär utveckling av systemet valdes då på grund av projektets storlek samt utvecklingsgruppens nära kontakt. Dessutom valdes metoden på grund av att det saknades en komplett kravlista för systemet innan projektets start.

I och med att en evolutionär process valdes för utveckling av systemet fick utvecklingen av systemet ske med hjälp av ett agilt tillvägagångssätt. Agila tillvägagångssätt lämpar sig väl för system som liknar det som byggs i detta projekt samt när kunden är villig att vara involverad i utvecklandet av systemet.

Extremprogrammering
Extremprogrammering valdes som metod vid implementeringen av prototypen.
Det gjordes på grund av att metoden överensstämde med projektets parametrar, det fanns ett exakt slutdatum där slutrapport skulle lämnas in samt ett antal bestämda möten med Sund Birsta där prototyper skulle visas upp för feedback. Extremprogrammering förespråkar parprogrammering, en metod som användes i projektet.

4.1.2 Uppdelning

Projektet delades upp i olika områden för att sprida ut utvecklingen på flera personer. Områdena bestod av teori, mock-up, statiska sidor, databas, implementering av systemet och rapportskrivande.

4.2 Gränssnitt

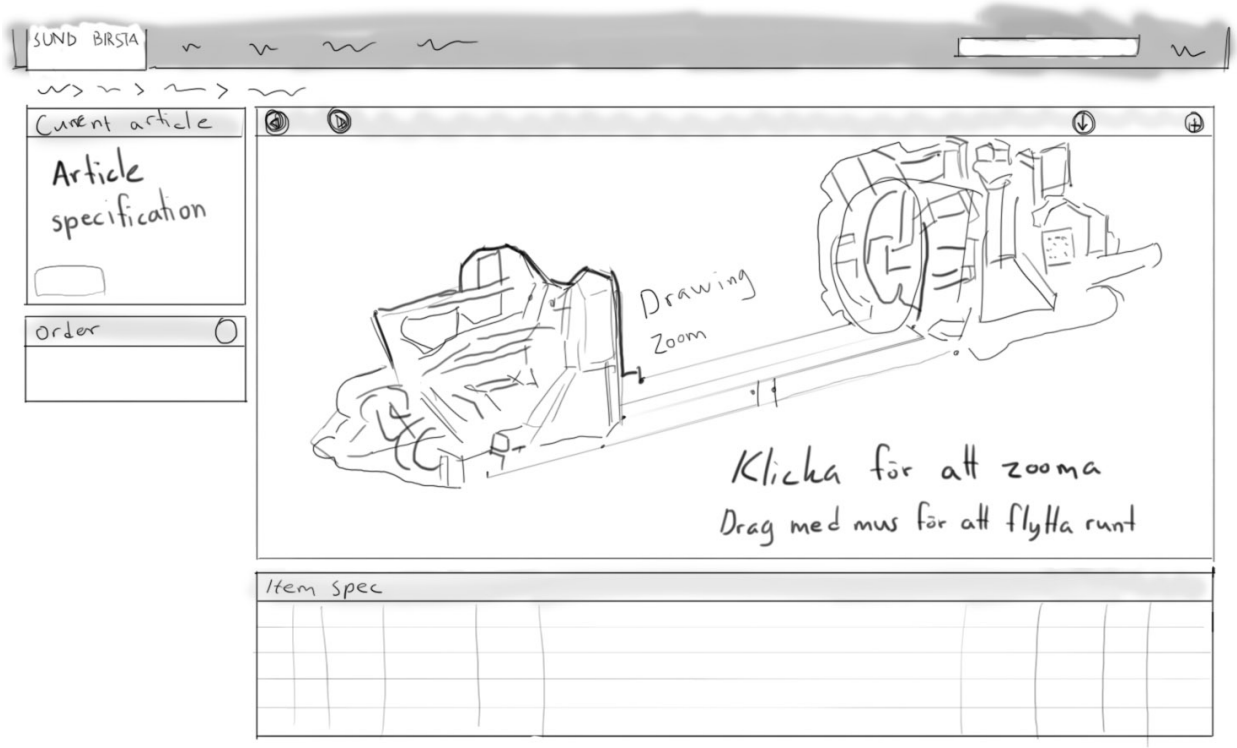
Ett av de tidigaste delmålen som sattes upp var att skapa ett användarvänligt system. Personal med liten datorvana skulle kunna använda systemet utan att gå någon kurs. User-centered design användes vid skapandet av gränssnittet vilket skapade en god grund för utvecklingsarbetet. Varje funktion implementerades så den var lätt att förstå och effektiv att använda.

4.2.1 Designmönster

Under skapandet av mock-upen, användes Jenifer Tidwells bok *Designing Interfaces*. I boken beskriver Tidwell 94 designmönster som hon rekommenderar att använda vid gränssnittsdesign. När de olika mock-uperna skapades kunde passande designmönster från boken identifieras och på så vis skapa ett bättre gränssnitt för användaren. De designmönster som användes beskrivs i teoridelen.

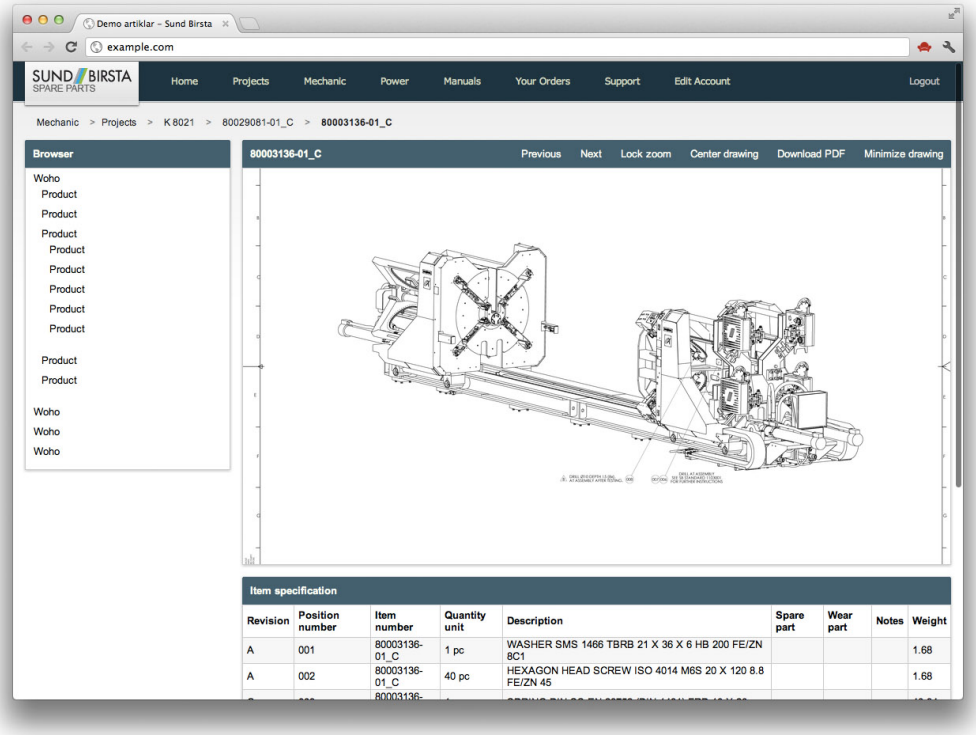
4.2.2 Mock-up

Skisser skapades på papper och whiteboard för att beskriva gränssnittets olika vyer. Det gjordes för att grunda en gemensam syn av systemets utseende. Feedback på idéerna utväxlades för att vidareutveckla dem. Tidiga versioner av den mock-up som skapades utfördes med penna och papper för att erhålla ett högt utvecklingstempo. Skisserna utgick från förstudiens resultat och information som erhållits från tidigare möten.

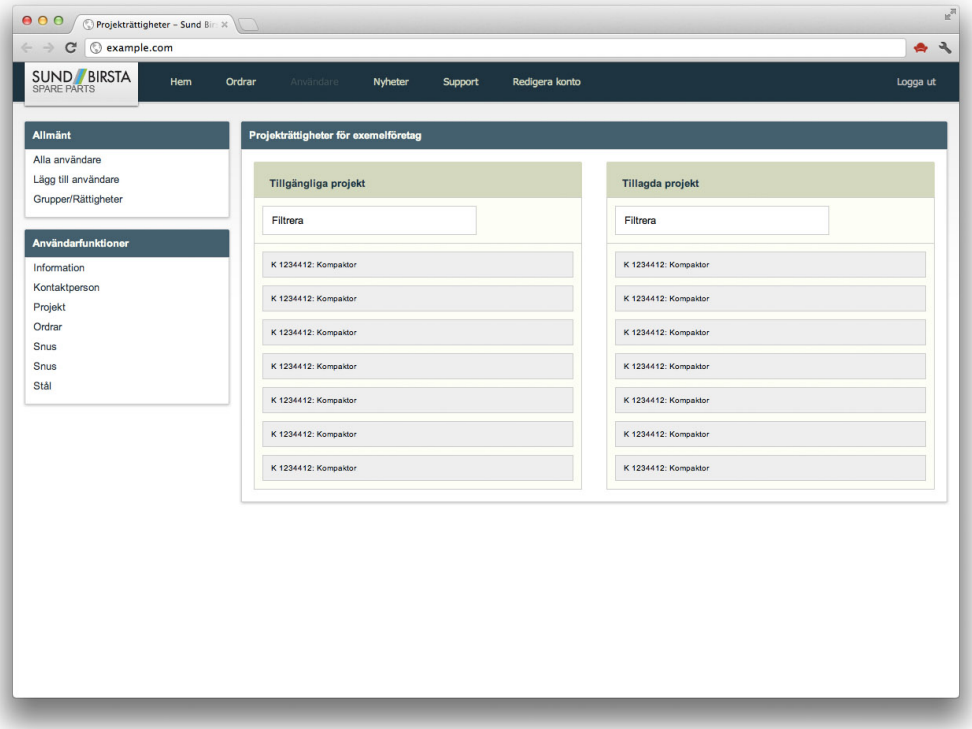


Mock-up av projektutforskaren

När skisserna var färdigställda, användes de som grund till statiska HTML-sidor. HTML-sidorna byggdes upp av generella kodelement som återanvändes i hela systemet. Dessa HTML-sidor följde vedertagna webbstandarder för att webbläsare skulle tolka dem korrekt.



Tidigt HTML-dokument innehållande projektutforskare.

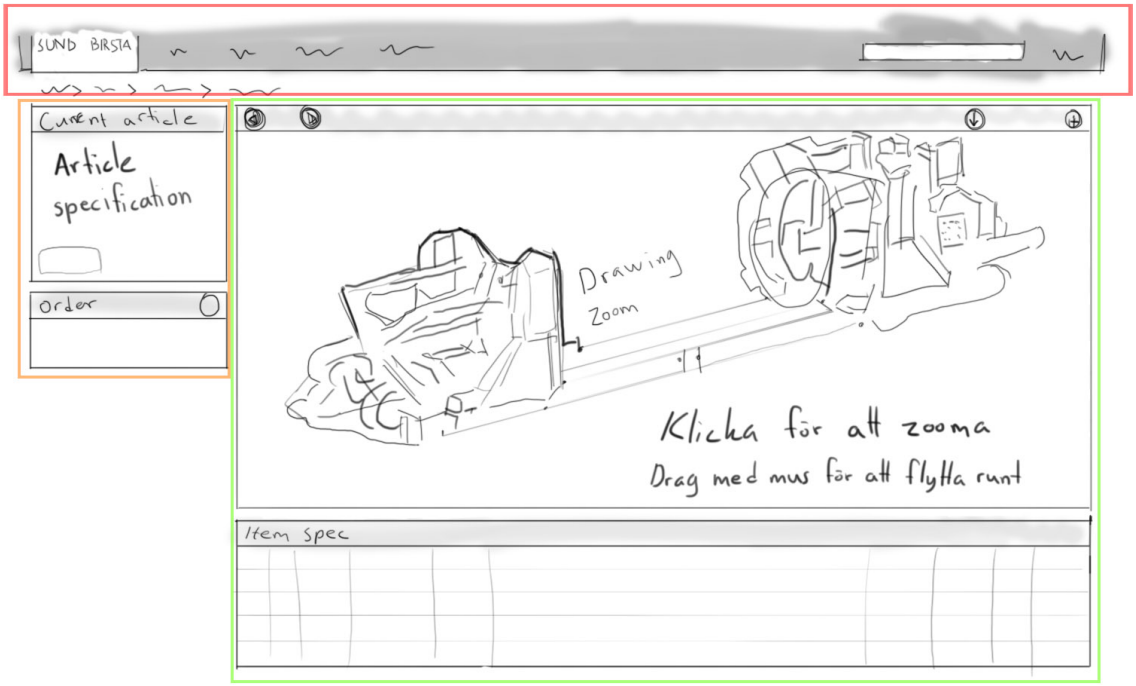


Tidig version av rättighetshantering

De statiska HTML-sidorna visades upp för Sund Birsta som ett förslag på hur hemsidan skulle kunna se ut. Sund Birsta gav synpunkter och lämnade förslag på vad som skulle läggas till eller ändras.

4.2.3 Separerade vyer

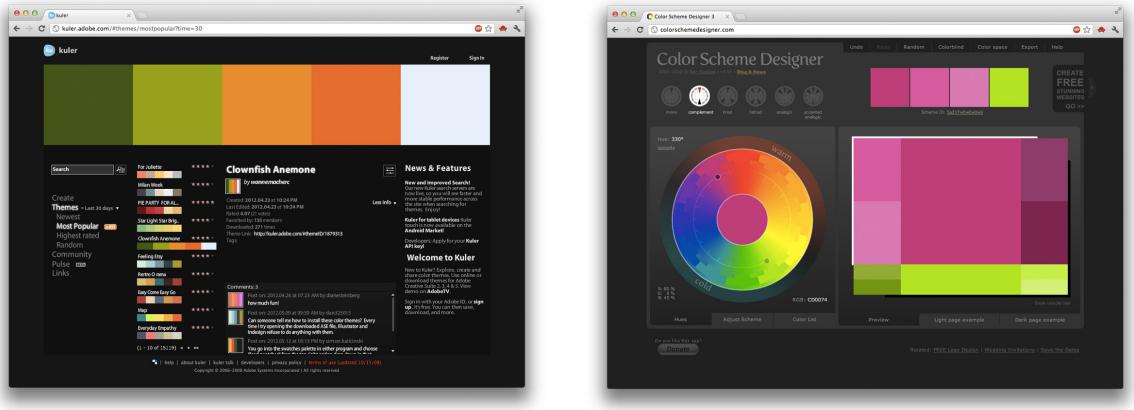
Generella moduler skapades bestående av olika visuella element för att åstadkomma enhetlighet i gränssnittet. Den mock-up som skapats tidigare analyserades för att få fram vilka element som skulle separeras. Eftersom toppmenyn planerades att användas på varje sida kunde den separareras till en separat modul.



Projektutforskaren skapades i tre olika delar: Toppmenyn, sidomenyn och utforskaren.

4.2.4 Färgschema

Vid utvecklandet av gränssnittet lades tid på att skapa ett färgschema som skulle ge ett professionellt intryck för användare och en angenäm användarupplevelse. Hjälpverktygen *kuler.adobe.com* och *colorschemedesigner.com* användes för att skapa passande färgscheman. De olika färgerna testades på de befintliga statiska HTML-dokumenten med hjälp av stilmallar, för att finna den bästa kombinationen av kontrast och tydlighet. För att få fram ett passande färgschema, bestående av kulörer som passar varandra och dessutom ger ett professionellt intryck, utfördes ett antal muntliga utfrågningar där olika scheman jämfördes.



Verktyg som användes för att ta fram färgschemat

Först skapades en mängd grova färgsättningar till sidan för att få en övergripande bild av sidan. Dessa kan liknas med mock-up-processen för systemets utseende. Färgerna utgick från Sund Birstas nuvarande hemsida och logotyp. Det gjordes i ett försök att skapa en gemensam färgsättning som skulle tydliggöra kopplingen mellan kundportalen och Sund Birsta. För att få ett lugnt intryck av sidan, användes neutrala färger som huvudfärger.

Eftersom den neutrala färgskalan hade en lugnande effekt var en röd färg för stark som accentfärg. Därför gick fokus över till en orange kulör som ersättare till det röda. Orange är bra på att få användarens uppmärksamhet utan att vara lika överväldigande som rött. När grundskalan var komplett ordnades ett antal kompletteringsfärger för att skapa bättre dynamik på sidan. Det sista som skapades var meddelandefärger för att visa olika nivåer av säkerhetsmeddelanden.

4.2.5 Navigering

Ett av projektets mål var att skapa ett användarvänligt system där en lättanvänd navigering var av stor vikt. I enlighet med den tidigare skapade statiska prototypen användes en enhetlig toppmeny med samma funktionalitet på alla sidor. Det fanns även en så kallad “breadcrumb”-navigering som låg under toppmenyn som indikerade var användaren befann sig i systemet. Den fick god feedback av Sund Birsta och implementerades därför i CodeIgniter.

Bread Crumb
I toppen av sidan placerades bread crumb-navigering för att tydligt visa var användaren befinner sig och på ett enkelt sätt kunna navigera sig uppåt i hierarkin. Bread crumb-navigeringens träd växer för varje nivå som användaren går ner i trädstrukturen.

Sökfunktion
En av de mest eftersökta funktionerna hos systemet var en välfungerande sökfunktion. För en kunnig användare medför en sökfunktion ett mer effektivt arbetsflöde, då användaren slipper gå ner i projektstrukturen för att välja en artikel.

Det största problemet som uppkom var att finna sökvägen till en artikel. Sökfunktionen skulle först finna artikeln i databastabellen, kontrollera om användaren har tillgång till artikeln och därefter leta sig upp till ett projekt artikeln tillhör. Om artikeln finns i fler än ett projekt ska sökfunktionen välja det första den finner. I systemets toppmeny skapades en sökruta i det övre högra hörnet för konstant åtkomst av funktionen, oavsett var användaren befinner sig.

Funktionsknappar
I systemet placerades ett antal funktionsknappar. Dessa kan beskrivas som funktioner som inte leder till en ny plats i systemet. De utför istället en uppgift och för sedan användaren tillbaka till platsen där den var innan. Länkarna utformades i form av cirklar med ikoner som tolkar funktionerna. Varje ikon skapades i en svart cirkel som därefter dämpades till mörkgrå genom att sänka opaciteten i stilmallen.

4.3 Systemarkitektur
Ett mål var, som skrivet ovan, att ett annat team ska ha möjlighet att ta över utvecklingen av systemet efter att prototypen är klar. Därför ansågs det som olämpligt att skapa en egen arkitektur. Det sågs som mer lämpligt att använda ett känt designmönster för att göra det lättare att sätta sig in i koden. Dessutom var det ont om utvecklingstid för projektet, därför bestämdes det att ett känt designmönster skulle användas som systemarkitektur.

Viktigt för projektet var att användare ska se olika vyer av systemet beroende på användartyp. Därför var det av yttersta vikt att systemet byggdes enligt en arkitektur som möjliggör att presentation samt hantering av data enkelt kunde ändras utan att påverka den andra. En systemarkitektur som klarar av just detta, tack vare att funktionalitet, databas och utseende är skilt från varandra, är MVC. Dessutom ansågs det som fördelaktigt att använda sig av MVC eftersom det är den arkitektur som begränsar valet av ramverk minst. Detta eftersom ramverk för webbutveckling, som skrivet ovan, oftast stödjer MVC.

4.4 Back-end

För implementeringen av prototypen behövdes ett programmeringsspråk och ett ramverk väljas. Att skapa systemet i ASP ansågs som en dålig idé eftersom det förvalda språket till ASP är VBscript (Visual Basic Scripting Edition) och som teorin nämner, är det generellt sett är svårare att lära sig än till exempel Perl eller PHP. ASP är dessutom inte plattformsoberoende. Då prototypen inte skulle begränsas till en viss plattform beslutades att ASP inte skulle användas.

CFML blev också det uteslutet ur valet av programmeringsspråk i ett tidigt stadium. Detta eftersom CFML, likt ASP, inte är gratis att använda och det inte fanns några ekonomiska resurser till det här projektet. Då systemet liknade en e-handelssida var systemet, enligt teorin, som gjort för att implementeras i PHP. Det fanns dessutom erfarenhet av PHP inom projektgruppen vilket sågs som en ytterligare fördel med språket.

4.4.1 Ramverk

Att använda sig av ett ramverk ansågs i början av projektet som en självklarhet. Detta eftersom de vedertagna konventioner och den enhetliga kod som detta leder till är ett måste ifall en framtida utvecklingsgrupp skall kunna dra nytta av detta projekt. Valet av ramverk begränsades av två faktorer, dels att ramverket skulle stödja MVC-modellen och dels att det skulle vara uppbyggt av öppen källkod och vara gratis av ekonomiska skäl.

Den viktigaste egenskapen hos ett ramverk till detta arbete var att ramverket skulle vara enkelt att förstå, då tiden för implementering var knapp och minimalt med tid skulle spenderas på att lära sig ett ramverk. Därefter ansågs prestanda som den näst viktigaste egenskapen och slutligen sågs funktionaliteten som den tredje viktiga egenskapen.

Utifrån dessa egenskapskrav ansågs det, från jämförelsen skriven ovan, lämpligt att använda CodeIgniter. Det sågs även som positivt att CodeIgniter inte har lika strikta konventioner som till exempel CakePHP vilket ytterligare förenklar användningen av ramverket.

4.4.2 Implementering

För att få fram den önskade funktionaliteten hos kundportalen skapades systemet med hjälp av CodeIgniter. Ramverket gjorde det enkelt att använda MVC-metoden utan att använda för strikta konventioner. Sidan byggdes upp med hjälp av ett antal controller-filer som symboliserade systemets olika funktioner. Varje controller-fil innehöll en klass med ett antal funktioner. Ramverkets adresshantering, där URI:n tolkas för att finna korrekt klass och metod medförde ett enkelt sätt att skapa en bra struktur av koden.

Under utvecklingen av prototypens back-end användes parprogrammering för att säkerhetsställa god kvalitet och optimering av koden. Utvecklingen utfördes evolutionärt där nya funktioner lades till efter korta faser.

Implementeringen påbörjades med en projektutforskare för att göra det möjligt att se olika artiklar. Projektutforskaren användes därefter som utgångspunkt för den vidare utvecklingen. Ett ordersystem flätades samman med projektutforskaren för att göra det möjligt att skapa orderförfrågningar.

Runt hela systemet skapades ett inloggningsskydd för säker åtkomst. Skyddet består av en superklass som alla andra klasser i systemet ärver. När klassens konstruktor anropas kontrolleras användarens behörighet.

4.5 Databas

Vid valet av databas till systemet fanns ett flertal faktorer som var viktiga att ta ställning till. Den främsta var att Sund Birsta redan använde sig av Microsofts SQL-lösning, Microsoft SQL Server. Den faktorn gjorde att MSSQL valdes även om andra databaser kunde gjort utvecklingsarbetet lättare då det inte hade varit något problem med ON UPDATE/DELETE CASCADE. Prisfaktorn hjälpte även MSSQL då Sund Birsta redan investerat i plattformen och byggt upp en tillhörande IT-infrastruktur. MSSQL har en bra utvecklings-community som gör felsökningar och faktaletande enkelt, då främst deras egna MSDN-sida. En nackdel som upptäcktes under utvecklingen är dess dåliga stöd för relationer mellan olika tabeller i kombination med regler för uppdateringar och borttagningar av rader. Även om MSSQL har en del brister valdes det på grund av en befintlig plattform hos Sund Birsta och den ekonomiska faktorn. Det visade sig att MSSQL behövde mycket mer arbete än beräknat. Därför användes MySQL parallellt med MSSQL för att göra de första utkasten av databasen.

4.5.1 Databasens uppbyggnad

Arbetet med databasen påbörjades med att gå igenom Sund Birstas önskemål om vad de ville ha med för information på portalen. Efter det var nästa steg i skapandet av en fungerande struktur att skapa ett ER-diagram. I diagrammet beskrevs alla ingående delar som skulle finnas med i systemet. Detta gjordes eftersom ER-diagram ger en bra överblick över hur strukturen ska se ut. Efter många iterationer av ER-diagrammet där det för varje iteration lades till eller ändrades i strukturen påbörjades jobbet att översätta diagrammet till SQL-kod.

Det bestämdes att MySQL skulle användas parallellt med MSSQL vilket innebar att två olika versioner av SQL-koden skrevs. Det berodde på att i visa fall är det små skillnader i koden mellan MySQL och MSSQL. I ett senare skede övergavs MySQL helt till förmån till MSSQL då slutprodukten skulle innehålla MSSQL.

För att komma runt problemet med ON UPDATE/DELETE CASCADE i MSSQL används triggers för att göra alla uppdateringar och borttagningar manuellt. Det medförde en del extra arbete då varje tabell måste gås igenom och se om den finns refererad i någon annan tabell.

Det skapades ett antal vyer för att göra de mer komplicerade anropen mot databasen. Detta gör att databaslogik flyttas från back-enden till databasen. På detta sätt blir det enklare för webbutvecklaren att använda sig av databasen.

4.5.2 Säkerhet

För att kunna höja säkerheten ytterligare i portalen skapades ingen direkt koppling mellan Sund Birstas databas och kundportalens databas. På detta sätt riskerar inte Sund Birsta att någon hemlig information kan läcka ut då systemet inte innehåller någon sådan.

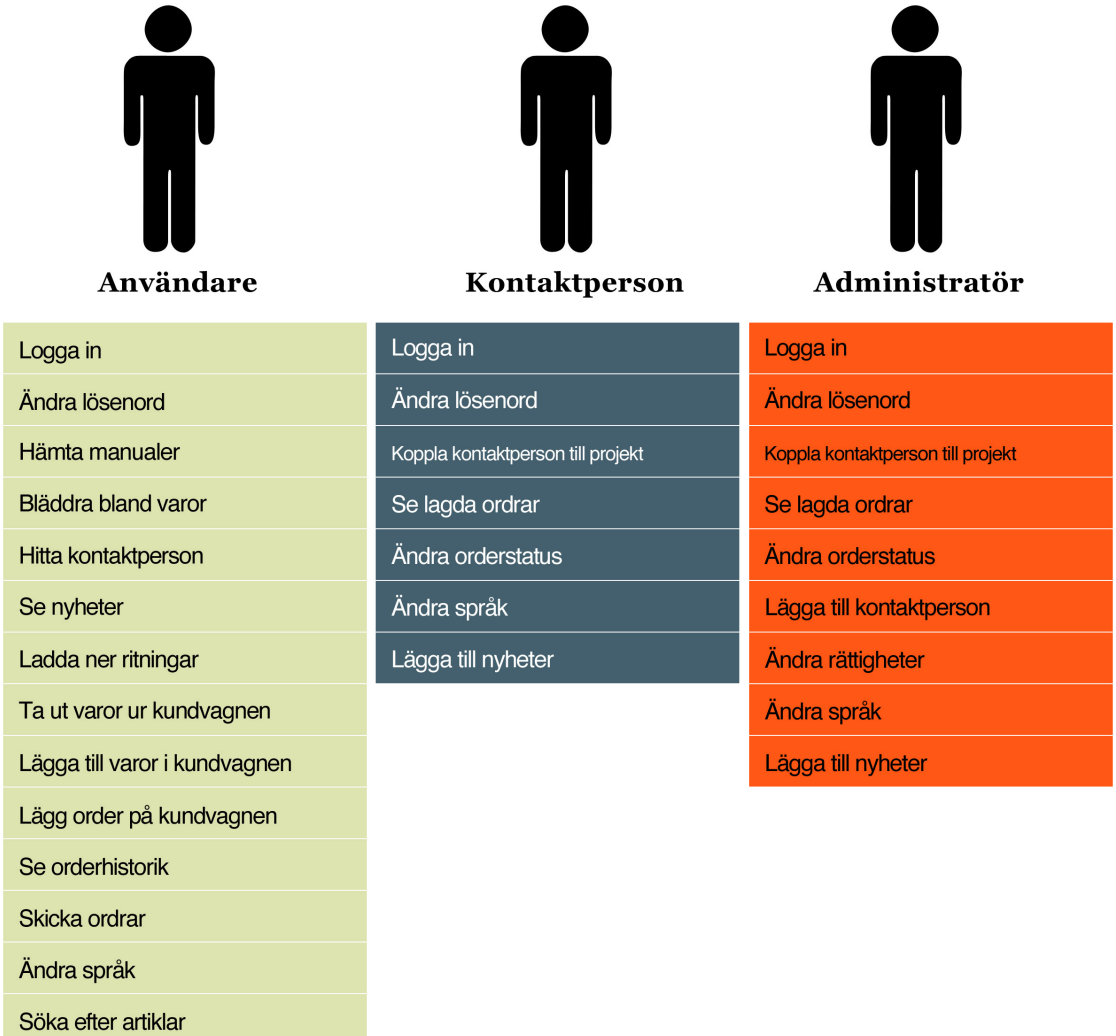
En annan åtgärd som togs för att höja säkerheten och dessutom även användarvänligheten, är att kunder bara kan se de reservdelar de ska ha tillgång till. Om till exempel en kund bara äger en maskin A ska det inte vara möjligt för just den kunden att kolla på delar som hör till maskin B på grund av säkerhetsskäl och önskemål från Sund Birsta.

5 Den färdiga prototypen

Resultatet av projektet är en prototyp som uppfyller Sund Birstas önskemål. Systemet fungerar, men saknar möjlighet att fyllas på med information på grund av projektets avgränsning angående export av data från Sund Birstas databas. Prototypen innehåller övergripande funktionalitet för användare av kundportalen, men saknar administrativ funktionalitet.

5.1 Förstudie

Resultatet av studien blev, förutom en djupare förståelse för systemet också en insikt om att systemet skulle bli litet men komplext och att det skulle ha flera likheter med en e-handelssida. Dessutom togs en lista fram med funktionalitet som systemet ska ha. Funktionaliteten kategoriserades sedan efter vilka sorters användare som skall ha tillgång till den. Nedan följer en bild som sammanfattar de funktionella krav som ställdes på systemet och i bilaga A finns en kravspecifikation som mer i detalj beskriver kraven:



Framtagna användarkrav

5.2 Gränssnitt

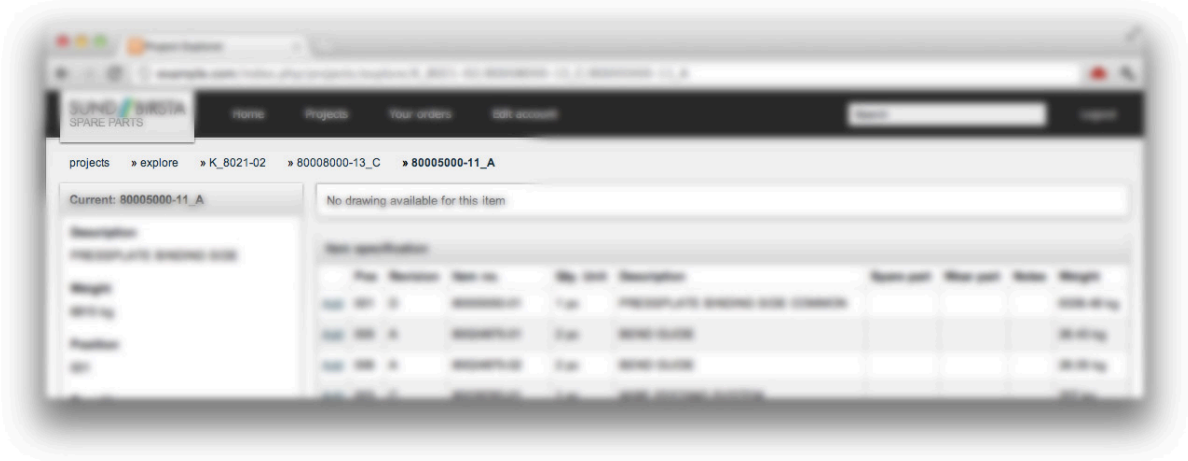
Prototypen har ett enhetligt gränssnitt uppbyggt av olika moduler. Gränssnittet har ett neutralt färgschema med gröna och blåa inslag tillsammans med orange som accentfärg. Nedan beskrivs de olika modulerna som utvecklats i detalj.

5.2.1 Navigering

Navigeringen i den implementerade prototypen hjälper användare ta sig runt mellan de olika sidorna i systemet. Navigeringen är uppbyggd av olika delar som alla hjälper användaren att navigera runt bland de olika funktionerna. Fokus har legat på användarvänlighet och tydlighet vid utvecklingen.

Bread Crumb

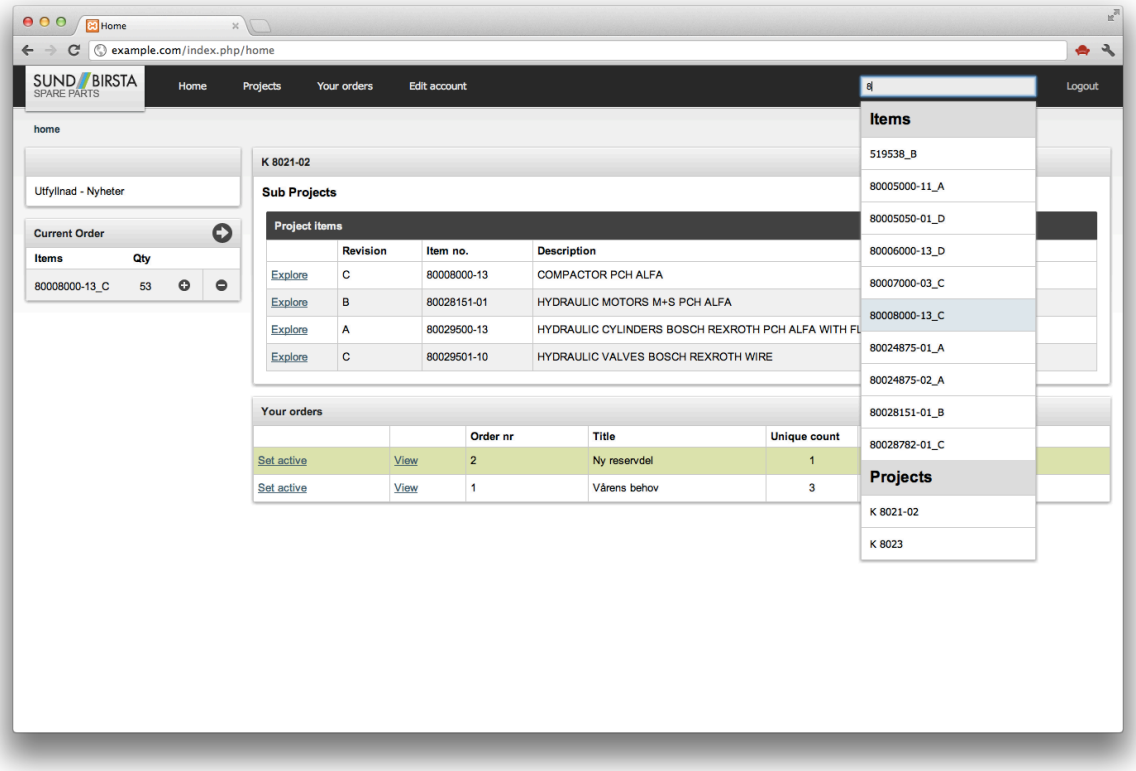
Den teknik som togs fram för att skapa bread crumb-navigering använder webbläsarens adressfält för att beskriva platsen kunden befinner sig på. För varje steg en kund tar, i till exempel en projektstruktur adderas platsen i det befintliga adressfältet. På så vis blir URL:en tydlig och kunden kan se vart länken leder. I koden itererar en funktion genom adressfältet och skriver ut varje segment av URL:en som en separat länk i en lista. Med hjälp av CSS, placerades pilar mellan länkarna, för att visa åt vilket håll strukturen går.



Bread crumb-navigering.

Sökfunktion

Den resulterande sökfunktionen består av en rekursiv funktion som letar efter sin förälder i strukturen. För varje rekursivt steg funktionen tar, läggs föregående resultat till i en sträng tillsammans med sökvägen. När funktionen inte hittar fler artiklar ovanför börjar den leta efter ett projekt som fungerar som topp i strukturen och som slut på sökningen. Vid klick och påbörjad textinmatning i sökrutan påbörjas ett ajaxanrop till sökfunktionen för varje nytt inmatat tecken.



Den implementerade sökfunktionen.

Sökfunktionen genererar en lista med matchande artiklar och projekt som returneras till vyn. Genom att använda upp- och nertangenterna på tangentbordet får användaren snabb access till resultaten. När användaren traverserar ner i listan till den eftersökta artikeln används enter-knappen för att välja artikeln och bli vidarebefordrad dit.

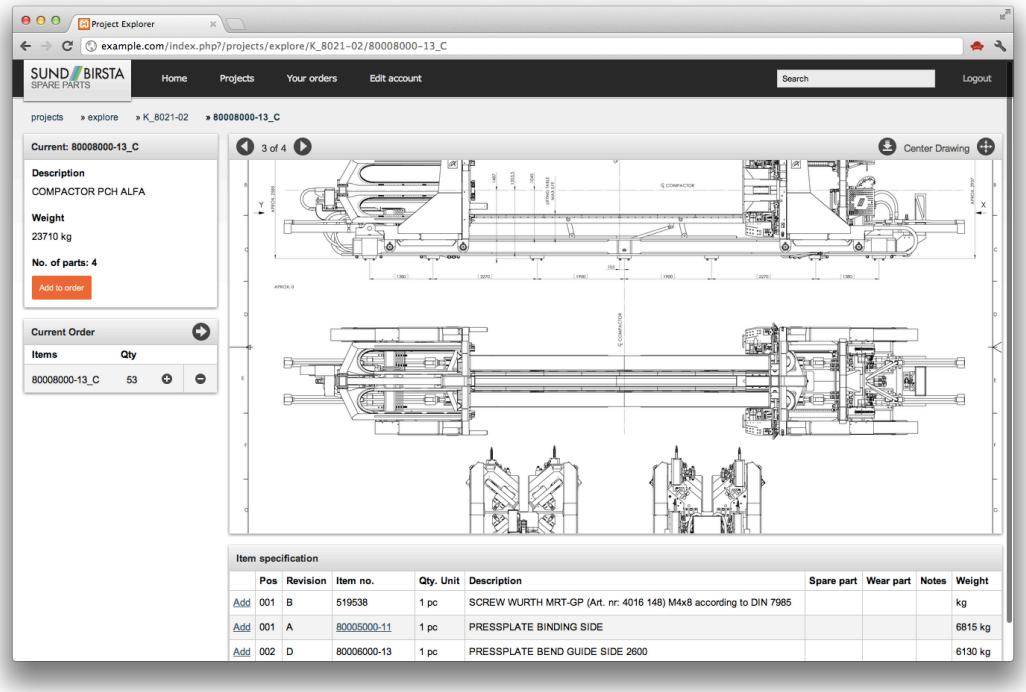
Funktionsknappar

I systemet finns funktionsknappar som utför olika funktioner istället för att navigera till en ny sida. Knapparna är visualiserade som svarta cirklar som dämpats i opacitet med hjälp av CSS. När en användare lägger musen över en länk dämpas opaciteten ännu mer för att visa att den är klickbar.

Funktionsknappar

5.2.2 Färgschema

Färgschemat som togs fram består av en neutral gråskala med blåa och gröna inslag. En orangeröd kulör bestående av mestadels rött och en aning gult används som accentfärg. Systemets olika moduler består av vita boxar med ljusgrå topplister. Runt boxarna finns en svagt grå skugga för att visa kanterna.



Kompletta projektutforskaren

I toppmenyn används en mörkgrå nyans för att ge en professionell känsla. Grå färgscheman används på företagshemsidor för att ge en formell, professionell och sofistikerad känsla. I listor lades en svagt gråblå kulör till på varannan rad som skapar en bättre överblick över de långa raderna. Färgen blå valdes då den associeras med företag och ger en seriös känsla så länge den inte blir för ljus i kombination med hög färgmättnad.

Systemet innehåller tre olika nivåer av meddelanden. En grön kulör används för positiva meddelanden, till exempel när en funktion returnerar önskat resultat. För meddelanden av mer neutral typ och mindre felmeddelanden används gult. Vid meddelanden som beskriver större fel används en röd färg. De tre färgerna grönt, gult och rött ger en tydlig och distinkt skillnad mellan de olika nivåerna av meddelanden. Färgerna har sänkt färgmättnad och en del grått inblandat.

Färdiga färgschemat

5.3 Databas

Databasen utvecklades med hjälp av ER-diagram och implementerades sedan i Microsoft SQL. För att flytta databasrelaterade funktioner från back-end-delen av systemet har triggers använts. Nedan följer en beskrivning av databasens struktur och närliggande funktionalitet.

5.3.1 Struktur

ER-diagrammet som finns i bilaga C utgår från en entitet med namnet *Item* som ska föreställa en artikel. En artikel kan bestå av en eller flera andra artiklar som i sin tur består av andra artiklar. I ER-diagrammet finns det en relation *Items_Items* som beskriver artiklar och deras komponenter. För att kunna beskriva hur olika artiklar förhåller sig till de olika projekten som en kund äger har de lagts till en relation *Project_Items* som definierar sambandet mellan artiklar och projekt. Då Sund Birstas sätt att organisera en kunds projekt, där ett projekt kan innehålla flera olika projekt finns det en relation *Projects_Projects* som definierar sambandet mellan olika projekt.

Diagrammet beskriver hur användare lagras och hur de är kopplade till olika projekt. Det finns tre olika nivåer av användare (*admin*, *contact*, *customer*), i form av entiteter med tillhörande relationer som beskriver hur de ska förhålla sig till varandra. Eftersom det bara är *customers* som kan kopplas till olika projekt, finns en relation *Projects_Customers* för att kunna koppla ihop olika projekt med olika *customers*.

En central funktion som beskrivs i diagrammet är hur ordrar hanteras. Relationen *OrderContains* beskriver vilka artiklar som ingår i en order. *OrderReceivers* innehåller personer som ska få email information om den specifika ordern.

Nyheter lagras i entiteten *News* och relationen *Upgrade_News* beskriver vilka uppgradering som associeras med en viss nyhet. För att ha möjligheten att lagra vilka artiklar som är en uppgradering för en specifik maskin lagras dessa separat i *Upgrades* för att inte blanda ihop dessa med vanliga artiklar. Kopplingen för att kunna se vilken uppgradering som hör till vilken maskin finns relationen *UpgradeFitsToItem*.

5.3.2 Triggers

Triggers konstruerades att temporärt lägga till den uppdaterade raden som en ny rad i modertabellen. Därefter går det att ändra i alla andra påverkade tabeller och i samband med detta, referera till den nya raden i modertabellen istället. När alla tabeller blivit uppdaterade tas den gamla raden i modertabellen bort.

De triggers som hanterar borttagningar fungerar efter samma princip. Den stora skillnaden är att istället för att ändra i modertabellen först, tas den påverkade raden bort i alla relaterade tabeller. Efter att den aktuella raden blivit borttagen ur alla tabeller går det att ta bort den ur modertabellen då det inte finns några referenser till den längre.

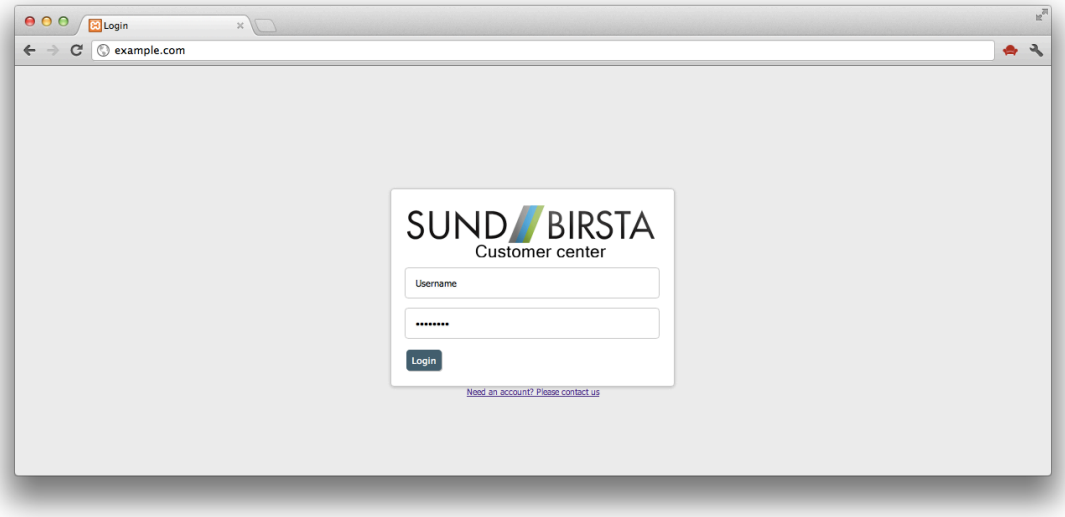
5.4 Funktionalitet

Implementeringen av prototypen har resulterat i olika funktioner. Prototypen är skyddad av ett inloggningssystem med olika användarnivåer. Prototypen innehåller en projektutforskare för att ge kunden åtkomst till alla artiklar som ingår i köpta maskiner. Systemets orderhanteringssystem ger kunder möjlighet att skicka efter artiklar via en enhetlig kanal.

5.4.1 Inloggning och användarnivåer

I prototypen finns ett implementerat inloggningssystem med olika användarnivåer. Om en användare försöker hämta en sida utan att vara inloggad blir användaren automatiskt omdirigerad till inloggningssidan. Varje sida i systemet har en fördefinierad användarnivå som krävs för att den ska visas. Vid försök att komma åt en sida utan korrekta rättigheter blir användaren direkt omdirigerad till startsidan och ett varningsmeddelande visas. Systemet har tre användarnivåer; "customer", "contact", "admin".

Customer: Kunder till Sund Birsta, användarna av systemet.
Contact: Personal på Sund Birsta. Kan se kundernas ordrar och bekräfta dessa. Kan skicka meddelanden och nyheter till användare.
Admin: Administrerar systemet. Kan ändra uppgifter för övriga användargrupper.



Inloggningssidan till systemet

5.4.2 Orderhantering

Systemet kan hantera ordrar som användare skapar. När en order är aktiv går det att lägga till artiklar och information om den. Ordrar kan sparas och användas vid flera tillfällen. Den order som är aktiv visas i ordervyn men även som en sidomeny i projektutforskaren. I projektutforskaren går det att lägga till nya artiklar och ändra befintliga artiklars kvantitet.

5.4.3 Excelnedladdning

När en order är färdig för att skickas bifogas en Excel-fil som innehåller artiklarna, aktuell kundinfo och ordernummer. Varje del av artikelinformationen ligger i separata kolumner. Excel-filen gör det enkelt för Sund Birsta att importera ordern till deras säljsystem.

5.4.4 Visning av ritningar

Ritningar i PNG-format för enkel visning i webbläsare ligger lagrade i en mappstruktur med olika maskinnamn. När projektutforskaren ska visa upp en maskin letar en funktion efter en mapp med samma namn som maskinen. Resulterar letandet i en mapp laddas mappens innehåll in som länkar till HTML-dokumentet som skapas. På klientsidan används javascript för att möjliggöra bläddring mellan ritningarna utan att sidan laddas om. Ritningarna skalas för att passa in i webbläsarens fönster men zoomas in vid musklick.

5.4.5 Språkstöd

Systemet är uppbyggt med hjälp av CodeIgniters språkbibliotek. Statiska texter är placerade i språkfiler tillsammans med ett tillhörande nyckelord. I koden används språkfunktioner för att få fram korrekt text med hjälp av nyckelorden. Varje språk ligger i en separat mapp för enkel åtkomst.

6 Diskussion

Under projektets gång har många val gjorts. Följderna av dessa har varit bra och andra mindre bra. I detta kapitel förs en diskussion kring de metoder som använts i projektet.

6.1 Planering

Vid uppstarten av projektet saknades en klar förståelse för projektets omfattning samt de problem som skulle mötas. Därför var det nödvändigt att göra en förstudie av systemet för att få en bättre förståelse av systemets omfattning och problem så att rätt strategiska beslut skulle kunna tas.

Under förstudien saknades kontakt med Sund Birstas kunder (de som faktiskt ska vara användare av systemet). Detta gjorde att största fokus hamnade på hur systemet skulle vara uppbyggt för att passa Sund Birsta. Då meningen med systemet är att locka fler kunder till Sund Birstas eftermarknad anses att fokus borde ligga på kundens upplevelse så att Sund Birstas system föredras över andra konkurrenters.

Då det troligen hade varit svårt att ha direktkontakt med några av Sund Birstas kunder (största kundkretsen finns utomlands), hade i alla fall en undersökning med slumpvis valda personer kunnat göras. Då hade troligen mer indata samlats kring systemets användarvänlighet.

För att få en bättre förståelse för implementationens storlek samt problem hade en person med tidigare erfarenhet kunnat intervjuats. Som det blev nu underskattades tiden som krävdes för att implementera funktionalitet.

Förstudien gav en bra förståelse även om det saknades kontakt med riktiga kunder. Studien visade systemets problem samt vad som krävdes för att lösa dem. En anledning till att förstudien blev lyckad var att Sund Birsta var hjälpsamma i uppstartsprocessen samt den goda kontakten med dem.

6.1.1 Planeringsmetoder

Tidigt i projektet användes brainstorming för att ta fram de användarkrav som skulle kunna ställas på systemet. Resultatet av detta blev en lång lista med användarkrav som ställdes på systemet samt att mycket funktionalitet till systemet togs fram. Detta ansågs som väldigt lyckat då målet med brainstorming, som skrivet ovan, är just att få fram många idéer och tankar runt ett problem.

Även om mycket fler goda idéer producerades än vad som hann implementeras hade det varit intressant att använda sig av ett annat brainstormingsupplägg. Det kanske, som skrivet ovan, hade producerats fler idéer om personliga brainstormingssessioner hade använts. Ett annat upplägg som hade kunnat användas är att alla har brainstormingssessioner själva först, därefter träffas alla för en gemensam brainstormingssession där det kan skapas gemensamma idéer samt spinnas vidare på de idéer som uppstod under de enskilda brainstormingssessionerna. På detta

sätt anses att det bästa från enskild brainstorming och gemensam brainstorming hade kunnat kombineras.

Dock ansågs det inte som viktigt att tömma ut alla användarkrav som skulle kunna ställas på systemet, detta eftersom flera av de extra användarkraven som hade kommit fram säkerligen hade ändrats allteftersom nya versioner av prototypen utvecklades. Dessutom skulle det enligt den evolutionära modellen diskuteras om nya användarkrav vid varje ny version av prototypen av systemet. Det ansågs lättare att hitta nya användarkrav vid dessa tillfällen.

6.2 Utveckling

Under utvecklingen av systemet användes ramverket CodeIgniter. Det var ett passande ramverk på grund av dess låga inlärningskurva och goda funktionalitet. Ramverket ökade effektiviteten vid programmeringen då färdiga standardfunktioner kunde användas för repetitiva uppgifter.

6.2.1 Utvecklingsmetoder

Att dela upp arbetet i olika faser var under arbetets gång mycket hjälpsamt. Att ha bestämda datum med deadlines för de olika faserna gav oss en god översikt på tidsschemat. Detta gjorde att tiden som gick åt till att göra mock-ups inte blev så stor att implementering inte hanns med.

De olika versionerna av systemet byggdes upp med hjälp av extremprogrammering som utvecklingsmetod. Att dela upp varje version i flera små delproblem under en planeringsfas har varit ett måste. Det gav en tydlig översikt på vad som faktiskt behövde göras för att bli klar med varje version samt att det gav en tydlig översikt på vilken funktionalitet som var viktigast. Översikten och prioriteringen av funktioner visade sig vara viktig då implementeringen drog ut på tiden och viss funktionalitet fick prioriteras bort.

Här märktes det också att gruppen saknade erfarenhet från att programmera projekt av denna typ. Funktionalitet skulle implementeras i delmål om fyra till 16 timmar men tiden för att implementera funktionalitet underskattades dock ständigt.

Om gruppen skulle implementera ett nytt liknande system hade nog metoden sett annorlunda ut. Funktionaliteten hade brutits ner till mindre delmål tillsammans med längre iterationer och färre möten per vecka. Dock hade exakta deadlines behållits från extremprogrammeringen i och med arbetets natur med ett exakt slutdatum.

Extremprogrammering förespråkar parprogrammering. Det första som märktes när denna metod användes var att effektiviteten blev högre. I och med att erfarenheten inom gruppen är ganska låg inom området var det dessutom ganska bra att ha någon nära till hands när problem uppstod. De flesta problemen kunde lösas på ganska kort tid när två personer samarbetade.

6.2.2 Databas

Det bestämdes i ett tidigt skede att MSSQL skulle användas som den slutgiltiga databasen även om det skulle medföra mer arbete. Parallellt med MSSQL skulle MySQL användas eftersom det gick snabbare att implementera och det var fler i gruppen som kunde MySQL.

Det kanske skulle varit bättre att helt fokusera på MSSQL istället för att arbeta med MySQL parallellt. Dels på grund av att det tog mer tid än beräknat att implementera databasen med MSSQL och att det troligtvis hade gått snabbare att få en färdig struktur om allt fokus hade legat på MSSQL. Att det hade gått snabbare hade hjälpt utvecklingen av portalen då en del onödiga iterationer rörande databasen hade kunnat undvikas.

Ett annat alternativ hade varit att välja enbart MySQL från början. Det hade lett till att tiden som lades på databasen hade minskat drastiskt på grund av att problemet med ON UPDATE/DELETE CASCADE inte hade funnits. Även problemet med att få PHP och CodeIgniter att kommunicera med databasen hade försvunnit. MySQL var inte motiverbart då det inte är gratis för företag så det var aldrig ett alternativ.

7 Rekommendationer

Den fortsatta utvecklingen av systemet kan bestå i att utöka funktionaliteten i systemet. Den nuvarande prototypen har fokus på användarens interaktion med portalen men saknar administrations- och kontaktfunktionalitet. En del av de krav som finns i kravspecifikationen har inte blivit implementerade, att göra klart dessa får ses som en naturlig fortsättning på systemet. Slutligen skulle det exportverkyg som behövs mellan Sund Birstas nuvarande system och prototypen behöva skapas för att prototypen ska kunna bli komplett.

8 Slutsats

Sund Birsta har ställt ett antal krav och söker en plattform för att hantera sin eftermarknad. De krav som varit uppställda har tagits hänsyn till och har till stor del blivit uppfyllda. Prototypen bedöms vara en bra representation av projektets mål och ger en utförlig bild av systemet Sund Birsta söker. Det utvecklade systemet har ett branschenligt professionellt utseende och uppfattas som användarvänligt.

Det finns många sätt att presentera olika stadier i ett projekt. Det visade sig att Mock-ups var ett bra hjälpmedel för att visuellt representera utvecklingen av prototypen. De gav möjligheten att visa upp systemet för personal på Sund Birsta som är mindre insatta i mjukvaruutveckling och i det här projektet.

Hela prototypen är omgärdat att ett inloggningssystem som skyddar systemets funktioner. Varje användare har sin egen inloggning som ger tillgång till en anpassad startsida innehållande inköpta maskiner och skapade ordrar. Inloggningssystemet ger ett effektivt skydd mot obehöriga användare och ger endast personer med åtkomst tillgång till portalen.

Ett av målen med projektet var att göra det möjligt att fortsätta utveckla systemet. Det målet anses vara uppfyllt då hela systemet följer vedertagna konventioner och är uppbyggt på standarder och öppen källkod. Prototypen kan fortsätta utvecklas av externa utvecklingsteam om Sund Birsta väljer att gå vidare med utvecklingen.

Källhänvisningar

Arisholm, E. Gallis, H. Dybå, T. Sjøberg, D. I. K. (2007) Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. *IEEE Transactions on Software Engineering*, vol. 33, nr. 2, ss. 65-86.

Ballakur, A. Steudel, H. J. (1984) Integration of job shop control systems: A state-of-the-art review. *Journal of Manufactoring Systems*, vol. 3, nr. 1, ss. 71-79.

Bosch, J. (2000) *Design and Use of Software Architectures*. Harlow, UK: Addison-Wesley.

Chapman, C. (2010). Color Theory for Designers, Part 1: The Meaning of Color. Smashing Magazine. 2010-01-28. http://www.smashingmagazine.com/2010/01/28/color-theory-for-designers-part-1-the-meaning-of-color/ (2012-05-02)

Clark, C. H. (1958) *Brainstorming - The Dynamic New Way To Create Successful Ideas*. California, US: MeJvin Powers Wilshire Book Company.

Cockburn, A. Williams, L. (2001) *The Costs and Benefits of Pair Programming*. Salt Lake City, US.

EllisLab (2012) CodeIgniter user guide.
http://codeigniter.com/user_guide/ (2012-05-03)

Glass, R. L. (2006) *Software Conflict 2.0.* Atlanta, US: developer. * Books.

Jackson, J. (2011) W3C: HTML5 will be finished in 2014. Computerworld. 2011-02-14
http://www.computerworld.com/s/article/9209322/W3C_HTML5_will_be_finished_in_2014 (2012-05-08).

Lerdorf, R. Tatroe, K. MacIntyre, P. (2006) *Programming PHP*. 2nd Edition. California, US: O'Reilly Media, Inc.

Microsoft (2012) Cascading Referential Integrity Constraints. http://msdn.microsoft.com/en-us/library/aa933119(v=sql.80).aspx (7 Maj. 2012).

Morales-Chaparro, R., Linaje, M., Preciado, J. C., Sánchez-Figueroa, F. (2007) *MVC Web design patterns and Rich Internet Applications*. Cáceres, Spain: QUERCUS Software Engineering Group.

Oracle (2012) Download MySQL Community Server.
http://www.mysql.com/downloads/mysql/ (7 Maj. 2012).

Parrish, A. Smith, R. Hale, D. Hale, J. (2004) A Field Study of Developer Pairs: Productivity Impacts and Implications. *IEEE Software*, vol. 21, nr. 5, ss. 76-79.

Paulus, B. Dzindolet, M. T. Poletes, G. Camacho, M. L. (1998) Perception of Performace in Group Brainstorming: The Illusion of Group Productivity. *Personality and Social Psychology Bulletin*, vol. 19, nr. 1, ss. 78-89.

Petrijevcanin Vuksanovic, I. Sudarevic, B. (2011) Use of Web Application Frameworks in the Development of Small Applications. *MIPRO 2011*; 23-27 may, 2011, Opatija, Croatia. s. 458-462.

PHP - MSSQL requirements (2012). http://www.php.net/manual/en/mssql.requirements.php (7 Maj. 2012).

Purer, K. (2009) PHP vs. *Python vs. Ruby - The web scripting language shootout*. Vienna, Austria: Vienna University of Technology Institute of Computer Languages Compilers and Languages Group.

Sache, A. (2010) Choosing the Open-Source Framework for Web Application Development. *Open Source Science Journal*, vol. 2, nr. 4, ss. 5-22.

Soegaard, M. (2004). Mock-ups. Interaction-design.org. http://www.interaction-design.org/encyclopedia/mock-ups.html (2012-05-04)

Sommerville, I. (2011) *Software Engineering*. 9th Edition. Massachusetts, US: Pearson Education, Inc.

The Institute of Electrical and Electronics Engineers, Inc. (2000) *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. New York, US: IEEE. (IEEE Std 1471-2000).

Rogers Y., Sharp H. & Preece J. (2011) Interaction design, beyond human-computer interaction. 3rd Edition. Chichester: John Wiley & Sons Ltd.

Tidwell, J. (2005) *Designing Interfaces*. 1st Edition, Sebastopol, Canada: O'Reilly Media, Inc.

TIOBE (2012) TIOBE Programming Community Index for April 2012. *TIOBE Software: Tiobe Index*. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (2012-05-07)

Upton, D. (2007) *CodeIgniter for Rapid PHP Application Development - Improve your PHP coding productivity with the free compact open-source MVC CodeIgniter framework!*. Birmingham, UK: Packt Publishing Ltd.

Wall, L. Christiansen, T. Orwant, J. (2000) *Programming Perl*. 3rd Edition. California, US: O'Reilly Media, Inc.

Williams, L. Kessler, R. R. Cunningham, W. Jeffries, R. (2000) Strengthening the Case for Pair Programming. *IEEE Software*, vol. 17, nr. 4, ss. 19-25.

Wilson, J. M. (2003) Gantt charts: A centenary appreciation. *European Journal of Operational Research*, vol. 149, nr. 2 , ss. 430-437.

World Wide Web Consortium (2012) http://www.w3.org/

Yu, H. Lu, Z. (2008) How to Choose Server-side Scripting Languages & Database Servers in Web Mining. *2008 International Symposium on Knowledge Acquisition and Modeling*, 21-22 dec, 2008, Wuhan, China. ss. 98-102.

Zaring, P (2011) Software Engineering Fundamentals. *Studieportalen*. https://www.student.chalmers.se/hp/hp/?hp_id=8312&hp_view=handout (2012-05-07)

A Systemkravsspecifikation

Användare:

1: Bläddra bland artiklar
En kund kan bläddra mellan artiklar i en lista. Genom att klicka på en artikel visas specifikation för artikeln och alla ingående komponenter (artiklar) visas i en ny lista. Om det finns en ritning för artikeln visas även den.

2: Hämta manualer
En kund ska kunna ladda hem manualer för ett valt projekt.

3: Kontakta kontaktperson
Alla kunder till Sund Birsta har en personlig kontaktperson. Genom systemet ska det vara möjligt att hitta kontaktuppgifter för kundens kontaktperson samt att kontakta personen direkt genom systemet.

4: Se 'nyheter'
På startsidan ska kunder se nyheter som Sund Birsta vill informera kunden om.

5: Ladda ner ritningar
Om en artikel har en ritning ska denna kunna laddas hem i pfd-format.

6: Ta bort varor från kundvagnen
I visningsvyn för kundvagnen ska det vara möjligt att ta bort en vara.

7: Lägga till varor i kundvagnen
I visningsvyn för artiklar ska det vara möjligt att lägga till dessa i kundvagnen.

8: Skicka iväg order
I visningsvyn för orderspecifikationer ska kunden kunna skicka iväg ordrar. När en order skickas iväg kommer den upp som en ny order hos Sund Birstas system. Ett e-mail ska skickas till kontaktpersonen för projektet. En Excel-fil som innehåller ordern ska vara bifogad för att kontaktpersonen lätt ska kunna kopiera cellerna till Sund Birstas befintliga säljsystem.

9: Se orderhistorik (startsida)
På startsidan ska länkar till de tre senaste ordrarna finnas.

10: Se orderhistorik (egen vy)
I den här vyn visas alla gamla ordrar. Genom att klicka på en order kan kunden få mer information om den valda ordern.

11: Logga in
Det första användaren möts av är en inloggningsskärm. För att kunna göra något mer med systemet måste användaren logga in. När användaren har loggat in får denne tillgång till systemet. Beroende på vilken sorts användare som loggar in får användaren tillgång till olika delar av systemet.

12: Ändra lösenord
På en inställningssida ska en användare kunna ställa in ett nytt lösenord. Användaren får först bekräfta sitt nuvarande lösenord i en lösenordsruta och därefter i två efterföljande rutor skriva in önskat lösenord i båda rutorna för att säkerställa att användaren verkligen skrev det lösenord som den tänkte byta till.

13: Ändra språk
En användare skall enkelt kunna byta språk genom att klicka på flaggor som representerar de olika språk som sidan kan visas på. Dessa flaggor skall befinna sig väl synligt för användaren.

20: Ändra personlig information
En användare skall genom en inställningssida, kunna ändra sin personliga information.

21: Söka efter artiklar:
En användare skall kunna söka efter en artikel i en sökruta ifall den har tillgång till artikels namn eller artikelnummer.

Kontaktperson:

11: Logga in
Det första användaren möts av är en inloggningsskärm. För att kunna göra något mer med systemet måste användaren logga in. När användaren har loggat in får den tillgång till systemet. Beroende på vilken sorts användare som loggar in får användaren tillgång till olika delar av systemet.

12: Ändra lösenord
På en inställningssida ska en användare kunna ställa in ett nytt lösenord. Användaren får först bekräfta sitt nuvarande lösenord i en lösenordsruta och därefter i två efterföljande rutor skriva in önskat lösenord i båda rutorna för att säkerställa att användaren verkligen skrev det lösenord som den tänkte byta till.

13: Ändra språk
En användare skall enkelt kunna byta språk genom att klicka på flaggor som representerar de olika språk som sidan kan visas på. Dessa flaggor skall befinna sig väl synligt för användaren.

14: Ändra orderstatus
När en kund skapar en order på hemsidan binds en status till denna order. Statusen ska till exempel kunna vara “order lagd”, “order behandlad” eller “order skickad”. En anställd på Sund Birsta ska kunna ändra statusen på en lagd order.

15: Koppla kontaktperson till kund
Varje kund har en kontaktperson. När en kund läggs till i systemet saknar kunden kontaktperson. En administratör ska kunna knyta en kontaktperson till en kund.

18: Lägga till nyhet
Personal på Sund Birsta ska genom att fylla i ett formulär kunna lägga upp nyheter. Personalen ska kunna välja vilka kunder som ska kunna se nyheten.

19: Se lagda ordrar
Personalen på Sund Birsta ska kunna se när en ny order har skickats. Det ska dessutom finnas möjlighet att få ett e-mail som berättar att en ny order har lagts.

20: Ändra personlig information
En användare ska genom en inställningssida, kunna ändra sin personliga information.

Admin:

11: Logga in
Det första användaren möts av är en inloggningsskärm. För att kunna göra något mer med systemet måste användaren logga in. När användaren har loggat in får denne tillgång till systemet. Beroende på vilken sorts användare som loggar in får användaren tillgång till olika delar av systemet.

12: Ändra lösenord
På en inställningssida ska en användare kunna ställa in ett nytt lösenord. Användaren får först bekräfta sitt nuvarande lösenord i en lösenordsruta och därefter i två efterföljande rutor skriva in önskat lösenord i båda rutorna för att säkerställa att användaren verkligen skrev det lösenord som den tänkte byta till.

13: Ändra språk
En användare skall enkelt kunna byta språk genom att klicka på flaggor som representerar de olika språk som sidan kan visas på. Dessa flaggor skall befinna sig väl synligt för användaren.

14: Ändra orderstatus
När en kund skapar en order på hemsidan binds en status till denna order. Statusen ska till exempel kunna vara “order lagd”, “order behandlad” eller “order skickad”. En anställd på Sund Birsta ska kunna ändra statusen på en lagd order.

15: Koppla kontaktperson till kund
Varje kund har en kontaktperson. När en kund läggs till i systemet saknar kunden kontaktperson. En administratör ska kunna knyta en kontaktperson till kunden.

16: Lägga till kontaktperson
En administratör ska kunna skapa ett nytt konto för en anställd på Sund Birsta.

17: Ändra rättigheter för kontaktpersoner

En administratör kan ändra vilka rättigheter en anställd har för systemet, det vill säga vilka funktioner en anställd kan använda.

18: Lägga till nyhet

Personal på Sund Birsta ska genom att fylla i ett formulär kunna lägga upp nyheter. Personalen ska kunna välja vilka kunder som ska kunna se nyheten.

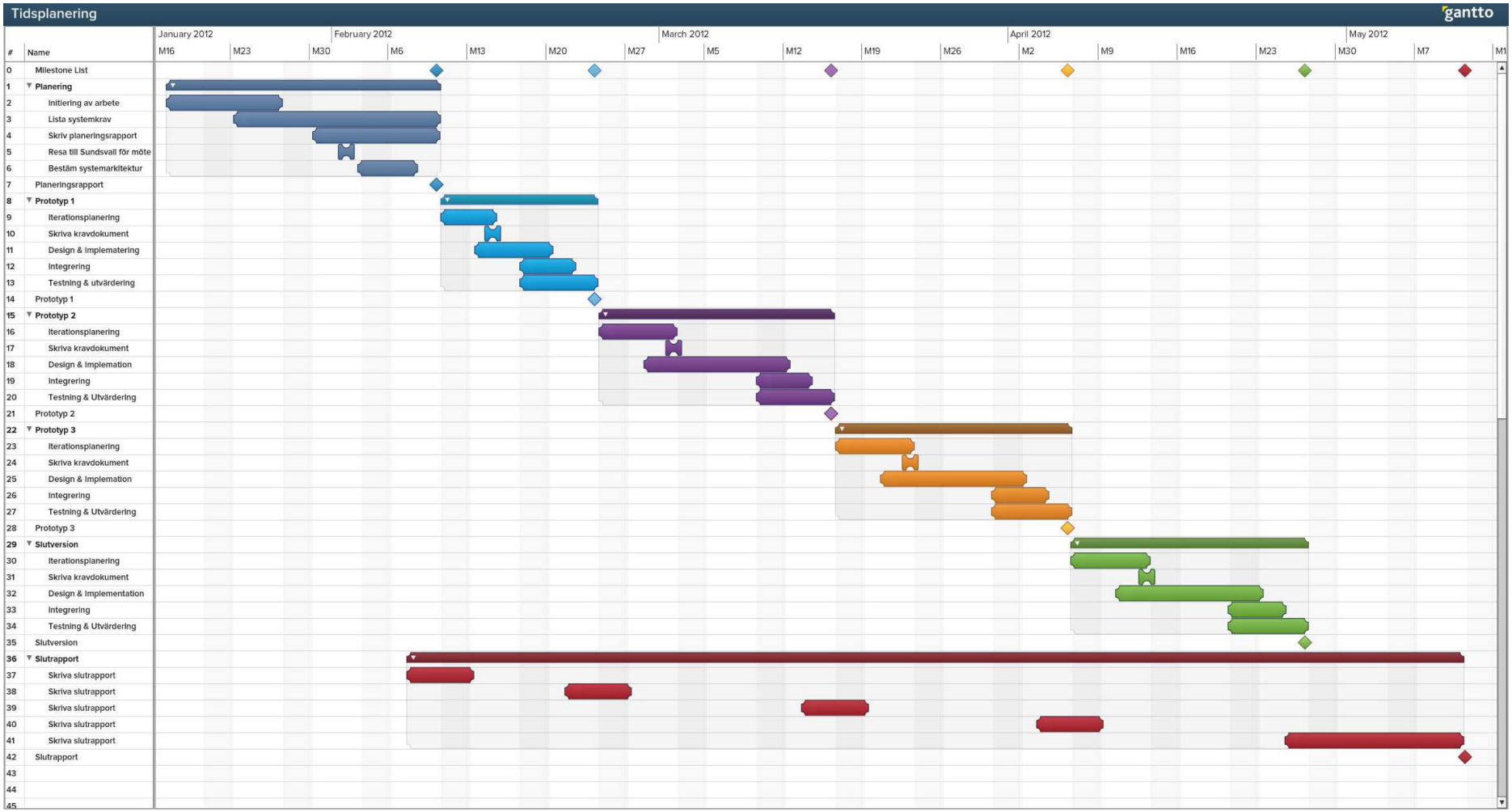
19: Se lagda ordrar

Personalen på Sund Birsta ska kunna se när en ny order har skickats. Det ska dessutom finnas möjlighet att få ett e-mail som berättar att en ny order har lagts.

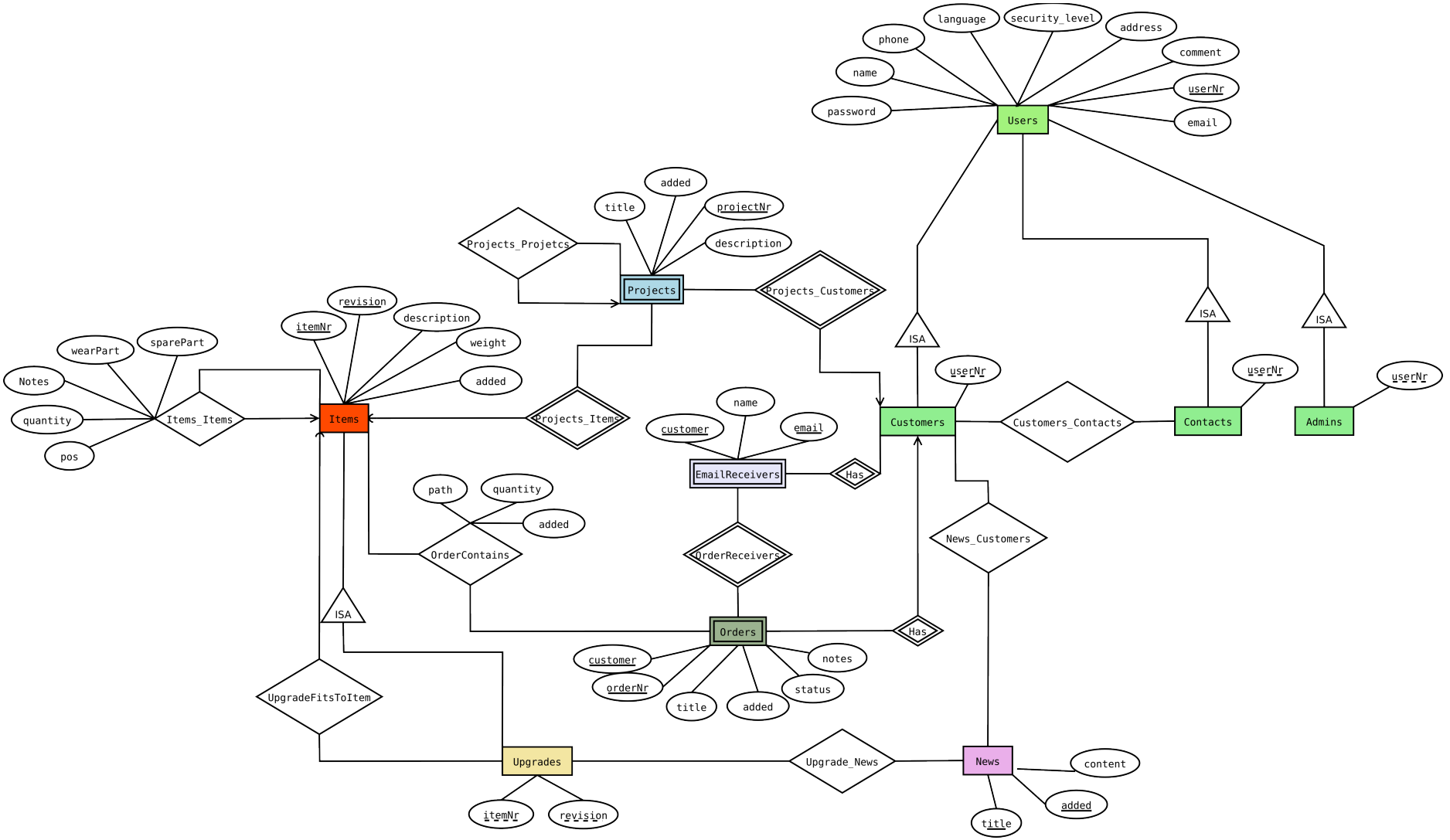
20: Ändra personlig information

En användare ska genom en inställningssida, kunna ändra sin personliga information.

B Gantt-schema



C ER-diagram


language
security_level
address
phone
comment
name
userNr
password
email
Users
added
title
projectNr
description
Projects_Projetcs
Projects
Projects_Customers
revision
itemNr
description
weight
added
wearPart
sparePart
Notes
quantity
pos
Items_Items
Items
Projects_Items
ISA
ISA
ISA
userNr
userNr
userNr
name
customer
email
Customers
Customers_Contacts
Contacts
Admins
EmailReceivers
Has
path
quantity
added
OrderContains
OrderReceivers
News_Customers
ISA
Orders
Has
customer
orderNr
title
added
status
notes
UpgradeFitsToItem
Upgrades
Upgrade_News
News
content
added
title
itemNr
revision