Utveckling av ett generellt bokningssystem - erfarenheter i projekt i programvaruutveckling

PUM grupp 3

13 maj 2014

Version 1.0

Dokumenthistorik

Version	Datum	Beskrivning
0.1	2014-03-10	Första utkastet
0.2	2014-05-09	Andra utkastet
1.0	2014-05-13	Första versionen

Sammanfattning

Rapporten behandlar utvecklandet av ett generellt bokningssystem för att tillåta att användare bokar tider hos en kund via en webbsida. Systemet ska också ha stöd för att automatiskt kunna generera faktureringsunderlag för att underlätta kundens administration av bokningar. För att bättre förstå vilken funktionalitet ett generellt bokningssystem behöver, har Vikbolandets ryttarförening använts som modellkund.

Utvecklingen har lett fram till ett generellt bokningssystem som kan anpassas för att ha samma funktionalitet som det system Vikbolandets ryttarförening tidigare använt sig av. För att undersöka hur generellt resultatet blev diskuteras hur systemet kan användas i andra scenarion.

Innehåll

1 Inledning

Projektets syfte är att ta fram ett webbaserat bokningssystem. Det finns många olika sätt för företag och föreningar att hantera bokningar. Det leder till att de också har olika krav på hur ett bokningssystem ska fungera. Användare ställer därigenom olika krav på vad systemet ska kunna och inte kunna. Därför går projektet huvudsakligen ut på att ta fram ett modulärt ramverk för att snabbt och enkelt kunna ta fram ett skräddarsytt bokningssystem som passar precis för kundens ändamål, snarare än att implementera ett bokningssystem för en specifik kund.

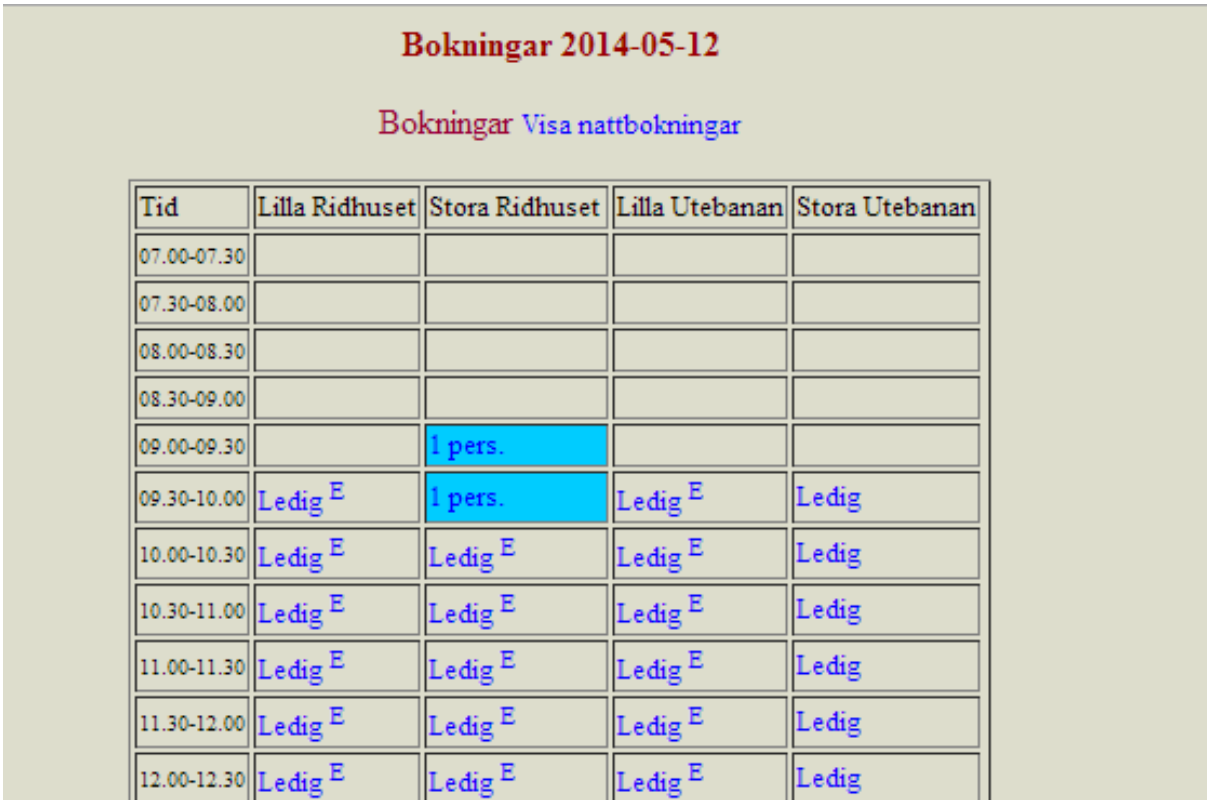
Fördelen med ett skräddarsytt bokningssystem är framför allt att kunden inte behöver ändra sitt sätt att hantera bokningar för att passa in i systemet. I stället byggers ett system som fungerar som kunden vill ha det. Utöver fördelarna för kunden, inger också ett skräddarsytt system en känsla av professionalism gentemot bokningssystemets slutanvändare. Den största nackdelen med skräddarsydda system är att de måste utvecklas specifikt för en kund. Därför är det extremt fördelaktigt att ha en generell bas att stå på, så att ett anpassat bokningssystem kan tillhandahållas snabbt och billigt.

Projektet utförs som ett kandidatarbete i kursen TDDD77, *Kandidatprojekt i programvaruutveckling* (1). Gruppen består av:

- Nils Axelsson, designansvarig för mobil
- Gustaf Brunberg, dokumentansvarig
- Andreas Larsson, designansvarig för desktop
- Claes Lööf, testansvarig
- Behnam Sani, arkitekt
- Peter Stockman, utvecklingsledare
- Filip Strömbäck, teamleader
- Per Wennberg, analysansvarig

1.1 Förutsättningar

Utgångspunkten för projektet var det gamla bokningssystemet som innan projektet användes på Vikbolandets ryttarförening, en ridskola utanför Norrköping. Det gamla systemet är ostrukturerat och underhålls inte längre. Därför önskade kunden beställa ett system som efter mindre anpassningar skulle kunna ersätta det gamla bokningssystemet. Projektgruppen har haft tillgång till det gamla systemet för att kunna undersöka dess funktionalitet och dess brister. Projektgruppen har även haft kontakt med Vikbolandets ryttarförening för att undersöka hur det gamla systemet använts, samt vilken ytterligare funktionalitet som önskas från det nya systemet.



Figur 1: Bild av det gamla bokningssystemet.

Projektgruppen var tvungen att ta hänsyn till att varje person endast hade 250 timmar att spendera på planering, dokument, testning och kodning. Tidsbegränsningen var avgörande för vilken prioritet somliga krav skulle få och hur mycket tid som skulle läggas på dokumentskrivning.

1.2 Definitioner

- Det gamla systemet: det system som Vikbolandets ryttarförening använder för att hantera bokningar för närvarande.
- Lokal: En lokal är här något som en användare kan boka. Behöver inte motsvara en fysisk lokal.
- Aktivitet: Definierar något en användare bokar. En aktivitet innehåller hur mycket plats aktiviteten i lokalen tar. Därmed kan systemet inse när en lokal är full.
- Platser: Anger hur stor kapacitet en lokal har i en godtycklig enhet. Det definieras i och med antalet platser olika aktiviteter tar upp.
- Trac: ett system för att hantera dokumentation och ärenden på en webbsida. Det system som gruppen har använt för att hantera projektet.

1.3 Systemkrav

De huvudsakliga kraven på systemet utgår från att kunden önskar ett generellt bokningssystem som ska kunna ersätta det nuvarande bokningssystemet som används på Vikbolandets ryttarförening. Eftersom systemet är generellt är det acceptabelt att det system som levereras behöver anpassas något innan det

kan ersätta det befintliga systemet. Dessa anpassningar ska dock vara snabba och enkla att göra för en erfaren webbutvecklare.

För att kunna ersätta Vikbolandets ryttarförenings nuvarande system, ska det levererade systemet kunna låta registrerade användare boka plats i en av ett förutbestämt antal lokaler. Både lokalernas storlek och vilka tider de är bokningsbara ska på förhand kunna bestämmas av en systemadministratör. Det ska också vara möjligt för en användare att boka alla platser i en lokal, eventuellt för en extra kostnad.

Utöver att hantera framtida bokningar ska systemet även tillhandahålla historik över de bokningar som lagts tillsammans med prisinformation, så att lokalernas ägare enkelt kan fråga systemet hur mycket en viss användare ska betala för att ha utnyttjat lokalerna under en viss tidsperiod.

Systemet behöver därför ha information om vad olika bokningar kostar, vilket innebär att systemet även behöver känna till de typer av rabattkort som finns tillgängliga och vilka användare som har vilka kort.

Kunden önskar också att systemet ska gå att köra på en server med bara öppen mjukvara, exempelvis Linux, Apache, MySQL och PHP.

1.4 Avgränsningar

För att gruppen inte ska behöva se till att systemet fungerar på alla webbläsare har följande avgränsningar gjorts. Systemet ska fungera på:

- Internet Explorer 9 och framåt.
- Mozilla Firefox 26 och framåt.
- Google Chrome 32 och framåt.

1.5 Frågeställning

Frågeställningen som gruppen gemensamt valt att besvara är:

- Kan vi skapa ett generellt bokningssystem som efter anpassningar kan användas i stället för det system som finns hos vår referenskund, Vikbolandets Ryttarförening?

Med detta menas också att systemet för slutanvändaren ser ut som ett system som har byggts speciellt för kunden, och inte på grund av generaliteten blir överdrivet krångligt.

1.6 Disposition

I det följande kapitlet, *Metod*, förklaras gruppens olika metoder. Först förklaras hur gruppen använt och utnyttjat sin utvecklingsmetod för att se till att projektet fortgår. Efter det förklaras hur gruppens forskningsmetod hjälpt till för att samla in data till den gemensamma frågeställningen.

De två efterföljande kapitlen beskriver två vitala delar i projektet. Det första kapitlet är *Systembeskrivning*, detta kapitel kommer förklara lite mer genomgående om hur systemet är uppbyggt i olika komponenter samt hur dessa fungerar. Det andra kapitlet är *Gruppens tekniska erfarenheter*, i detta kapitel förklaras de verktyg som gruppen använt för att nå önskat resultat.

I det femte kapitlet, *Gruppens processrelaterade erfarenheter*, kommer det förklaras mer över hur kommunikationen, dokumentationen och dylikt fungerat i gruppen.

I det sjätte kapitlet, *Individuella bidrag*, finns små rapporter av varje enskild medlem i gruppen där de svarar på sin frågeställning och förklarar hur de kommit fram till resultatet.

I de tre sista kapitlen, *Resultat*, *Diskussion* och *Slutsats*; analyseras gruppens resultat och diskuteras utifrån vad som kunde blivit bättre respektive sämre vid andra metod val.

2 Metod

Nedan beskrivs gruppens metod, uppdelad i utvecklingsmetod, dokumenthanteringsmetod, testmetod och forskningsmetod.

2.1 Utvecklingsmetod

För att utveckla systemet har projektgruppen valt en agil utvecklingsmodell. Under den givna tidsperioden har tre iterationer och en förstudie planerats in. Efter varje iteration, och även mitt i den andra och i den tredje iterationen, har projektgruppen visat upp projektets status för kunden.

Den iterativa modell som valts baseras huvudsakligen på Scrum. Intervallen mellan möten och mötens längd har dock anpassats utifrån att projektet inte har utvecklas på heltid. I stället för dagliga Scrummöten har projektgruppen valt att formellt sammanträda en gång i veckan för en gemensam avstämning av projektets status. Utöver det har webbplatformen Trac använts för att administrera projektet.

Eftersom projektgruppen inte har tillgång till en fast lokal där post-it-lappar kan sättas upp, hanteras ärenden med hjälp av Trac i stället. En fundamental skillnad från Scrum är att varje medlem i utvecklingsteamet är specialist inom ett område och av den anledningen är alla i någon mening produktägare.

Gruppen insåg tidigt värdet av att sitta i samma rum under utvecklingen. Gruppen kallade detta för *kodstugor*. Kodstugorna underlättade kommunikationen mellan gruppmedlemmarna. Det inrättades snabbt ett system där en person i gruppen bokade ledig tid mellan klockan tio och fem, då alla gruppmedlemmar kunde komma och utveckla systemet tillsammans. Dessa möten var inte obligatoriska för någon, men eftersom alla gruppmedlemmar insåg värdet av att snabbt och enkelt kunna fråga alla andra i gruppen om systemets status etablerades att alla gruppmedlemmar satt och jobbade större delen va dagen på sådana här möten.

Kodstugorna gjorde att värdet av formella dagliga Scrum-möten reducerades till noll. Möten blev i stället av genom att relevanta frågor i stället kunde ställas under de tillfällen då de flesta i gruppen ändå var närvarande. De möten som gruppen hade veckovis gick allt mer åt att mer formellt gå igenom systemet och se vad som var kvar att göra. Detta gjordes i praktiken genom att listan över ärenden i Trac gicks igenom, för att alla skulle få en bättre bild av vad som var kvar att göra, samt att få ett tillfälle att stänga eventuella kvarglömda ärenden. Denna genomgång var även bra för att gruppen fick tillfälle att tillsammans bilda en uppfattning om vad som var viktigast att implementera, samt vem som var lämpligast att göra vad utifrån en helhetsbild.

I samband med slutet av en iteration hade gruppen också större möten för planering av nästkommande iteration, samt avstämning med kunden. Alla dessa möten bestod av att gruppen gemensamt gick igenom alla krav i kravspecifikationen och prioriterade kvarvarande krav inför kommande iteration. Vid denna punkt i planeringen presenterade gruppen en preliminär plan för kunden, så att kunden dels fick information om vad som kunde förväntas under kommande iteration, dels så att kunden kunde välja att prioritera om innan något onödigt arbete hade lagts ned.

Under projektets gång har projektgruppen demonstrerat systemet regelbundet för kunden. Demonstrationerna har bestått av att gruppens analysansvarige och en till gruppmedlem har visat systemet för kunden. Demonstrationerna har fungerat som en försäkran att gruppen hade tolkat kundens krav korrekt, och inte missat några vitala detaljer som för kunden hade varit självklara. Under de sista iterationerna minskade intervallet mellan kunddemonstrationerna något, främst eftersom det då var många små, men specifika funktioner som implementerades i systemet.

2.2 Dokumenthanteringsmetod

Den metod som har använts för dokumenthantering har ganska snabbt vuxit fram ur gruppens tidigare erfarenheter från liknande projekt. Metoden har huvudsakligen fyra steg, vilka är som följer:

1. Gruppen samlas för att enas om vad dokumentet ska innehålla. Detta är huvudsakligen en övergripande diskussion som inte tar upp mer än de nödvändiga detaljerna i rapporten.
2. En ansvarig för dokumentet utses och denne får till uppgift att ta fram en disposition över dokumentet samt att utse ansvariga för respektive del i dokumentet.
3. De utsedda personerna ser till att deras delar av dokumentet blir skrivna. Detta antingen genom att skriva dem själva, eller genom att delegera arbetet till någon de anser bättre lämpad för uppgiften.
4. Gruppen samlas för en genomgång av dokumentet. Detta innebär att gruppen korrekturläser dokumentet för att hitta eventuella fel som uppkommit i och med att olika delar av dokumentet har skrivits parallellt. Ansvarig för dokumentet antecknar och korrigerar dessa fel.

Denna metod har använts för att författa och uppdatera dokument såsom kravspecifikationen och projektplanen. För just kravspecifikationen var den första steget i listan mer betonat än för projektplanen, eftersom gruppen ansåg att det var viktigt att alla skulle få en möjlighet att vara med i diskussionen om vilka krav som var viktiga. All brödtext i dokumentet författades dock under punkt 3 ovan.

Metoden kan liknas vid iterativa utvecklingsmetoder för mjukvara, som exempelvis Scrum. Då representerar hela processen ungefär en iteration i den iterativa utvecklingsmodellen. Punkt 1 kan då väl liknas vid planeringsmötet i början av varje iteration (eller sprint, som det heter i Scrum). Punkt 4 kan liknande liknas vid det avslutande mötet i varje iteration.

2.3 Testmetod

För att testa systemets helhet och funktioner delades testningen in i tre olika delar; enhetstestning, systemtestning och acceptanstestning.

- Enhetstestning validerar att en specifik funktion fungerar och beter sig som förväntat utifrån kravspecifikationen. Black-box-testing är metoden som användes för enhetstesterna.
- Systemtestning går ut på att kontrollera att systemet uppfyller de krav som finns specificerade i kravspecifikationen och att systemet är användarvänligt. Användarscenarion och felhanteringstest exekveras mot systemet för att testa dess helhet.
- Acceptanstestning ser till att systemet har utvecklats enligt kravspecifikationen och att systemet är redo för användning hos kunden.

Det var tänkt att systemet skulle integrationstestas, men enhetstesterna täckte tillräckligt mycket av gränssnitten att integrationstestningen inte behövdes.

2.4 Forskningsmetod

Under projektets gång har gruppmedlemmarna samlat in individuella erfarenheter inom det ämne som varje medlem specialiserade sig på. Elektroniska källor har samlats in som referenser till hur olika tekniska delar av PHP-ramverket *Yii* eller delar av PHP eller CSS fungerar. Tryckta källor till designdelen av projektet, samt personliga erfarenheter och viss kundfeedback via ett användbarhetstest, samlades även tidigt in från systerkursen TDDD60, i *Interaktiva system* (2), där bland annat en uppgift var att användbarhetstesta en grafisk prototyp av systemet. Fokus på vilka erfarenheter som gruppmedlemmar har samlat in har varit på vad som skulle kunna göra projektet lättare för en grupp nästa år, eller vad som gruppmedlemmen själv hade velat veta innan projektet börjar.

3 Systembeskrivning

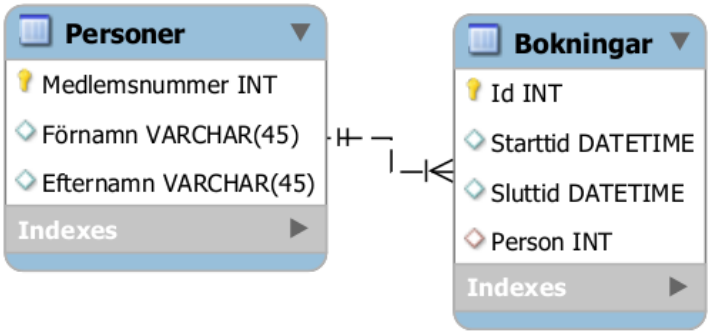
Systemet består huvudsakligen av fyra komponenter, användarhantering, meddelandehantering, bokningshantering och fakturahantering. Användarhanteringen, meddelandehanteringen och bokningshanterin-

gen kan användas utan fakturahanteringen om kunden inte är intresserad av fakturor, eller vill hantera detta på annat sätt.

3.1 Datalagring

Den första och kanske viktigaste komponenten i bokningssystemet är datalagringen. Denna komponent har som ansvar att säkert lagra data som resten av systemet behöver för att fungera. I gruppens system implementerades denna komponent som en MySQL-databas.

MySQL är en relationsdatabas där data lagras i olika tabeller och binds samman av relationer. Syftet med relationsdatabaser är att inte lagra alla data i en enda stor tabell, utan att dela upp data och lagra dessa i relevanta tabeller.



Figur 2: Exempel på tabeller med relation i MySQL.

Varje tabell har ett bestämt antal kolumner där data kan lagras. För en tabell med personuppgifter skulle kolumnerna kunna vara medlemsnummer, förnamn och efternamn. En bokningstabell skulle exempelvis kunna ha kolumnerna starttid, sluttid, kommentar och vem det var som gjorde bokningen. I en relationsdatabas kan man istället för att ha alla personuppgifterna i bokningstabellen, lagra något som är unikt för varje medlem och sedan slå upp i personuppgiftstabellen för att få ut mer information om en person. Något som skulle kunna vara unikt för en person är medlemsnumret. Det skulle då gå att slå upp sidoinformation angående användaren genom att titta på de raderna i andra tabeller som pratade en användare med användarens medlemsnummer.

3.2 Dataåtkomst

Utöver att lagra data behöver systemet också kunna komma åt lagrade data på ett sätt som för utvecklaren är enklare än att direkt skriva frågor till databasen. Detta delsystem underlättar ramverket Yii med, eftersom det tillhandahåller många bra sätt att generera SQL-frågor utan att programmeraren behöver skriva dem.

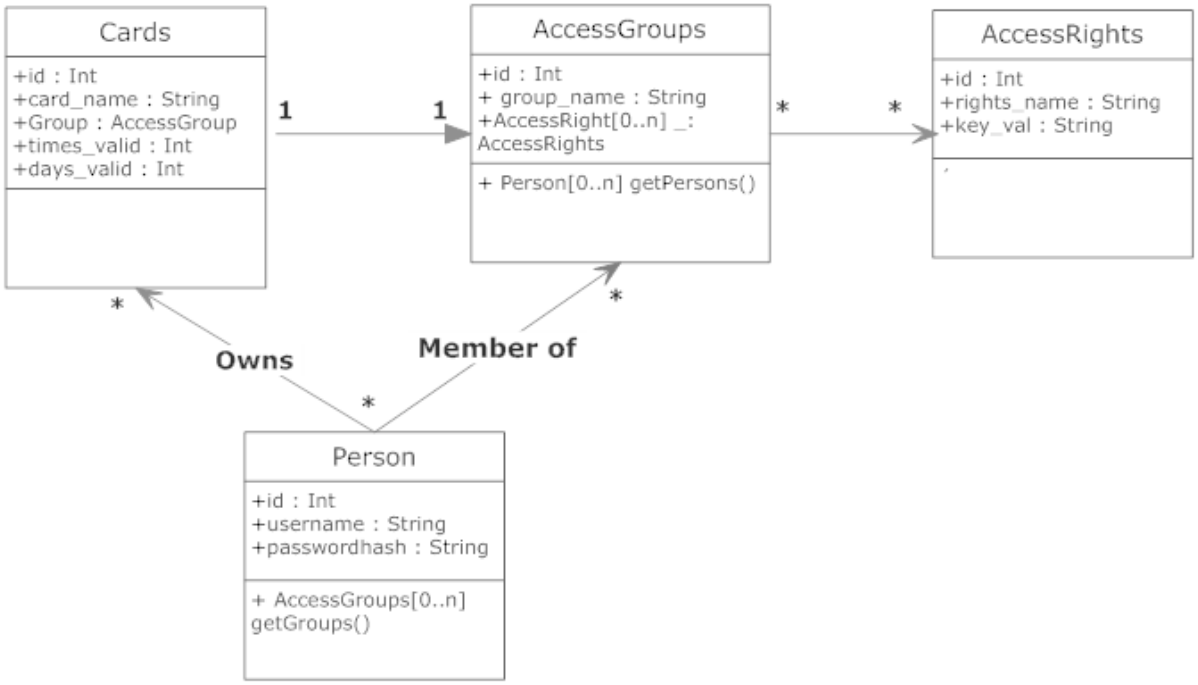
Yii arbetar med designmönstret MVC, vilket innebär att logiken för datahantering, användargränssnitt och styrlogik separeras (3). Modeller i Yii är ett objekt med tillhörande data. Läsning och modifiering av data sker på samma sätt som ett vanligt objekt i PHP, men Yii kommer att utföra de nödvändiga SQL-anropen för att hämta data från databasen.

I modellerna för de olika tabellerna går det även att definiera relationer. Detta förenklar hämtning av information som som en tabell refererar till. Exempelvis går det, om det finns en bokningstabell som refererar till en persontabell, att få ut personuppgifter genom bokningsmodellen. Detta är användbart om man vill få reda på mer om personen som har gjort en viss bokning.

Modellerna utför även validering för att se till att data som skrivs in är korrekt. Validering är viktigt då det på andra ställen i koden kan antas att viss data är på någon form. Ett exempel är att en räknare ska ha

numeriska värden. Att inte validera kan ge andra följdfel för att data som finns är felaktiga: till exempel kan ett namn stå i fältet istället för det förväntade talet. Genom att validering sker i modellerna samlas all validering på ett och samma ställe. Alternativet är att validering sker vid inläsning. Att validera vid inläsning gör att validering måste spridas ut till alla ställen där ett värde kan hamna i modellen. Risken att glömma validera ökar om valideringen sker på olika ställen.

3.3 Användare



Figur 3: Klasstruktur för användares relation till grupper och rättigheter.

För att skapa ett konfigurerbart rättighetssystem valde gruppen ett gruppbaserat rättighetssystem där användaren får rättigheter genom de grupper han är medlem i. Som man kan se enligt klassdiagrammet, kan en användare bli medlem i en grupp antingen direkt eller genom de kort han äger. På så sätt kan en administratör skapa automatisk uppgradering av användare som har mer exklusiva kort och då ge dem extra rättigheter utan att själv behöva ge användare både kort och medlemskap i en grupp. Ett strikt hierarkiskt system med olika typer av superanvändare övervägdes också att implementera. Det valdes bort då det inte gav samma flexibilitet samtidigt som att systemet som valdes leder till att gruppen dessutom kan lägga över delar av gränsdragningsproblematiken på systemadministratören, som förhoppningsvis känner till verksamheten.

3.3.1 Inloggning och lösenordshantering

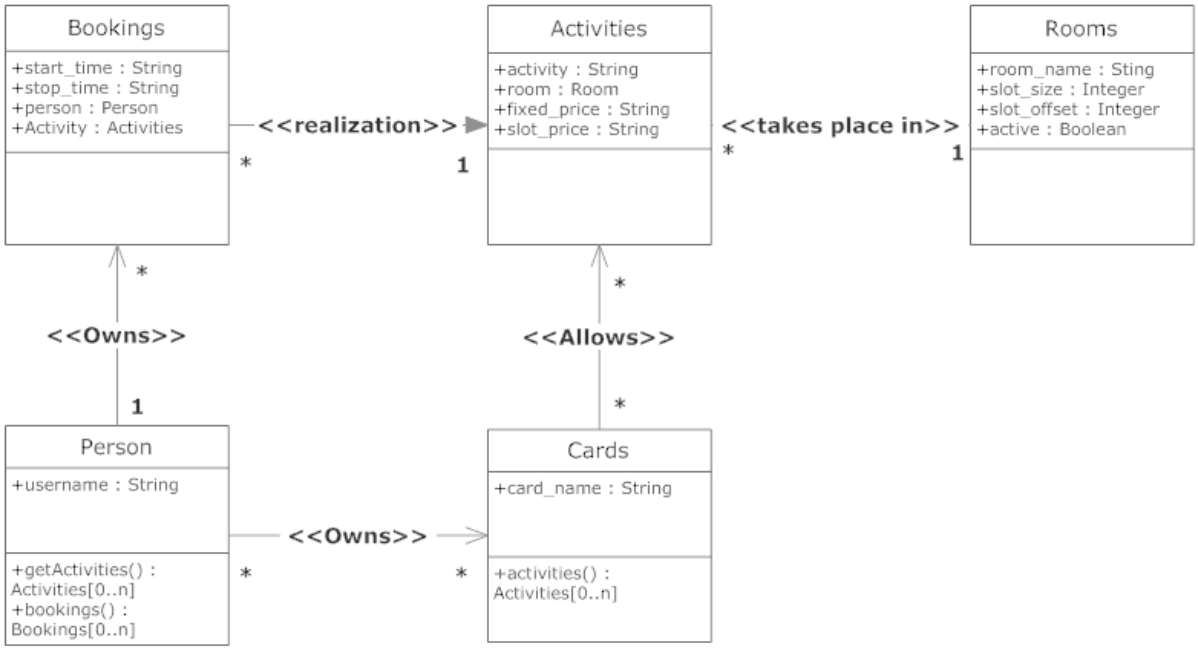
För en säker lösenordshantering krävs det att system inte lagrar användarens lösenord. Istället används Blowfish-chiffret för att lagra en hashad sträng (4). När en användare sedan ska logga in sig, hashas det lösenord han använder för att verifiera sig. Sedan jämförs det med den lagrade strängen. Överensstämmer de antas det att rätt lösenord har använts.

3.3.2 Virtuella användare

För att kunna skilja på användaren och vissa roller som en användare kan tänkas anta, implementerades även virtuella användare. Det användningsfall som tänks, där detta kan vara relevant, är då en person på ridklubbens kansli ska göra en bokning åt klubben. Då kan han växla från sitt personliga konto till klubbens konto, som är en virtuell användare, och sedan agera i systemet i klubbens namn.

Virtuella användare identifieras systemmässigt genom att lösenordsträngen är tom. Då inget lösenord hashas till en tom sträng är det då omöjligt att direkt logga in som en virtuell användare och möjligheten ges endast till inloggade användare med den rättighet som skapades i samband med att den virtuella användaren skapades.

3.4 Lokaler och aktiviteter



Figur 4: Klasstruktur för användares relation till bokningar och lokaler.

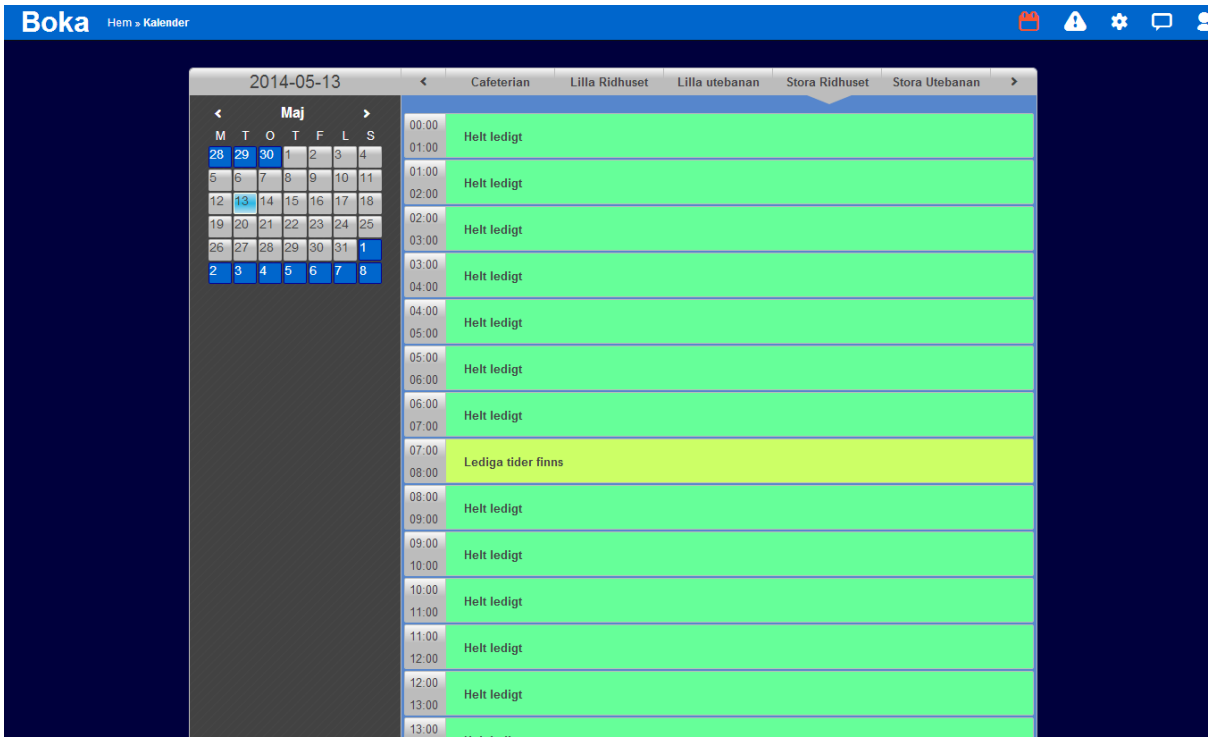
Systemet hanterar i grund och botten bokningar i ett antal lokaler. De lokaler som finns tillgängliga läggs till när systemet sätts upp för första gången och är sedan statiska. Lokaler kan dessutom läggas till och avaktiveras i efterhand.

Varje lokal har i sin tur en eller flera aktiviteter associerade till sig. Aktiviteter beskriver de bokningsbara aktiviteterna i lokalen. En sådan lokal kan för referenskunden vara exempelvis hoppning eller longering. Varje aktivitet tar upp ett förbestämt antal platser i lokalen. I samband med att varje lokal har en förutbestämd storlek, gör detta att bokningssystemet kan kontrollera att en lokal inte blir överbokad, men ändå tillåta optimalt utnyttjande av lokalen. Priset på bokningen anges per aktivitet, vilket gör att olika aktiviteter kan kosta olika mycket, även om de tar upp lika mycket plats i lokalen.

3.5 Kalender

Kalenderdelen av systemet håller reda på bokningar och visar dem för användaren. En bokning binds till exakt en person och en aktivitet. Eftersom aktiviteter är exklusiva till rum, innebär aktivitetsbindningen att bokningar kan spåras tillbaka till rummet där de bokades. En bokning tar upp ett av de

möjliga tidsutrymmena för aktiviteten som bokades. Om ett rum eller en aktivitet redigeras, påverkar inte det bokningar som redan har lagts i rummet eller med aktiviteten. Start- och sluttid sparas undan, tillsammans med hur många platser bokningen tog upp i lokalen.



Figur 5: Den nya kalenderns utseende.

På själva grafiska kalendern, systemets förstasida, visas tidsutrymmen i ett rum i taget. Per tidsutrymme får användaren se om det ligger några bokningar där, eller om det är tomt eller fullbokat. Innan en bokning kan läggas i ett tidsutrymme som inte är fullbokat, får användaren se en lista över bokningarna under tiden som har valts. Detta görs först på skärmen där användaren hamnar efter att ha klickat på tidsutrymmet för att spara plats och kunna visa mer information per bokning.

Stående bokningar är en funktion som är tänkt att bara kunna tillgås av systemadministratörer. De fungerar som vanliga bokningar med det undantaget att många kan läggas samtidigt med en valbar upprepningstid. Det tillåter systemadministratörer att boka till exempel städning varje torsdag klockan fyra på eftermiddagen utan att behöva gå genom det vanliga systemet. En annan fördel med stående bokningar gentemot vanliga bokningar är att alla stående bokningar som läggs samtidigt också kan tas bort samtidigt. Om städning flyttar till klockan fem varje torsdag, är det alltså lätt för administratörer att ta bort alla bokningarna som motsvarar städning klockan fyra.

3.6 Kort

I systemet används kort som en form av rättigheter. En användare kan tilldelas ett eller flera kort under en viss period av en systemadministratör. Ett kort tillåter i sin tur att alla användare som har kortet kan boka de aktiviteter som kortet är associerat med. Kortet kan dessutom ge rabatt på aktiviteter. Rabatter kan vara relativa rabatter (exempelvis 20 %), eller en absolut rabatt (exempelvis 20 kronor).

Utöver att kort har en giltighetstid, kan korten också vara giltiga för ett förutbestämt antal bokningar. Detta gör att så kallade klippkort kan implementeras genom att ge en användare ett kort som är giltigt ett bestämt antal gånger, men med 100 % rabatt på vissa aktiviteter.

3.7 Meddelanden

För att kunna notifiera användare om förändringar till deras bokningar finns också ett meddelandesystem. Meddelandesystemet tillåter också att en användare måste bekräfta att de läst meddelanden innan de får fortsätta använda systemet. Alla meddelanden skickas också till den e-post adress användaren använde vid registrering. Användare med tillräckliga rättigheter kan också skicka meddelanden till användare eller grupper för att snabbt och enkelt nå ut med viktig information.

3.8 Fakturering

Fakturering kontrolleras per bokning på ett sådant sätt att det markeras för varje boking när en faktura som innehåller bokningens kostnad skickas ut och betalas. På så sätt är det omöjligt att få samma bokning på två fakturor. Användare kan därmed vara säkra på att de bara behöver betala varje kostnad en gång. Detsamma gäller för en användares kort.

3.9 Utseende

Kod för systemets utseende skrivs endast i CSS. Ingen Javascript används utanför vad ramverket Yii använder. För bästa utseende bör användaren köra det i en modern webbläsare med stöd för CSS3. Det finns dock andra utvägar för de webbläsare som inte stödjer CSS3. Kod för utseendet separeras från systemets logik så mycket som möjligt. På vissa ställen genereras kod automatiskt efter databasinnehåll. På dessa ställen är det en smärre omöjlighet att separera all utseendemässig kod från logiken.

I systemet ingår också en temaväljare. Teman består av prioriterade komponentfärger som skapar ett unikt färgschema för systemet, liknande Twitter Bootstrap (5). Egna teman för systemet innebär större möjlighet för användare att göra systemet mer personligt. Dessa teman kan väljas i inställningarna för användare. I och med att Javascript undviks och att utseende och logik separeras, underlättar det för eventuella webbdesignrar som inte har stor kodvana men ändå vill göra ett eget tema i CSS.

Rent tekniskt är systemets utseende sammansatt av vyerna i det MVC-mönster som ramverket Yii använder. I dessa vyer anropas komponenter som i sin tur genererar den HTML-kod som behövs. Yii ansvarar för att generera länkar och bildelement.

4 Gruppens tekniska erfarenheter

Under denna rubrik gås gruppens tekniska erfarenheter och vad gruppmedlemmarna har lärt sig igenom.

4.1 Programmeringsspråk

Gruppen har huvudsakligen arbetat med fyra olika programmeringsspråk: PHP, HTML, CSS och SQL.

4.1.1 PHP

PHP har främst använts för funktionaliteten i systemet, där det likt många andra programmeringsspråk har stöd för funktioner och att skicka med parametrar mellan olika delar eller moduler. Alla i gruppen har kommit i kontakt med PHP under projektets gång. Det har till och med varit så pass centralt att vissa av projektgruppens medlemmar i princip enbart har skrivit PHP-kod. Att PHP är likt andra objektorienterade språk som C++ och Java, som gruppens medlemmar har erfarenhet i sedan tidigare, medförde att gruppen kunde komma igång snabbt. Andra faktorer ledde dock till att det inte blev helt enkelt till en början, då det valda ramverket Yii har många specifika särdrag och funktioner som man i PHP-koden behöver ta hänsyn till och använda sig av.

4.1.2 HTML

HTML ligger som grund för visning och uppritning av de olika sidorna i systemet, och ramverket Yii tillhandahöll HTML-kod som behövdes för att komma igång. Gruppens designansvariga har efterhand bytt ut och utökat mycket av den HTML-kod som fanns. Resten av gruppen har också behövt lära sig grunder i HTML om de inte redan haft erfarenhet av det, eftersom de också behövt skapa innehåll.

4.1.3 CSS

I språket CSS skrivs stilmallar som ska gälla för olika delar av systemet. CSS används genom hela systemet och har främst skrivits av de designansvariga i gruppen. Användandet av CSS medför att utseendet på sidor blir enkelt att byta och kan göras enhetligt genom hela systemet.

4.1.4 SQL

Detta språk används för databashantering, som är mycket centralt för systemet. Yii tillhandahåller mycket på just denna front, i form av ett gränssnitt mot databasen med inbyggda funktioner som är enkla och smidiga att använda. Detta gränssnitt har förenklat läsning och skrivning i databasen mycket. För mer avancerade läsningar från databasen har gruppen dock behövt ställa SQL-frågor, där man skickar in krav på data för att få ut önskade värden. I det stora hela har ramverket medfört att inga större mängder SQL-kod har behövt skrivas, men även i detta språk har det för de flesta i gruppen varit viktigt att ha grunderna i språket klara för sig. Gruppen har alltså mestadels fått lära sig hur ramverket förespråkar att man arbetar mot databaser.

4.2 Ramverket Yii

Tidigt i projektet togs beslutet om att ett PHP-ramverk skulle användas som en grund och ett sådant behövde väljas för att passa in på den specifika uppgiften. Valet hamnade på ramverket Yii, som lämpade sig då det fungerar bra på både mindre och större projekt. Det använde sig även av objektorientering och programspråket PHP, vilket gruppen också kände talade för ramverket i fråga. I efterhand tycker gruppen att valet var bra även om det kanske har skapat en del huvudvärk på vägen. Det finns en hel del väldigt specifika saker för Yii, till exempel hur själva MVC:n fungerar, hur man kommer åt data från databasen och Yiis egna så kallade Widgets, som används för att skapa speciella vyer. Detta medförde en sorts tröskel, men då man kommit över denna flöt arbetet på mer och mer tills dess att man till slut kände sig hemma med ramverket och kunde arbeta tillsammans med det istället för mot det.

4.3 Versionshantering med Git

För versionshantering har gruppen använt sig av Git. Stor nytta har kommit ut detta under hela projektet. En gruppmedlem satte först upp systemet och därefter har resterande gruppmedlemmar endast behövt lära sig enkla kommandon för att hämta ifrån eller lägga upp kod på den centrala Git-servern. Alla har haft en egen gren som använts för att skriva kod lokalt och sedan kontinuerligt sammanslagit koden med huvudgrenen (master) för att snabbt kunna hitta eventuella buggar då kod slås ihop. Har konflikter uppstått då någon slagit ihop sin kod med Git-servern så har detta lösts genom att prata med personen som skrivit den andra koden, här har det hjälpt att vi suttit tillsammans. Hela processen med versionshanteringen har fungerat väldigt bra och varit till stor nytta.

5 Gruppens processrelaterade erfarenheter

Gruppen har i huvudsak använt två olika processer som har ett stort antal likheter. En process har funnits för att författa och revidera dokument och en annan för att utveckla bokningssystemet.

5.1 Dokumenthantering

Gruppens modell för dokumenthantering har fungerat bra i projektet eftersom den har hjälpt till att producera stora dokument på förhållandevis kort tid, utan att en enskild individ har behövt lägga ner oproportionerligt mycket tid av den som krävdes för att författa dokumentet. Metoden fungerar därför också bra då flera dokument behöver sammanställas under samma tid, där de flera dokument innehåller delar som en viss person är bäst lämpad att skriva.

Nackdelarna med metoden är att det blir svårare för alla deltagare att få en bra helhetsbild av dokumentet. Detta kan leda till repetition av förklaringar och definitioner, men även saker som inkonsekvent benämning av saker och inkonsekvent språkbruk. Dessa brister har gruppen hanterat genom revideringsmötet i punkt 4, men också genom att se till så att alla i gruppen enkelt har tillgång till den senaste versionen av dokumentet via webbplattformen Trac.

Eftersom metoden innehåller två relativt omfattande möten tar metoden relativt mycket tid i anspråk för att få fram färdiga dokument. Till de dokument gruppen har skrivit har drygt en vecka används från start till slut, vilket är acceptabelt för större dokument som projektplan och kravspecifikation. För mindre dokument är det inte fördelaktigt att göra på detta sätt, eftersom en eller två personer antagligen kan göra ett bättre arbete på kortare tid och med mindre arbetsbelastning.

5.2 Utvecklingsmetod

Gruppens utvecklingsmetod har i stort sett fungerat mycket bra. De regelbundna kodstugorna har underlättat kommunikationen inom gruppen väldigt, vilket har lett till att den formella hanteringen av ärenden i Trac mest har fungerat som en minneslista över saker som skulle göras. I stället för att spendera tid med att tydligt formulera vad som ska göras, har det varit enkelt för alla att se på rubrikerna och sedan fråga en annan gruppmedlem om vad som ska göras. Detta har sparat mycket tid, eftersom gruppen har kunnat fokusera på att bygga vidare på systemet i stället för att lägga tid på att beskriva vad som ska göras.

Den korta, enkla och något informella beskrivningen av ärenden som användes i projektet sparade inte bara tid, den bidrog också till att utvecklare hellre lade upp nya ärenden på Trac än att bara skriva ner vad som borde göras någonstans för sig själva. Risken med att kräva en formell och detaljerad beskrivning på alla nya ärenden är att utvecklare oftare tycker att det inte är värt arbetsinsatsen att publicera ett ärende, vilket i sin tur leder till att små uppgifter har större sannolikhet att glömmas bort.

Det finns också nackdelar med den informella hanteringen av ärenden. Det syns tydligt om kodstugorna inte fungerar väl, eftersom det då inte går snabbt och enkelt att be en annan gruppmedlem att förtydliga ärenden. Att det inte går att sitta tillsammans kan bero på flera saker, antingen att gruppen är för stor för lokalen, eller att gruppen helt enkelt inte inser värdet av att vara på plats, gentemot att exempelvis arbeta hemifrån. Det senare kan också leda till en ond cirkel: om majoriteten av gruppen inte är på plats regelbundet, försvinner också fördelarna med att vara på plats för resten av gruppen, vilket i sin tur leder till att ännu färre är på plats.

Ytterligare en nackdel med denna metod är att det, även om det är enkelt för alla gruppmedlemmar att få en översiktlig bild över vad alla håller på med, är betydligt svårare att i efterhand gå tillbaka och undersöka vad som har gjorts och vad som har tagit tid. Detta har också märkts då gruppen har tagit fram tidsplaner för kommande iterationer. I dessa tidsplaner har det bara funnits uppskattningar utifrån hur lång tid andra saker har tagit innan. Detta har inneburit att det har varit svårt att justera uppfattningen om hur lång tid saker kommer att ta utefter projektets gång.

6 Individuella bidrag

Som en del av kandidatrapporten skulle samtliga gruppmedlemmar skriva en individuell mindre rapport som ingick i den stora kandidatrapporten.

6.1 Nils Axelsson

Som designansvarig var min uppgift under projektet delad. Precis när projektet började fanns knappt någonting att designa utseendemässigt. Ju närmre projektet kom sitt slut, desto mer gick min roll över till ren design av hur systemet såg ut. Som koppling till min uppgift som designansvarig tänker jag diskutera vad för åtgärder som kan tagas för att undvika att webbsidan som man designar bara fungerar i vissa webbläsare och vad för element på en sida som löper större risk att inte fungera i alla.

Även om min roll tidigt i projektet formellt var som designansvarig för den *mobila klienten*, rörde sig projektet redan efter den första iterationen mot att sidan skulle designas så att samma design fungerade på både mobila och ickemobila enheter. Därmed tog jag på mig ungefär lika mycket ansvar för det som Andreas, den andre designansvarige och därför flöt våra roller ihop något.

6.1.1 Frågeställning

Följande frågor försöker besvaras som resultat:

- Vilka olika sorters programmering ingår i projektets webbdesign och vilka av dem löper vilken risk att inte fungera i vilka webbläsare?
- Hur testar man effektivt olika webbläsare på sin sida?

6.1.2 Metod

Varken jag eller Andreas, som var den andre designansvarige, hade någon större erfarenhet av CSS innan projektet. När vi skulle lösa designproblem ledde detta till att vissa designelement gjordes med mer komplicerade CSS-attribut och HTML-taggar än vad som behövdes. Detta skedde framför allt tidigt i projektet Projektmedlemmarna satt och utvecklade sidan både i Googles Chrome-webbläsare och Mozillas Firefox-webbläsare, två webbläsare som använder olika renderingsmotorer (Firefox använder *Gecko* (6) och Chrome använder *Blink* (7)). Vid ett par tillfällen upptäcktes därför designelement som fungerade på olika sätt i de två webbläsarna genom att någonting fungerade för en gruppmedlem och inte för en annan, trots samma siddesign.

Senare under projektet började sidans design testas i Microsofts Internet Explorer-webbläsare. Det var kravsatt att bokninghemsidan som gruppen tillverkade skulle fungera i version 9 av denna webbläsare. Kravet kom från att kunden kunde visa data på att en majoritet av systemets användare skulle sitta i Internet Explorer 9 eller äldre. Av detta skäl satt jag som designansvarig mot projektets slut och testade varje designelement som jag designade i Internet Explorer 9.

6.1.3 Resultat

Här besvaras varje fråga ur frågeställningen utifrån det som jag upplevde när jag arbetade enligt de arbetssätt som beskrivs i *metod*.

Vilka olika sorters programmering ingår i projektets webbdesign?

CSS lades till HTML-standarden i samband med att HTML 4.0 släpptes (8). *Color*- och *font*-taggarna från HTML-versioner tidigare än 4.0 finns kvar i specifikationen, för att HTML skall vara bakåtkompatibelt. Tanken sedan dess är dock att all design på en sida skall bestå av CSS. Olika webbläsare greppade CSS-standarden långsammare, men redan innan CSS var det olika standard om vad olika grafiska HTML-taggar gjorde (9 s. 429-520).

CSS kan kopplas till ett HTML-dokument på flera sätt. Sättet som utjyttjades mest i bokningssystemet var att lägga CSS-koden i separata filer i en mapp, så att nästan all CSS låg på samma plats och sedan inkludera alla CSS-filerna på en och samma plats. PHP-ramverket Yii hade stöd för att ladda alla CSS-filer som inkluderades i en viss fil. Detta tillät att gruppen inte behövde skriva hela listan överst i varje

.php- eller .html-fil, vilket i sin tur gav att många fler .css-filer kunde skapas på ett sådant sätt att varje fil bara innehöll utseendet för en viss del av sidan.

Ett annat sätt att lägga till CSS till element i HTML är att skriva det direkt i HTML-taggen. CSS och HTML har stöd för att följande görs:

```
<td style=''height: 200px''>
```

Style-attributet inuti *td*-taggen skulle i detta fall applicera CSS på taggen i fråga så att tabellrutan (taggen *td*) skulle bli 200 pixlar hög.

I bokningssystemet användes dessa två sätt att applicera CSS på HTML-element. Grafiskt Javascript användes inte direkt i projektet, men i vissa fördefinierade element som ingick i Yii använde det. I dessa fall ansvarade Yii för att *jQuery*-kod och liknande fungerade i webbläsare längre tillbaka än Internet Explorer 9 (10).

Element som bakas in i HTML och tar upp en viss plats, men i de flesta fall inte påverkar sidan runtomkring, är element som Flash och webbläsarplugin. Ingen del av projektet, varken Yiis fördefinierade delar eller bokningssystemet, använde några sådana element.

Vilka av dem löper vilken risk att inte fungera i vilka webbläsare?

Det mesta av CSS:en som projektet använde fungerade bra i Internet Explorer 9. När ett fel inträffade, var det oftast ett mindre fel i en stor komponent som gjorde att en stor del av sidan hamnade på en oväntad plats. Små grafiska fel som inte ansågs vara viktiga nog för att lista ut hur man kunde lösa (om alls) i Internet Explorer 9 dök också upp här och var. Det största sådana felet var att rundade hörn helt försvann på ramar runt element. Ingen designansvarig ansåg alltså detta fel som så allvarligt att det behövde fixas. I Internet Explorer 10 dök hörnen upp.

Ett fel som upptäcktes sent i projektet var att Internet Explorer 9 inte tolkade *z-index*-kommandot på samma sätt som webbläsarna som projektgruppen satt med. I projektgruppens webbläsare är *z-index* ett globalt kommando: en *div*-tagg som har *z-index* 1000 kommer alltid dyka upp framför andra element med lägre *z-index*. I Internet Explorer 9 är *z-index* relativa, på ett sådant sätt att *z-index* endast beräknas utifrån element som ligger inuti samma element. CSS-problemet kunde inte lösas då Internet Explorer 9 inte hade något sätt att uttrycka djuprelationen som behövdes. I slutändan gjordes behållaren som de dolda elementen låg i helt enkelt större, så att allt syntes i alla webbläsare.

Ett återkommande fel var hur mycket utrymme olika webbläsare lade till höger och vänster om element. I CSS är egentligen tanken att man väldigt strikt skall kunna styra exakt hur många pixlar utrymme finns innanför, på och utanför en ram runt ett element. Vid en punkt lade Internet Explorer 9 till extra utrymme till höger och vänster om ett element på ett sådant sätt att elementet bildade en ny rad, trots att elementet, i detta fall listan över bokningar på en viss dag, hängde ihop med en kalender där man kunde välja vilken dag som skulle visas. Lösningen blev att ge elementet något större marginal att växa till höger och vänster om elementet. Detta är inte en heltäckande lösning och tyvärr någonting man behöver testa för alla komplicerade uppställningar av element i alla webbläsare.

Ett tredje, mindre stort, fel som dök up var att Internet Explorer 9 lade rull-lister (precis sådana som dyker upp till höger i webbläsaren när sidan är för hög för fönstret) inuti element som blev för stora och var inställda till att få en rull-list i sådana fall och att samtliga andra webbläsare som testades lade sina rull-lister på utsidan. Så fort som elementets innehåll flödade över blev alltså elementet lika mycket större som rull-listen tog plats i alla webbläsare utom Internet Explorer. Eftersom rull-listens storlek är någonting som kan vara större eller mindre i olika webbläsare och skillnaden därför inte nödvändigtvis går att förutsäga, är nästan Internet Explorers lösning i detta fall att föredra.

6.1.4 Slutsats

Det finns mindre skillnader mellan hur mer avancerade webbläsare som Chrome och Firefox hanterar olika CSS. Skillnaderna dem emellan är dock i princip försumbara i jämförelse med hur stora skillnader

det är till hur Internet Explorer 9 hanterar olika element. Det är inte nödvändigtvis en lösning att utveckla hela sidan med Internet Explorer 9 som målwebbläsare. Internet Explorer 9 hanterar nämligen inte nödvändigtvis element alls likadant som de andra webbläsarna, så en hemsida som har designats så kanske inte fungerar alls i den senaste versionen av Chrome.

Därför är det idealt att försöka uppnå följande punkter i ett projekt:

- Diskutera upp kravet på vad för version av Internet Explorer som behöver utvecklas för. Om gruppen lyckas diskutera sig till Internet Explorer 10 istället för 9, blir det både mycket säkrare att stora delar av samma design fungerar på alla webbläsare och att någon i gruppen kan få tag på webbläsare. Internet Explorer 9 går knappt att ladda ner, men går att emulera via vissa sidor på internet.
- Hitta det enklaste sättet att lösa problem. Ju fler CSS-knep man använder för att lösa någonting, desto större är risken att någon webbläsare tolkar det som man har gjort annorlunda. Därför är det viktigt att man hittar det enklaste, eller om inte det i alla fall det mest stödda, sättet att lösa problemet.
- Testa sidan i många stora webbläsare. Även om data säger att bara en procent att sidans användare kommer att använda Internet Explorer, är inte det anledning nog att utveckla en sida som inte fungerar för dem. Om en hemsida inte fungerar i en webbläsare kommer det inte leda användaren att byta webbläsare, det kommer driva användaren att byta sida till en som fungerar.

6.2 Gustaf Brunberg

Gustaf har varit dokumentansvarig i gruppen. I denna del av rapporten kommer därför referenser till ”jag” innebära Gustaf Brunberg.

6.2.1 Inledning

Som dokumentansvarig har jag tagit fram dokumentmallar för bland annat projektplan och kravspecifikation.

Till dessa dokument användes dokumentmallar från LIPS-modellen (11 s. 13-19). Denna modell användes av alla gruppens medlemmar i en tidigare projektkurs. Detta underlättade mitt arbete något då alla lätt kunde förstå de rubriker som fanns i dokumenten. Eftersom mallarna för kravspecifikation och projektplan följer IEEE-standarden, kunde de modifieras efter gruppens behov (12 s. 211-227). Andra dokument som skapats följer, även de, standarden för dessa dokument. Detta gjordes då det blev enklare att färdigställa även dessa dokument om alla dokumenten följde samma standard.

Utöver detta har jag även agerat sekreterare under de flesta möten som gruppen haft. Det som skrivs under möten har jag sammanställt och skickat ut till alla dels via mail. Jag har efter detta också lagt upp sammanfattningen på Trac.

Frågeställning

Jag har valt att kontrollera dessa frågor:

- Hur fungerar det att dela upp skrivandet av dokumentation?
- Hur bra fungerar Tracs wikifunktion jämfört med Google Docs vid parallellt arbete?

6.2.2 Metod

Forskningsmetoden som jag använt för att samla in data till detta kapitel består egentligen endast av egna erfarenheter och observationer. En del av den information jag använder kommer från andra medlemmars åsikter om dokument eller andra moment som behandlas i detta kapitel.

Något som underlättat mitt arbete som dokumentansvarig har varit Tracs wiki-funktion. Denna funktion har gjort att alla våra dokument funnits tillgängliga för alla gruppens medlemmar under hela projekttiden. Trac har även tagit hand om versionshanteringen av dokument. Dokument som behövts har skapats på Trac av gruppmedlemmen som funnit dokumentet nödvändigt. Som nämndes i inledningen, har alla gruppens dokument utgått ifrån tidigare dokument på Trac och därför följer alla gruppens dokument en övergripande standard. Dokumenten har senare kunnat laddas ner från Trac som PDF:er via *LaTeX*. Detta gör att alla dokument som laddas ner har samma standard för till exempel sidhuvud, tabellförteckning och innehållsförteckning.

Eftersom hela gruppen har varit aktiv i att skriva dokument har det aldrig uppstått försening i varken själva dokumentskrivandet eller annan aktivitet.

6.2.3 Resultat och slutsats

I detta kapitel sammanflätas resultatet ifrån mina frågeställningar och slutsatsen. Diskussion angående resultat och metod återfinns också i detta kapitel.

Hur fungerar det att dela upp skrivandet av dokumentation?

Det har fungerat bra att dela upp skrivandet av dokument. Uppdelningen har ofta skett enligt dokumentprocessen som beskrivs i kapitel 2.2.

En medlem i gruppen skapar ett dokument och stolpar endast upp rubriker till dokumentet. Sedan delegeras rubrikerna ut till dem som är mest lämpade. Detta har gjort att det är enklare för gruppen att se vem som inte gjort sitt arbete och även gjort att dokumentation aldrig fördröjt andra delar av projektet. Ett problem som kan uppstå vid detta är inkonsekventa benämningar av komponenter och repetition av definitioner samt förklaringar. Detta har lösts genom punkt 4 i dokumentprocessen, där gruppen alltså har ett revideringsmöte.

Som det beskrivs i kapitel 2.2, tar det gruppen ungefär en vecka att färdigställa ett dokument oavsett storlek. Att det tar en vecka för stora dokument såsom kravspecifikation och projektplan är bra, men inte lika bra då mindre dokument ska färdigställas. Revideringsmöten för större grupper kommer inte heller fungera i längden. Ett möte på två timmar för åtta personer tar redan där 16 timmar. Om gruppen skulle växa till att vara till exempel 15 personer skulle samma möte då dra 30 timmar från andra delar av projektet.

Uppdelning av dokumentation är alltså något som går att göra. Det är dock inte att föredra om gruppen skulle vara allt för stor. Då skulle man kanske kunna dela upp gruppen i mindre grupper som var och en får ett eget dokument att ansvara för. Resultatet av detta skulle dock kunna bli att folk inte vet vad som står i de olika dokumenten då det tar tid från annat att läsa dem.

Hur bra fungerar Tracs wikifunktion jämfört med Google Docs vid parallellt arbete?

Tracs wikifunktion fungerar bra då man ska dela dokument med en grupp där alla ska ha möjlighet att redigera. Det finns alternativ för just denna funktion, till exempel Google Docs. Däremot innehåller Trac annan funktionalitet som inte finns på Google Drive, till exempel möjlighet att hantera milstolpar och ärenden för olika iterationer. Tracs wikifunktion saknar dock möjligheten att arbeta parallellt, något som till exempel finns i Google Docs. Om man bara ska dela dokument med varandra där alla ska ha möjlighet att redigera vid exakt samma tidpunkt är nog Google Docs att föredra. Google Docs har stöd för upp till 50 st användare som redigerar samtidigt (13).

Något som är värt att notera är problemet med att man inte kan arbeta parallellt endast inträffat ett fåtal gånger för projektgruppen. Detta har inträffat då gruppens medlemmar själv fått bestämma när de ska skriva snarare än att alla sitter tillsammans och skriver. Om gruppen hade varit större eller suttit mer tillsammans och skrivit så hade detta antagligen varit ett mycket större problem och gruppen hade nog snabbt bytt till ett annat verktyg för att skriva dokument.

Något som skulle få Tracs wikifunktion att fungera bättre är möjlighet att slå ihop olika dokumentversioner snarare än att användaren själv måste skriva in alla förändringar för hand, ungefär som Git arbetar med ihopslagning av konflikter (se kapitel 4.3 för mer information). Skulle detta göras skulle det inte bli ett problem med möjligheten att arbeta parallellt.

Huruvida Tracs wikifunktion fungerar bra beror alltså på hur man definierar ”parallellt arbete”. Definieras det som i att man arbetar vid exakt samma tidpunkt kommer det orsaka problem. Detta beror alltså på problem vid ihopslagning av olika dokumentversioner. Lösningen på detta problem beskrivs i föregående stycke. Skulle man däremot definiera det som i hela projekttiden skulle Trac antagligen hålla måttet. Det beror på gruppens arbetsmetod och storlek. För projektgruppens dokumentmetod fungerar Tracs wikifunktion bra.

Diskussion över resultat och metod

Min forskningsmetod går helt klart att förbättra. Nu fungerar den dock för att gruppen inte består av mer än åtta personer som jag träffar dagligen. Hade gruppen varit större eller inte träffats lika ofta, hade informationen som jag fått in kunnat ge dåliga resultat. En lösning på detta hade kanske varit att lämna ut en enkät som jag bad alla fylla i efter ett färdigskrivet dokument eller dylikt.

Resultatet som tagits fram är också lite missvisande då mycket består av egna erfarenheter och observationer. Till exempel finns problem med jämförelsen mellan Tracs wikifunktion och Google Docs. För att få ett bättre resultat här hade man behövt dela upp gruppen i små grupper och se hur de skilt sig åt för att kunna ge ett bra resultat. Detta är dock inget som kunnat göras i detta projekt då det hade tagit för mycket tid ifrån det faktiska projektet. Nu består resultatet till största del på egna erfarenheter mellan Tracs wikifunktion och Google Docs.

Angående om det är bra att dela upp dokument, är det också något som skulle behövas kontrolleras i ett flertal grupper av varierande storlek. Nu består vår grupp av åtta personer och det finns inget som säger att arbetet med dokumentation hade fungerat varken bättre eller sämre om gruppens storlek varierat.

6.3 Andreas Larsson

Som delansvarig för design kändes det bra att välja ett projekt där det fanns mycket spelrum för att designa fina komponenter. Mycket av den första tiden i projektet ägnades åt att fastställa vilka krav som skulle finnas på systemet. Det var en process där alla måste vara med men som ändå kändes fruktansvärt överbemannad. Mycket arbete i början lades på projektledare, dokumentansvarig och analysansvarig. Därför passade jag på att lära mig grunderna för CSS och HTML under tiden samt att utvärdera vilka ramverk som skulle passa projektet. Under projektet ritade jag upp en gränssnittsskiss (på engelska, *wireframeskiss*) över en prototyp till systemet som jag tillsammans med Behnam skapade en interaktiv prototyp av. Därefter skapade jag tillsammans med Nils (också designansvarig) majoriteten av den CSS som nu används i systemet.

6.3.1 Frågeställning

De frågor jag ställt under projektet och framförallt i projektets sista iteration är följande:

- Hur mycket tid ska läggas på utseendet av en applikation?
- Kunde till exempel Twitter Bootstrap ha använts och därmed gjort min roll som designansvarig mindre omfattande?
- Hur undviks kompabilitetsproblem gällande stilmallar i olika webbläsare?

6.3.2 Metod

För att besvara de frågor som jag under arbetets gång samlat på mig så behövs något slags mätvärde på hur mycket möda som lagts på design relativt till slutkundens nöje av att systemet har ett modernt utseende som lätt kan bytas ut. Eftersom jag inte har kännedom om någon metod för att göra detta kommer jag dra egna slutsatser så att läsaren av min individuella del själv kan göra en uppskattning. Tidigare under utbildningen arbetade jag i ett liknande projekt där en webbapplikation utvecklades. I det projektet användes andra metoder för att slippa ha en man i utvecklingsteamet som arbetade på design. För att bedöma vad som blir bäst kommer erfarenheter från båda projekten att jämföras. Det material som användes i kursen TDDD60 *Interaktiva system* kommer att användas som referens då detta innehåller information som delvis kan besvara frågorna.

6.3.3 Resultat

Utseendet av en applikation har en otrolig inverkan på vana internetanvändare. Det är till och med så att många lämnar en hemsida eller webbapplikation om denna inte ser trovärdig ut eller inte fungerar som användaren tror att den gör (14).

Tid för design

De personer som kommer använda systemet är inte nödvändigtvis intresserade av om applikationen har rundade hörn, gradienter eller snygga menyövergångar i Javascript. Det var också den känslan som infann sig under projekt då varken kunden eller Vikbolandets ryttarförening nämnde något om systemets utseende. Med tanke på att runt 90% av min tid under projektet har lagts på att skapa estetiskt tilltalande komponenter kändes min arbetsinsats inte särskilt högt värderad. Systemets framgång med avseende på systemets utseende verkar endast ha en inverkan då det är en av nyckelfaktorerna som kunden värderar när det väljs mellan konkurrerande erbjudanden på marknaden (15).

Utifrån detta drar jag slutsatsen att det är viktigt att skilja på design och utseende. Utseende är nämligen bara en delmängd av design. Eftersom kunden måste kunna använda systemet med enkelhet, kommer layout vara en stor del av systemets framgång, till exempel hur lätt användaren hittar i menyer eller bokar en tid. Detta är av större vikt än grafiskt vackra komponenter. Det är som sagt mycket beroende på vad kunden sätter värde i och även om kunden inte sätter värde i ett estetiskt tilltalande system, kan det ändå vara det som får kunden att välja ens system och inte konkurrenternas.

Använda färdiga stilmallar

Frågan om användandet av färdiga stilmallar är värt mödan, relaterar direkt till föregående avsnitt. Det som avgör är vilket värde kunden sätter i att deras produkt ska skilja sig från mängden. Är det så att kunden vill ha något unikt så bör en i utvecklingsteamet jobba med design. I förra webbutvecklingsprojektet som jag deltog i användes en färdig stilmall, Twitter Bootstrap (5), som egentligen gör exakt det som gruppens färdiga system gör med teman. Det som är nackdelen är att de komponenter som ingår i Twitter Bootstrap är beroende av *jQuery* vilket frångick projektets vision angående användandet av Javascript. I detta projekt hade det gått lika bra att använda en färdig stilmall som t ex Twitter Bootstrap. Det går inte att undgå faktum att skapandet av egna stilkomponenter i CSS är en lärorik process och om lärande är huvudsakliga syftet så bör det tas i åtanke. Det hade också satt en till medlem i utvecklingsteamet på att utveckla back-end-kod som i detta fall var viktigt eftersom många funktioner önskades från kunden.

Internet Explorer

För att undvika kompabilitetsproblem för stilmallen i olika webbläsare, är den mest värdefulla metoden att hålla stilmallen enkel, det vill säga att försöka ha enkla hierarkier mellan element och undvika att använda webbläsarspecifik CSS (till exempel -webkit, -moz). Som designansvarig i ett webbutvecklingsprojekt bör riktiga verktyg för webbläsarkompabilitet införskaffas, så att alla webbläsare testas efter

ändringar i stilmallen. Det hände under projektets gång att stora ändringar i elementhierarkier och stilmallen behövdes eftersom stilmallen endast testades på en webbläsare under längre perioder. Det krävde så klart mycket mer tid än nödvändigt för till synes inget resultat alls.

6.3.4 Slutsats och diskussion

Med de erfarenheter som jag har samlat på mig under projektet kan jag säga att det förmodligen är bäst att utvärdera huruvide kunden värdesätter design (främst utseende) i ett projekt av denna magnitud, för att bedöma hur mycket tid som behover åsidosättas. Är utseendet av låg prioritet tycker jag att andra alternativ skall övervägas som t ex Twitter Bootstrap, speciellt om utvecklingsteamet inte har någon kunskap sedan tidigare. I detta projekt har jag blivit mest frustrerad över min egen insats då det känns som om jag har underpresterat, jag har ibland blundat för processen och fastnat i min egen värld fylld av CSS och stilmallsångest. Prototypning är ett underskattat verktyg och jag tror med att mer tid lagd åt prototypen så hade jag haft mer direktiv gällande utseendet och därmed inte behövt lägga lika mycket tid (för att bedöma om något ser bra ut eller ej) och sedan kunnat ägna mer tid åt back-end-kod.

6.4 Claes Lööf

Som testansvarig har det varit mitt ansvar att ta fram en testplan, testspecifikation och testrapport. Jag har även sett till att alla genomförda tester dokumenterats väl i testprotokollet och att testspecifikationen uppdaterats allteftersom.

Testplanen jag tog fram följde inte någon standard, utan jag anpassade den till projektet och tog med de rubriker jag ansåg vara väsentliga för testarbetet.

6.4.1 Frågeställning

I mitt arbete som testledare har följande frågor uppkommit:

- Hur noggrant måste testplanen följas?
- Måste alla testfall specificeras vid framtagningen av testplanen eller kan de uppkomma löpande under projektets gång?

6.4.2 Metod

För att få fram testdokument som var bra anpassade till utvecklingen av ett bokningssystem läste jag om vilka olika testmetoder som fanns och hur dessa implementerades. Jag övervägde vilka för- och nackdelar som fanns och använde sunt förnuft vid framtagningen av testdokumenten.

Under projektets gång har många revideringar av testdokumenten gjorts. Ju mer testningar man genomför desto bättre koll får man över vad som är väsentligt i testdokumenten, vilka tester som är nödvändiga och vilka tester som inte behövs.

6.4.3 Diskussion

Testarbetet har en avgörande roll i systemets utveckling och säkerhet. Med strukturerat och väl planerat testarbete kan man undvika många buggar och säkerhetshål.

Hur noggrant måste testplanen följas?

Eftersom jag aldrig varit testledare innan hade jag ingen aning om vad jag gav mig in på när jag skrev testplanen. För min del blev framtagningen av testplanen ett sätt att sätta mig in i vad testarbetet egentligen går ut på. Jag blev tvungen att läsa på vilka olika tester som det finns och vilka metoder som används för de olika testerna.

Jag har under testarbetets gång inte använt mig av testplanen. Således har det inte varit viktigt att följa den. Anledningen till att jag inte använt mig av testplanen är brist på tid. Jag ansåg det inte vara viktigt att spendera 50 timmar på att ta fram ett dokument. För mig var det betydligt viktigare att spendera timmarna på att utföra testerna istället och utveckla bokningssystemet.

Däremot tror jag att det är viktigt att ha en bra testplan i större och mer omfattande projekt. Något jag märkte under det här projektets gång var att testningen väldigt lätt blir stökig och ostrukturerad. Med mer tid och i ett större projekt hade jag lagt betydligt mer tid på att tidigt ta fram en bra testplan och tillhörande testspecifikation.

Måste alla testfall specificeras vid framtagningen av testplanen eller kan de uppkomma löpande under projektets gång?

Under iteration 1 hade projektgruppen ingenting att testa och jag visste inte hur jag skulle strukturera upp de olika testfallen som behövdes. Av den anledningen var testspecifikationen inte fullständig förrän iteration 2. För att ta fram testfall gick jag igenom kravspecifikation och se till att varje krav skulle testas. Ytterligare testfall uppkom löpande under systemets utveckling, främst säkerhetstest.

I ett större och mer omfattande projekt hade jag nog tagit fram en testspecifikation så tidigt som möjligt. Jag tror att ju bättre planerade och strukturerade testdokumenten är desto bättre blir testningen.

6.4.4 Slutsats

Oavsett projektets storlek är testdokumenten en väldigt viktig del av testarbetet. Med välplanerade och strukturerade testdokument effektiviseras testarbetet och färre timmar går till spillo. Testplanen är inte tänkt att följas bokstavligt, den är mer tänkt som en riktlinje.

Det är väldigt viktigt att ha en fullständig testspecifikation för att säkerställa att alla krav uppfylls, men testspecifikationen kommer väldigt sällan förbli oförändrad. Det kan dyka upp nya krav, nya funktioner och dylikt under projektens gång.

6.5 Behnam Sani

Som arkitekt har jag jobbat med att ta fram ett arkitekturdokument för det system som utvecklats. Arkitekturdokument skulle vara det utvecklarna utgick från när systemet utvecklades. Det innebar att mycket tid gick åt att försöka strukturera upp och dokumentera innan utvecklingen skulle börja.

6.5.1 Frågeställningar

- Hur viktigt är arkitekturdokumentet för mindre projekt?
- Gör arkitekturdokumentet någon nytta för mindre projekt?

6.5.2 Metod

Data som används kommer huvudsakligen från egna erfarenheter. Jag har även frågat andra gruppmedlemmar om hur de har arbetat under projektets gång.

Under förstudien och iteration 1 gick det åt mycket arbete för att planera och dokumentera. Innan utvecklingen av systemet skulle börja behövde det vara klart vad det var som skulle utvecklas. Mycket tid gick åt att förstå kundens behov och att formulera dessa som krav för systemet. Att ha kravspecifikationen färdig var bara början.

Från kravspecifikationen skulle ett arkitekturdokument skrivas. Arkitekturdokumentet skulle beskriva det system som utvecklades. De krav som fanns skulle ligga till grund för hela systemet. Att arkitekturdokumentet skulle vara det dokument som beskrev hur systemet skulle se ut, gjorde det till ett viktigt dokument att tänka genom noggrant innan utvecklingen började.

Innan systemet skulle formgivas togs några av de större besluten av hela gruppen. Val av plattform, programmeringsspråk och ramverk gjordes av hela gruppen. Andreas fick till uppgift att ta fram några alternativ tillsammans med en beskrivning av för- och nackdelar för respektive alternativ.

Jag har varit ansvarig för att ta fram ett arkitekturdokument. Till största del var det jag som skrev dokumentet, men jag fick också hjälp av andra gruppmedlemmar. Databasdesign, som hängde ihop med arkitekturen, gjordes i samband med arkitekturdokumentet. Jag och Filip gjorde varsin modell av databasen och satte dessa samman genom att ta de bästa delarna från båda. Det kontrollerades även så att de krav som fanns kunde uppfyllas med denna databasmodell.

6.5.3 Resultat

Arkitekturdokumentet var ett dokument som aldrig blev färdigt. Det blev ett dokument där endast tidiga idéer skrevs ner. Jag kunde inte tillräckligt mycket om ramverket Yii för att veta hur systemet borde ha sett ut för att utnyttja ramverket fullt ut. Det var inte heller någon annan i gruppen som hade någon kunskap om hur ramverket fungerade. Arkitekturdokumentet blev senare ett dokument som ingen tittade på, än mindre att utgå från vid utveckling av systemet.

Databasdesignen som gjordes blev viktig för utvecklingen. Det var den designen som alla utgick från. Systemets komponenter växte fram från de tabeller som fanns i designen. Den design som först gjordes var däremot inte därför den slutgiltiga; det gjordes omkring 20 förändringar av databasen. Dessa 20 ändringar var både mindre ändringar, och mer omfattande.

6.5.4 Diskussion

Detta har varit ett projekt med åtta gruppmedlemmar. Antalet gjorde det möjligt för alla att vara delaktiga i det mesta. Alla var varit medvetna om vad det var som skulle utvecklas och vilka krav som fanns för systemet. Gruppen var på ungefär samma nivå vad gäller kunskapen av ramverket Yii. Det har även varit något nytt för gruppen, något som ingen tidigare gjort. Det har gjort att ingen kunde veta om någon speciell utformning av systemet skulle fungera bra eller inte i förhand.

Genom att lära sig om ramverket genom att programmera har även systemet växt fram. Genom att utgå från databasmodellen så har funktionalitet kontinuerligt lagts på och förbättrats. Systemet strukturerades bättre allt eftersom gruppen lär sig att programmera i ramverket.

Det som har varit viktigt var att reda ut olika begrepp och hur dessa skulle hänga samman. Exempel på begrepp var lokal, aktivitet och kort. Dessa förklarades delvis med databasdesignen där det framgick vad det var för något och hur de kopplades till varandra.

Om arkitekturdokumentet hade skrivits färdig och gruppen följt dokumentet, skulle det vara en stor risk för att utvecklingen skulle ha gått åt fel håll. Det hade varit en stor risk eftersom gruppen hade försökt utveckla tvärtemot ramverkets principer. Det skulle bara ha orsakat problem och ramverket skulle inte ha gjort någon nytta alls, utan skulle endast ha varit i vägen.

Arkitekturdokument ska bland annat användas till att utvärdera arkitekturen och alternativa arkitekturer. Det ska även vara ett stöd för att planera aktiviteter (16). För vår del har det inte alls varit till någon nytta, varken för att utvärdera arkitekturen eller att planera aktiviteter. Att jobba med arkitekturdokumentet har bara kostat timmar som inte har gett något.

6.5.5 Slutsats

För projekt med få gruppmedlemmar har arkitekturdokumentet inte varit viktig alls för utvecklingen av systemet. Ingen har tittat på eller utgått från arkitekturdokumentet vid utvecklingen. Därmed har arkitekturdokumentet inte heller varit till någon nytta för projektet. Det har bara kostat tid.

Det som har varit till nytta var databasdesignen. Databasdesignen har varit det som gruppen utgick från när systemet utvecklades. Tillsammans med databasdesignen definierades även viktiga begrepp inom systemet. Det har gett den information som behövdes för att utvecklingen skulle starta.

6.6 Att lära sig ett jobb: Peters erfarenheter

Personligen saknade jag erfarenhet av både PHP i allmänhet och Yii i synnerlighet när jag började med projektet. Mina kunskaper i SQL var mycket begränsade. Situationen kan tänkas vara jämförbar med att börja på en ny arbetsplats där det finns etablerade rutiner, verktyg och kodbas. Vet man om att det är omöjligt att utveckla perfekt mjukvara (17) så är det enkelt att börja koda. Det första steget var att börja med enkla övningar på Code academy för att få en uppfattning om PHP. Språket har en syntax som är inspirerad av Perl (18) som i sin tur är bygger på C (19). Då jag tidigare har erfarenhet av C har det därför inte varit något problem att lära sig PHP. Det största problemet med PHP har nog varit att det är löst typat och en del enkla onödiga misstag letar sig in i koden.

Efter att vi i gruppen hade lärt oss grunderna började vi med utvecklingsfasen i mitten av mars. Vårt första mål var att skapa ett skelettsystem med funktionalitet för att lagra bokningar och för användarhantering. Jag tillsammans med två andra började med bokningsdelen. I det här skedet gick mycket tid åt till att läsa Yiis dokumentation och klassreferenser för att förstå hur vi kunde använda ramverket för att läsa från och skriva till databasen. Värdet av den kod som skapades i det här skedet måste i efterhand sägas ha varit ringa, men arbetet var ändå nödvändigt, och av värde då det gav oss något konkret att diskutera (20) och gav en bättre förståelse av vilken typ av problematik vi faktiskt skulle bli tvungna att lösa. Även om det inte var ett medvetet val att den tidiga koden skulle skrivas för att bytas ut, lät det oss låta systemet gradvis växa upp med mer och mer komplexa kringfunktioner.

Följer man utvecklingsflödet för bokningar, kunde man först enbart boka på hårdkodade tider i hårdkodade lokaler. Genom att lära sig hur man skrev till databasen kunde detta lösas. Nästa steg var att koppla bokningarna till lokalerna och få bokningarna att respektera lokalernas begränsningar, vilket kunde lösas när vi kunde läsa från databasen. Nästa steg var att implementera aktiviteter vilket krävde förståelse för hur referenser fungerar i SQL.

Innan arbetet hade startat, trodde jag personligen att jag, när jag kunde SQL-referenser, att läsa, att skriva och relationer, skulle vara i princip fullärd och därefter kunna fokusera på att skapa vårt slutgiltiga system. Det visade sig dock att jag underskattat ramverket och de möjligheter det skulle ge mig att skapa flexibla och kraftfulla funktioner enkelt och snabbt. När jag efter bokningarna gick vidare till att implementera stöd för fakturering märkte jag att jag fick bättre förståelse för Yiis dokumentation och kunde dra nytta av ramverket vilket gjorde att jag framförallt skapade mer robust kod. I ramverket fanns verktyg som, när man förstod dem, gav färdiga lösningar för många av de problem vi löst för hand tidigare. Mycket av den SQL-kod som vi använde för att skapa ett dynamiskt utseende kunde vi nu låta ramverket generera utifrån modellerna. Det gjorde att vi fick bra separation på hur databasen var skriven och på hur data hanteras i vyer, så att när vi blev tvungna att ändra i databasen så räckte det med att ändra i modellfilerna.

Jag gick in i en period då jag inte var speciellt produktiv utan lärde mig ramverket och tiden gick åt till att förbättra och arbeta om tidigare lösningar. Tid gick åt till att arbeta om modelldelen av systemet så att den bättre svarade mot vår mentala modell av hur systemet hängde ihop. Det var först efter dessa två faser som jag faktiskt kunde börja producera kvalitativ kod i hög takt och gå över till att börja arbeta med att skapa ett användbart system. Det är först i det här skedet som jag har känt att jag har behärskat mitt arbete och inte varit en lärling.

Som utvecklingsledare har jag fokuserat på att se till att veta vad team-medlemmarna arbetar med och vilka funktioner de lär sig för att kunna föreslå vem som ska ta på sig nya uppdrag. Det har fungerat

för dem som valt att närvara vid kodsstugorna då den dagliga kontakten gjort att jag kunnat hålla reda på det utan några formella rutiner. För en större grupp eller en mer utspridd grupp skulle ett mer strukturerat arbetssätt ha behövts. Det som var svårast var att få en uppfattning om vad den medlem som mestadels arbetade hemifrån bidrog med och involvera honom i utvecklingsprocessen. I och med att resten av gruppen arbetade så pass bra såg jag dock inte någon vinst att försöka lösa det, utan jag accepterade de förlorade timmarna mot att resten av gruppen kunde arbeta fritt. Dagliga Scrum-möten kunde tänkas vara en bra kompromiss mellan frihet och struktur (21) och i efterhand tror jag nog det skulle ha hjälpt gruppen till ett bättre resultat men att införa det endast för att en enskild medlem arbetar på annat håll vore en risk som inte var motiverad.

Som utvecklare har rollen som utvecklingsledare stört vid gemensamt arbete. Frågor dyker ständigt upp och det är svårt att fokusera på det egna arbetet vilket gjort att jag i slutet av projektet har tillbringat mindre tid på kodstugorna. Den tiden har jag valt att fokusera på informationsspridning, att diskutera och att förankra tekniska lösningar så att alla drar åt samma håll, och på att diskutera vilka funktioner som ska prioriteras och utvecklas näst. Sedan har jag arbetat själv hemma för att kunna arbeta ostört och producera de delar jag tagit på mig. Tänker vi oss en framtida arbetsplats så kommer jag behöva ett eget kontor där jag kan arbeta ostört för att kunna vara produktiv vilket stöds i forskning gällande programmering och produktivitet (22).

6.7 Filip Strömbäck

Som teamleader har mina huvudsakliga uppgifter bestått av att se till så att arbetet i gruppen fungerat väl genom att samordna det som ska göras och delegera uppgifter till ansvariga för respektive område. Detta arbete har gett mig många insikter i vikten av kommunikation i en grupp, samt den problematik som kan uppstå i och med att storleken på arbetsgruppen växer.

6.7.1 Frågeställning

I mitt arbete som teamleader har följande frågor uppkommit:

- Vad har fungerat bra med den metod vi har använt? Vad har fungerat mindre bra?
- Behöver något förändras i metoden för att den ska fungera även i större projektgrupper?

6.7.2 Bakgrund

Projektgruppen hade fått till uppgift att utveckla ett generellt bokningssystem, som efter mindre anpassningar ska kunna användas hos vår referenskund, Vikbolandets ryttarförening. Vår kund hade nära kontakt med Vikbolandets ryttarförening och kunde tidigt tillhandahålla det gamla bokningssystem som användes av Vikbolandets ryttarförening.

Trots att projektgruppen fick fri tillgång till det existerande systemet var det fortfarande inte helt klart exakt vilken funktionalitet som kunden önskade och exakt hur kunden ville hantera de olika koncept som fanns i det gamla systemet. Det fanns dessutom väldigt mycket funktionalitet i det gamla systemet som inte användes längre, vilket också var svårt att inse utan god kommunikation med kunden. Denna osäkerhet tillsammans med vetskapen om vikten av att det producerade systemet passar in väl i den existerande modellen för att hantera kunder och bokningar gjorde att gruppen snabbt insåg vikten av god kommunikation med både kunden och referenskunden.

6.7.3 Metod

Från projektets start enades gruppen om att arbeta enligt en modell inspirerad av Scrum. Anledningen till att en agil modell valdes är att vi tidigt insåg att bokningssystemet innehåller mycket funktionalitet som är hårt knuten till hur kunden hanterar saker internt. Att få en komplett och korrekt bild av hela kundens process är problematiskt, eftersom det är enkelt att ta små detaljer för givet. Därför ansåg

gruppen att det skulle vara fördelaktigt att med jämna mellanrum demonstrera gruppens tolkning för kunden, så att kunden fick möjlighet att rätta eventuella feltolkningar i systemet innan leverans.

Metoden som användes var i grund och botten Scrum, men de dagliga Scrum-mötena ersattes med lite större veckovis möten med liknande innehåll. Anledningen till denna modifikation var att gruppen inte arbetade heltid med projektet och att de dagliga mötena därför också borde vara något glesare. Ett möte i veckan valdes för att det var nästa större intervall.

För att hantera kvarvarande arbete valde gruppen att använda ärendesystemet i utvecklingsplattformen Trac. Gruppen lade vid ett möte inför varje iteration upp ärenden på Trac, vilka beskrev vad som behövde göras under kommande iteration, för att enkelt kunna hålla reda på vad som behöver göras. Mellan iterationerna gicks också kravspecifikationen igenom för att säkerställa att inga krav hade glömts bort.

6.7.4 Diskussion

Under utvecklingsarbetet satt huvudsakligen projektgruppen tillsammans i ett rum, inte för att nödvändigtvis arbeta på samma upggifter, utan främst för att underlätta kommunikationen mellan gruppmedlemmar. Detta ledde i sin tur till att beskrivningarna i de ärenden som lades upp kunde hållas relativt korta. I stället var det enkelt att fråga någon annan om eventuella oklarheter. Att anordna kodstugor var till en början inte ett aktivt beslut, utan något som växte fram utifrån behovet av bättre kommunikation (23 s. 605).

Detta var positivt för gruppen eftersom det var en snabb och enkel process att lägga till en brist i systemet som ett ärende. Hade gruppen i stället haft högre krav på att alla ärenden skulle vara utförligt beskrivna, så hade processen som krävdes för att lägga upp ett ärende varit mer utdragen för utvecklaren. Det hade antagligen inneburit att färre brister rapporterades formellt, eftersom många brister i stället bara hade skrivits upp någonstans och sedan glömts bort. Den informella och enkla beskrivningen uppmuntrar alltså att lägga upp alla småsaker som inses på det centrala ärendesystemet. Det har den stora fördelen att det blir mycket mindre risk att någonting glöms bort, men det blir då också möjligt att andra utvecklare kan fortsätta på de små lösa trådarna som annars hade skjutits upp av utvecklaren som inte hade orkat rapportera dem. Ytterligare fördelar med att ha kunnat sitta tillsammans och utveckla systemet var att det var väldigt enkelt att få en uppfattning om vad alla andra i gruppen gjorde för tillfället och enkelt kunde fråga om oklarheter i andra utvecklares lösningar.

En nackdel med detta system är att det i efterhand är svårt att följa upp omfattning och tidsåtgång för specifika ärenden. Detta märktes tydligt vid planeringsmötena inför iterationerna. Vid dessa möten försökte gruppen ta fram en preliminär tidsplan inför den kommande iterationen, och det är vid dessa tillfällen det hade varit värdefullt att ha information om hur pass korrekta tidsuppskattningar för de ärenden som behandlades förra iterationen varit.

En annan nackdel med konstanta kodstugor är att de snabbt kommer att falla samman då projektgruppen blir större. För att metoden ska fungera var det viktigt att projektgruppen kunde sitta tillsammans större delen av tiden, eftersom medlemmar annars inte kunde fråga varandra om oklarheter i ärendebeskrivningarna.

Ett ytterligare problem är att de diskussioner som uppstår lätt stör de gruppmedlemmar som inte berörs (23 s. 606). Detta är också ett problem som blir större med växande gruppstorlek, eftersom fler diskussioner då kommer att uppkomma. Detta skulle exempelvis kunna lösas genom att gruppen inte alltid sitter samlad. Detta minskar dock fördelarna med den avslappnade och öppna diskussion som uppmuntrats genom att alltid vara samlade och metoden börjar då allt mer likna Scrum igen.

Vi kan alltså här se orsaken till de dagliga möten som Scrum uppmuntrar till. De dagliga mötena där är en bra kompromiss mellan att alltid sitta tillsammans och utveckla produkten och att samtidigt inte alltid bli störd av alla andras frågor och diskussioner. Scrums dagliga möten tillåter också större grupper, eftersom dessa möten ska vara korta. Möteslokalen behöver därför inte vara så stor så att den rymmer alla utvecklares arbetsplatser, som i den metod som projektgruppen utnyttjade.

Även Scrum har insett problematiken med för stora grupper av utvecklare och har därför myntat begreppet ”Scrum of Scrums” (24), vilket innebär att hela projektet delas in i mindre grupper där varje grupp använder Scrum-modellen. Representanter från varje mindre grupp möts sedan regelbundet för att kunna bilda sig en uppfattning om hela projektets status (25 s. 5-11).

6.7.5 Slutsats

Den metod som användes fungerade i gruppens fall väldigt bra, eftersom den tillät enkel kommunikation när som helst. Därmed behöver inte hanteringen av exempelvis ärenden vara lika strikt och formell, vilket sparar tid i projektet. Däremot kommer inte denna metod att fungera lika väl när antalet medlemmar i projektet ökar. Då blir det allt viktigare att ha väl genomtänkta och väl använda metoder för att kommunikationen mellan projektmedlemmarna ska fungera väl.

6.8 Per Wennberg

Som analys- och kundansvarig har jag under projektet haft mycket kontakt med i princip alla intressenter för projektet; beställare, kund och slutanvändare. Den huvudsakliga uppgiften för mig har varit att genom kontakt med beställare och kund försöka få deras önskningar att speglas i arbetet och kravspecifikationen. Utöver beställare och kund har även personer som tidigare använt och varit insatta i det gamla bokningssystemet varit involverade och haft en del att säga till om under arbetets gång.

6.8.1 Frågeställning

Jag ska i denna del av rapporten försöka besvara följande frågor:

- Hur fungerar det att ha flera olika intressenter involverade vid utvecklandet av mjukvara?
- Hur kan dessa intressenters tekniska kunnande och erfarenheter spela in?

6.8.2 Metod

För att besvara mina frågor kommer jag mestadels att tala om mina erfarenheter i rollen som kund- och analysansvarig i projektet. De erfarenheter jag tycker känns relevanta för frågeställningarna är den indirekta kontakt som sker mellan utvecklare och övriga intressenter. Där gruppen å ena sidan har möten där man diskuterar krav, går kund- och analysansvarig å andra sidan vidare med detta som underlag till kund/beställare.

Mina frågeställningar har haft relevans under hela projektets gång, ända från första början med skrivandet av kravspecifikation till slutdemonstration där intressenterna fått sista ordet angående slutprodukten. Därför kommer jag att tala om det mesta i processen med analysarbetet.

6.8.3 Erfarenheter

I den inledande fasen under projektet låg mycket av mitt och hela projektgruppens fokus på framtagning av en kravspecifikation. Detta arbete började med att gruppen på egen hand diskuterade om vad ett generellt bokningssystem faktiskt skulle innehålla, för att få ett slags första underlag till kravspecifikationen. Under ett möte använde vi en metod där vi brainstormade fram idéer angående detta, för att få ett brett spektrum med olika möjliga funktioner och krav. Utöver det som framtogs under detta möte hade vi också mer att gå på: det fanns ett gammalt system som det nya systemet skulle ersätta. Då vårt system skulle innehålla de funktioner som fanns med i detta gamla system fick vi tillgång till det. Gruppen kunde sedan gå igenom det gamla systemet för att se vilka funktioner som fanns med där. Detta var mycket till hjälp, då alla funktioner som fanns med i det gamla systemet också skulle finnas med i det nya.

Då vi var nöjda med det första utkastet av kravspecifikationen hade vi ett möte med beställaren där vi gick igenom den i dess helhet och fick feedback, både saker som skulle ändras och nya krav som borde läggas till. Det var inför dessa första möten mycket behjälpligt att vi kunde ha en tämligen utförlig kravspecifikation som underlag. Gruppens metoder med att brainstorma fram krav och det faktum att vi hade ett gammalt system som vi kunde titta på gav alltså stor utdelning. Att få ett bra första underlag är viktigt då det kan spara in både tid och pengar senare (26).

Beställaren av denna produkt hade ett stort tekniskt kunnande vilket resulterade i att kravspecifikationen granskades tämligen hårt. Beställaren påpekade direkt om denne tyckte att vi hade varit otydliga på något sätt. Om vi hade siktat lite för lågt på någon punkt fick detta ofta revideras. Beställaren hade också haft att göra med det gamla systemet och var väl insatt i det vilket också betydde att kravspecifikationen kanske behövde vara mer omfattande än den kanske annars hade behövt vara. Dessa fakta resulterade i att kravspecifikationen enligt gruppen tog längre tid att få godkänd än förväntat. Det innebar inga direkt avgörande konsekvenser för arbetet mer än att det var något ovisst under en tid.

Efter den första revideringen av kravspecifikationen och visning av denna för beställaren skickades den vidare till kunden, de som faktiskt kommer att använda systemet. Denna gång var det frågan om administratören till det gamla systemet och en användarrepresentant. En tid efter detta hade jag mitt första samtal med denna administratör, som fick komma med synpunkter. Samtalet var mycket givande då det gav ett slags användarperspektiv på det hela och resulterade i att åtskilliga krav samt någon revision tillfördes dokumentet. Administratören hade också administrerat det gamla systemet under en lång tid och ansågs därför vara en god källa för kunskap om bokningssystemet och vad användarna ville ha. Detta faktum medförde också att denna persons åsikter vägde tungt. Under hela processen med att ta fram kravspecifikationen kändes det som om gruppen och alla intressenter drog åt samma håll. Alla krav skrevs och definierades på en lagom nivå som alla inblandade kunde förstå, vilket också gjorde det enkelt att prata om dem.

Efter att vi någon revision senare hade fått kravspecifikationen godkänd hade arbetet redan kommit en bra bit på vägen. Det kunde nu också fortsätta på det spåret, då förhandlingar med kravspecifikation i regel gått bra bortsett från tidsåtgången. Här hade beställarens tekniska kunnande också spelat in på ett positivt sätt, då vi tidigt hade fått veta om något i kravspecifikationen inte hade stämt. Detta betydde att vi tidigt visste om vi kunde sätta igång att arbeta med delar av systemet, även om kravspecifikationen inte hade blivit godkänd ännu. Möten med beställaren hölls ungefär varannan vecka, vilket passade båda parter väldigt bra. Vi i gruppen kunde få mer gjort som kunde visas och gås igenom vid dessa möten. I iteration två gick projektet framåt väldigt snabbt och både beställaren och kunden var väldigt nöjda. I detta skede låg fokus mest på funktionalitet i systemet. Under denna tid hade jag viss kontakt med administratören för det gamla systemet. Samtalen handlade mest om detaljer för vissa viktiga funktioner. Några exempel på sådana funktioner är hur faktureringen i systemet skulle fungera och hur kort, som säsongskort, hanterades i systemet. Dessa samtal utmynnade ofta i bra saker där jag och gruppen fick reda på hur kunden faktiskt ville att systemet skulle arbeta. Kunden hade sina sätt och rutiner att arbeta med till exempel fakturor och de ville i regel att det nya systemet skulle fungera på samma sätt fast bättre. För det mesta var deras önskningar också vettiga vilket gjorde att gruppen gärna tillgodosåg dem.

6.8.4 Slutsats

I detta projekt har det fungerat bra att ha flera olika intressenter inblandade. Det har medfört väldigt få, om ens några, konflikter. Kontakten mellan alla parter har under hela projektets gång fungerat väldigt smidigt. Har någon velat kontakta någon eller hålla möten har detta kunnat tillgodoses snabbt. Om någon fråga har uppstått angående till exempel någon funktion i systemet har denna kunnat besvaras bra och i tid. Intressenternas tekniska kunnande och erfarenheter har egentligen bara tillfört positiva aspekter. Trots att vi skapat ett generellt bokningssystem, är ju kundens bokningssystem specifikt. De har haft ett bokningssystem i drift länge och de vet vad de vill ha för något. Detta talar mot det som ofta sägs (25 s. 146) om att intressenter sällan vet vad de faktiskt vill ha. Det har alltså varit personer inblandade som velat vara med och arbeta för att ta fram ett nytt och bra bokningssystem.

7 Resultat

Gruppen lyckades under projektets gång producera ett generellt bokningssystem som tillåter användare att boka aktiviteter i lokaler via Internet. Utöver att låta användare boka tider, tillhandahåller systemet också underlag för fakturering, så att administratörer till systemet enkelt kan generera den information som behövs för faktureringen.

För att kunna generera underlag för faktureringen behöver systemet information om den prismodell som används. Att representera alla möjliga prismodeller på ett sätt som ger full frihet resulterar antagligen i att användaren måste implementera sin egen logik. I stället valde vi konceptet med kort som beskrivs i kapitel 3.5, vilket ger tillräcklig flexibilitet för vår referenskund utan att bli svårt för användare att förstå.

Vår referenskund har även ett antal sektioner som ska kunna boka lokaler för speciella aktiviteter. I detta fall är det inte praktiskt att låta sektionen ha ett gemensamt användarkonto för att lägga bokningar i sektionens namn. I stället används här virtuella användare, som beskrivs i detalj i kapitel 3.4.2. Då skapas en virtuell användare som representerar sektionen i stället. Fördelen med en virtuell användare är att denna virtuella användare kan länkas till sektionsmedlemmar som har till uppgift att göra bokningar som sektionen. Då kan alla ansvariga klubbmedlemmar enkelt boka i sektionens namn från sina vanliga användarkonton, och bokningssystemet kan inse att det är sektionen som ska faktureras.

8 Diskussion

Det bokningssystem som gruppen har implementerat lämpar sig väl för de bokningalternativ som finns på Vikbolandets ryttarförening. Systemet är även så generellt att det kan representera andra modeller, dock inte en godtycklig bokningsmodell.

Om bokningssystemet i stället skulle anpassas för att passa en frisörsalong, kan varje enskild frisör modelleras som en lokal i bokningssystemet. De olika typerna av behandling hos frisörer kan också representeras med en aktivitet per behandling per frisör. Eventuella rabatter som frisörsalongen har kan då representeras med olika kort. Om frisörsalongen har ett erbjudande på formen *var 5:e behandling på köpet* går detta för närvarande inte att automatiskt hantera i systemet. Däremot kan ett kort för *gratis behandling* skapas och manuellt ges till kunden då kunden ska få sin gratisbehandling.

Just i fallet med frisörsalongen kommer problemet med att alla behandlingar inte tar lika lång tid att finnas. Eftersom systemet för närvarande har förutbestämda tidsgränser för varje lokal går det inte att ange att exempelvis klippning tar en timme, medan exempelvis färgning tar tre timmar. Detta går dock att lösa genom att i stället modellera de lediga platserna i frisörsalongen som en lokal och exempelvis reservera två av platserna för mer tidkrävande behandlingar. Att annars ändra implementationen så att tids- och längdbegränsningar på bokningar kan anges per aktivitet är inget problem att göra.

I allmänhet märks inte den underliggande konfigurerbarheten för slutanvändaren av systemet, eftersom användaren bara ser bokningsbara lokaler (som i exemplet för en frisörsalong skulle vara frisörer) och bokningsbara aktiviteter. Att systemet är generellt märks dock mer för dem som administrerar systemet, främst för den som sätter upp systemet från början. Detta beror på att administratören måste skapa alla aktiviteter separat för varje lokal. Antag exempelvis att frisörsalongen har fem frisörer och varje frisör kan erbjuda sju behandlingar. Då måste administratören lägga till 35 aktiviteter, vilket inte kan göras snabbt i nuläget. Samma problematik finns också för vår referenskund, Vikbolandets ryttarförening. De har fyra olika lokaler, med ett antal aktiviteter i varje. Många av dessa aktiviteter finns i alla fyra lokaler, men aktiviteterna måste ändå läggas till i varje lokal manuellt. Detsamma gäller ifall ytterligare lokaler tillkommer.

Detta är ett mindre problem, i alla fall med ridklubbar, eftersom de lokaler som finns och de bokningsbara aktiviteterna per lokal inte ändras särskilt ofta. Skulle systemet däremot användas på en plats där det som lokalerna representerar är mer dynamiskt blir detta snabbt ett större problem.

Problemet med den repetitiva processen som krävs för att sätta upp systemet är att administratören riskerar att bli uttråkad och därmed riskerar att göra fel. Detta är ett problem som skulle kunna arbetas vidare på, men på grund av den begränsade tidsramen har detta inte prioriterats.

9 Slutsats

I slutändan lyckades projektgruppen skapa ett väl fungerande system, som kan anpassas för att fungera väl för Vikbolandets ryttarförening. Systemet skulle också kunna användas för andra ridskolor och för andra typer av föreningar. Eftersom olika föreningars modeller för fakturering skiljer sig mycket åt, kommer systemet i vissa fall behöva utökas utöver den modell som används för närvarande.

Komplexiteten i systemet har projektgruppen också lyckats undanhålla från slutanvändaren. Däremot exponeras komplexiteten mycket mer mot administratörer och gruppen inser redan nu att vissa delar av systemet nästan är för komplicerade för att kunna användas effektivt i verkliga fall.

Referenser

1. Tekniska högskolan vid Linköpings Universitet. TDDD77 Kandidatprojekt i programvaruutveckling. 2013-09-20 [Hämtad 2014-05-11]. Tillgänglig online: `http://kdb-5.liu.se/liu/lith/studiehandboken/svkursplan.lasso?&k_budget_year=2014&k_kurskod=TDDD77`
2. TDDD60 > Hemsida. Mattias Arvola. 2014-03-10 [Hämtad 2014-05-11]. Tillgänglig online: `http://www.ida.liu.se/~TDDD60/index.sv.shtml`
3. Fundamentals: Model-View-Controller (MVC). Datum saknas [Hämtad 2014-05-11]. Tillgänglig online: `http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc`
4. The PHP Group. PHP: password_hash. Datum saknas [Hämtad 2014-05-12]. Tillgänglig online: `http://docs.php.net/manual/en/function.password-hash.php`
5. Twitter. Bootstrap. Datum saknas [Hämtad 2014-05-12]. Tillgänglig online: `http://getbootstrap.com/2.3.2/`
6. Gecko. Mozilla. Datum saknas [Hämtad 2014-05-08]. Tillgänglig online: `https://developer.mozilla.org/en-US/docs/Mozilla/Gecko`
7. Blink. The Chromium Project. Datum saknas [Hämtad 2014-05-08]. Tillgänglig online: `http://www.chromium.org/blink`
8. CSS introduction. W3Schools. Datum saknas [Hämtad 2014-05-08]. Tillgänglig online: `http://www.w3schools.com/css/css_intro.asp`
9. Thomas A. Powell. HTML & CSS: The Complete Reference- Fifth Edition. McGraw-Hill/Osborne. 2010.
10. Utan namn. Yiiframework.com. 2011-06-20 [Hämtad 2014-05-08]. Tillgänglig online: `http://www.yiiframework.com/forum/index.php/topic/20720-cross-browser/page__view__findpost__p__101484`
11. Svensson, Tomas; Krysander, Christian. Projektmodellen Lips, upplaga 1:1. Lund: Studentlitteratur AB. 2011. ISBN: 978-9-144-07525-9.
12. Chikh, Azeddnie; Aldayel, Mashael. New Prespectives in Information Systems and Technologies, Volume 1. Springer International Publishing: 2014. ISBN: 978-3-319-05950-1.
13. Google. Limits on sharing. Datum saknas [Hämtad 2014-05-08]. Tillgänglig online: `https://support.google.com/drive/answer/2494827?hl=en`
14. Krug S. Don't Make Me Think: A Common Sense Approach to Web Usability, 3rd edition. Berkeley: New Riders Publishing, 2014.
15. Hartog, Niklas; Lundqvist Anders. Var finns Kundvärdet? En fallstudie om hur B2B-företag hanterar kundvärde i sin verksamhet. Luleå: Luleå tekniska universitet, 2008. ISSN: 1402-1773. Tillgänglig online: `http://epubl.ltu.se/1402-1773/2008/299/LTU-CUPP-08299-SE.pdf`
16. Institute of Electrical and Electronics Engineers, Inc. ISO/IEC/IEEE 42010:2011, Systems and software engineering. Switzerland: ISO copyright office, 2011.
17. Hunt, Andrew; Thomas, David. The Pragmatic Programmer, From Journeyman To Master. Addison-Wesley Professional, 1999.
18. The PHP Group. History of PHP. PHP: Hypertext Preprocessor. Datum saknas [Hämtad: 2014-05-08]. Tillgänglig online: `http://se2.php.net/manual/en/history.php.php`
19. Ashton, Elaine. The Timeline of Perl and its Culture. `http://history.perl.org/`. 1999 [Hämtad: 2014-05-08]. Tillgänglig online: `http://history.perl.org/PerlTimeline.html`
20. Warfel, Todd Zaki. Prototyping: A Practitioner's Guide. Brooklyn, N.Y.: Rosenfeld Media, 2009. ISBN: 978-1-933-82021-7.

21. Sutherland, Ken Schwaber & Jeff. Scrum. Scrum Guide™. juli 2013. [Hämtad 2014-05-08]. Tillgänglig online: https://www.scrum.org/Scrum-Guide

22. Paris, Jon; Gantner, Susan. Looking For Ways to Reduce Your Programmer's Productivity? IBM-Systems MAGAZINE. 2008-09-09 [Hämtad: 2014-05-08]. Tillgänglig online: http://ibmsystemsmag.blogs.com/idevelop/2008/09/looking-for-way.html

23. Sommerville, Ian. Software Engineering. Harlow: Pearson Education Limited. 2007. ISBN: 978-0-321-31379-9.

24. Agile Alliance. Guide to Agile Practices: Scrum Of Scrums. Datum saknas [Hämtad: 2014-05-04]. Tillgänglig online: http://guide.agilealliance.org/guide/scrumofscrums.html

25. Sutherland, Jeff. Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies. Cutter IT journal. 2001 dec;14(12). Tillgänglig online: http://www.controlchaos.com/storage/scrum-articles/Sutherland%20200111%20proof.pdf

26. Business Requirements Analysis. Datum saknas [Hämtad: 2014-05-11]. Tillgänglig online: http://www.mindtools.com/pages/article/newPPM_77.htm

Bilagor

Databasdiagram