

Schemaläggning av personal i publik verksamhet

Tilde Crew:

Lovisa Dahl

Emma Forsling Parborg

Martin Gråd

Linnea Malcherek

Julia Nilsson

Linnéa Nåbo

Linköpings Universitet

2 juni 2014

Sammanfattning

Denna rapport är en del av ett arbete för kandidatexamen i Civilingenjör i medieteknik vid Linköpings Universitet (kurs: TNM094). I rapporten återges utvecklingsprocessen för ett systemutvecklingsprojekt som genomförts under våren 2014. Systemet som utvecklats i detta projekt är en webbaserad schemaapplikation för den publika verksamheten på Visualiseringscenter C. Kortfattat är målen med projektet att skapa ett system för att göra ett schema som både är lättillgängligt och redigerbart.

De viktigaste frågeställningar som undersökts i utvecklingsarbetet är hur schemavisningen kan underlättas med hjälp av visualisering och filtrering och hur funktionalitet för att schemalägga personal kan göras enkelt via en webbapplikation. Utöver utredning av frågeställningarna inkluderar rapporten också redogörelse för den valda agila metoden *scrum* och hur systemutvecklingsprocessen sett ut. Klassiska delar som systemarkitektur, programdesign, testning och dokumentation är inkluderade och diskuterade. Även verktyg och rutiner som projektgruppen använt sig av diskuteras och värderas. Resultatet av arbetet är ett lättillgängligt system i form av en webbapplikation. I denna kan personal logga in och se ett schema och administratör har därutöver möjlighet att lägga till och ta bort personal på bokningar.

INNEHÅLL

1	Inledning	1
1.1	Bakgrund	1
1.2	Syfte och frågeställning	2
1.3	Avgränsningar	2
2	Mål och prioriteringar	3
2.1	Mål för produkten	3
2.1.1	Funktionella krav	3
2.1.2	Icke-funktionella krav	4
2.1.2.1	Kapacitetskrav	4
2.1.2.2	Kvalitetskrav	4
2.1.2.3	Säkerhetskrav	4
2.1.2.4	Designkrav	5
2.2	Prioriteringar och avgränsningar	5
3	Utvecklingsprocessen	6
3.1	Agil utvecklingsmetod: Scrum	6
3.2	Rutiner	7
3.2.1	Dokumentation	7
3.2.2	Testning och kodgranskning	7
3.3	Projekthantering	8
3.3.1	Projektdokumentation	8
3.3.2	Möten	9
3.3.3	Tidshantering	9
4	Beskrivning av schemaläggningssystemet	10
4.1	Systemarkitektur	10
4.1.1	Programmeringsspråk	12
4.2	Programdesign	13
4.3	Verktyg	16
4.3.1	FullCalendar	16
4.4	Designmässiga val gällande interaktion	16

5	Resultat	18
5.1	Systemets övergripande delar	18
5.1.1	Databas	18
5.1.1.1	Extern databas	18
5.1.1.2	Intern databas	19
5.1.2	Klient	19
5.1.3	Server	19
5.2	Funktionalitet	19
5.2.1	Schemavy	19
5.2.2	Administratörsvy	20
5.2.2.1	Filtreringsfunktionalitet	20
5.2.2.2	Funktionalitet för att följa en grups alla aktiviteter	21
6	Diskussion och framtida arbete	22
6.1	Diskussion	22
6.2	Framtida arbete	23
7	Slutsats	24
	Litteraturförteckning	26
A	Projektgruppen - Tilde Crew	27
B	Bilder på systemet funktionalitet	30
B.1	Systemets huvudmeny	30
B.2	Systemets inloggningsvy	31
B.3	Informationsruta för en aktivitet	32
B.4	Demonstration av lägga till/ta bort personal	33
B.5	Demonstration av de olika vyerna	34
B.6	Demonstration av schemaläggningsmomentet	35
B.7	Demonstration av filtreringsalgoritmen	36

FIGURER

4.1	Illustration av ett MVC-system och hur dess tre undersystem interagerar med varandra.	11
4.2	Illustration av systemets arkitektur.	12
4.3	Illustration av de fem stegen i <i>Five-step-UML</i>	13
4.4	Systemets Use Case diagram.	14
4.5	Systemets aktivitetsdiagram uppdelat i fyra instanser.	15
B.1	De två olika menyerna för applikationen. Den övre för administratören och den undre för övrig personal.	30
B.2	Inloggningssidan för applikationen.	31
B.3	Personalvyn	32
B.4	Administratörsvyn: Lägga till och ta bort personal	33
B.5	Schemapresentation: Illustrering av de vyer som finns i applikationen.	34
B.6	Sidopanel på Administratörsvyn	35
B.7	En beskrivning av hur filtreringsknapparna agerar utifrån union och snitt.	36

KAPITEL

1

INLEDNING

På Visualiseringscenter C arbetar ett flertal heltids- och deltidsanställda som värdar för den publika verksamheten. Att schemalägga den personal som krävs för olika bokningar är tidskrävande och behöver ofta revideras. Denna rapport beskriver arbetet kring framtagning av ett digitaliserat och lätthanterligt system för att ersätta dagens metoder.

1.1 Bakgrund

De problem som kundens befintliga system inte hanterar är att personal inte når uppdateringar och förändringar i schemat utifrån. Utöver det är schemat också svårläst och själva schemaläggningsmomentet är krångligt för administratören.

I praktiken går schemaläggningsprocessen i det befintliga systemet till enligt följande: Administratören skriver ut en lista med inbokade aktiviteter från bokningssystemet. Därefter skapas en tabell där alla aktiviteter i samma lokal läggs in i samma cell med en tid och eventuell inbokad personal. Schemat går sedan ut till berörd personal via mail för att få in anmälningar till de pass som saknar personal. När anmälningar tagits emot går en ny uppdaterad version ut på ytterligare ett mail. Den skarpa versionen av schemat hänger sedan i personalrummet och det är enbart där som sena förändringar går att se. Det finns således problematik för både administratör och personal.

Ur administratörens perspektiv ligger det största problemet i själva schemaläggningsmomentet, då schemat är svåröverskådligt. Förutom det är det även problematiskt att alla bokningar inte syns i den utskrivna listan, vilket är direkt kopplat till hur själva bokningssystemet är uppbyggt. Det slutgiltiga schemat är svåröverskådligt och det är således lätt att råka dubbelboka eller skriva fel då det inte finns någon koppling eller automatisk validering mot den utskrivna listan. För berörd personal är schemat dels svårtillgängligt, för att få den senast uppdaterade versionen, men primärt är problemet att hitta sina egna pass i schemat. Annan information som är värdefull, till exempel vilka andra som jobbar samtidigt eller om gruppen har flera aktiviteter på rad, är även den onödigt svår att läsa ut från schemat.

Kunden önskar ett nytt system där personalen enkelt ska kunna nå senaste versionen och där själva

schemaläggningsmomentet ska vara enklare och mer systematiserat. De önskar också att schemat går att filtrera i olika vyer och att det ska kunna gå att visa alla pass för en enskild anställd.

1.2 Syfte och frågeställning

Syftet med produkten är att göra schemat tillgängligt, enkelt att redigera, visuellt tilltalande samt intuitivt att läsa.

Den huvudsakliga frågan lyder: Går det att med hjälp av en IT-lösning underlätta och effektivisera schemaläggning av personal och att förenkla dennes åtkomst till schemat, utifrån ett givet och redan etablerat bokningssystem? Vilka är i så fall de mest passande teknikerna och systemstrukturerna för att skapa detta system?

1.3 Avgränsningar

Produkten är avgränsad till att vara en webbaserad applikation som är skräddarsydd för den databas som hämtas direkt från kundens befintliga bokningssystem med dess fördefinierade tabeller och typer.

MÅL OCH PRIORITERINGAR

För att uppnå önskemål och förväntningar på systemet har målsättningar för slutprodukten tagits fram.

2.1 Mål för produkten

Projektets mål är satta utifrån de krav som kunden ställt på lösningen och den funktionalitet som kundens befintliga lösning inte hanterar. Målen komponerades tillsammans med kund och en tidigare anställd som berörts av det tidigare systemet. Önsksningar och problemställning är därefter översatta till krav sammanställda i produktens kravspecifikation.

De övergripande målen för produkten är:

- Att skapa ett system där alla aktiviteter som schemaläggs är väl synliga, för både personal och administratörer, via en hemsida.
- Att administratörer ska kunna schemalägga personal på bokningar hämtade från databasen.
- Att passerade veckor arkiveras, för att i efterhand kunna nås, i ett lämpligt icke-serverkrävande format.

Målen är därefter omformulerade till *funktionella* krav, de specificerade funktioner som systemet ska leva upp till, och *icke-funktionella* krav, övriga krav som handlar om systemet i sin helhet, för systemet.

2.1.1 Funktionella krav

De funktionella kraven på systemet är:

- Att visa ett lättåtkomligt schema som all personal kan ta del av (minsta godkända produkt).
- Att ge möjlighet att filtrera schemat i olika vyer och genom olika parametrar, så som lokaler eller personal.
- Att ge möjlighet att länka personal till en aktivitet i kalendern.

- Att kunna sammankoppla systemet med anställdas Google Kalender.
- Att utföra en semiautomatisk beräkning av schema.
- Att skapa personalinlogg med kompetenslista.
- Att ge möjlighet att prenumerera på scheman.

I praktiken innebär det att utgångspunkten är att skapa ett schema, därefter skapa filtreringsmöjligheter på de parametrar som kunden önskat (personal och lokaler). Nästa steg är skapa funktioner för administratören, primärt en funktion för att kunna koppla en anställd till en aktivitet i kalendern. Därefter ligger fokus på att underlätta schemaläggningsmomentet i form av att kunna se anställdas Google Kalendrar för att avgöra vilka ur personalen som ska visas som tillgängliga för varje aktivitet, och att automatiskt generera ett förslag på personalläggning för den aktuella veckan. Ett annat mål för att underlätta schemaläggningen för administratören är att varje anställd har en kompetenslista över de moment de bemästrar. Dessa listor kan sedan användas för att endast visa den personal som hanterar stationen. Att slutligen kunna prenumerera på schemat, där varje anställd själv kan bestämma vilka filtreringsparametrar som ska gälla och kunna få uppdateringar direkt till sin Google Kalender är det sista och därmed lägst prioriterade kravet.

2.1.2 Icke-funktionella krav

De krav som inte är direkt kopplade till funktionalitet för systemet kallas icke-funktionella krav. Dessa krav som ställs på systemet kan delas upp i fyra olika kategorier: kapacitetskrav, kvalitetskrav, säkerhetskrav och designkrav.

2.1.2.1 Kapacitetskrav

För att skapa ett system som går att använda på det sätt som kund behöver ska systemet hantera att minst 30 personer ska kunna vara inloggade samtidigt. Därutöver ska alla användare kunna utföra olika uppgifter samtidigt och uppdatering av schema ska kunna utföras samtidigt som annan personal är inloggad.

2.1.2.2 Kvalitetskrav

Användare av systemet består av administratör som skapar och uppdaterar schemat, samt personal som ska kunna se schemat. Dessa två typer skiljs åt genom att administratör har extra funktioner som övrig personal inte har tillgång till.

2.1.2.3 Säkerhetskrav

För att eliminera risken att information når ut på fel sätt ska systemet hantera följande:

- Endast behörig personal ska ha tillgång till schemat.
- Endast administratör ska ha behörighet att skapa och uppdatera schema.
- Inga tidigare scheman ska gå förlorade.
- Schemat ska endast uppdateras vid aktivt beslut av administratör.
- Känsliga uppgifter ska vara skyddade.
- Källkoden ska vara väl dokumenterad och en separat, väl uppdaterad dokumentation ska finnas.

2.1.2.4 Designkrav

Schemat ska vara lättillgängligt, till exempel genom att vara åtkomligt via en pekskärm på kontoret. Personalen ska kunna nå det slutliga schemat via webbläsare, även från mobila enheter. Produkten ska ha ett användarvänligt gränssnitt och följa kundens befintliga grafiska profil.

2.2 Prioriteringar och avgränsningar

Ursprungliga mål och prioriteringar för produkten är framtagna i inledningen av utvecklingsprocessen, något som under arbetets gång löpande revideras i samråd med kund. Minsta godkända produkt, eller *minsta godkända produkt* är att ersätta den befintliga schemavisningen med en digital sådan.

De satta målen är internt rangordnade utifrån den avgörande vikt de har för den färdiga produkten. Med de grundläggande kraven uppfyllda är produkten att betrakta som en god ersättare till det befintliga systemet och de primära problemen är lösta.

Projektet är tidsbegränsat, vilket innebär att en prioriterad lista av mål är av stor vikt, då det är utifrån denna som beskrivningar av systemet görs. Det primära målet är då att skapa den minsta godkända produkten. Därtill läggs nya krav till i projektets mål allteftersom tillgänglig arbetstid finns.

UTVECKLINGSPROCESSEN

För att utvecklingsprocessen ska gå så smidigt som möjligt har det satts upp rutiner för arbetet. Verktyg för agil utvecklingsmetod och en struktur för projekthantering har också fastställts. Nedan beskrivs de rutiner som använts och hur projekthanteringen har gått till.

3.1 Agil utvecklingsmetod: Scrum

Projektet har utförts enligt den agila arbetsmetoden *scrum*, med *sprints* om två veckor (Rubin 2013). Varje *sprint* innehåller de uppgifter som ska utföras under den valda tiden. Innan själva arbetet startade gjordes en *sprint 0* där bland annat projektplanen och systemarkitekturen skapades. Den agila utvecklingsprocessen innefattar att gruppen gör en så kallad *backlog* för produkten. Produktens *backlog* är en prioriteringslista som innehåller önskemål utifrån de krav som ställts upp, där det viktigaste för kunden är listat högst upp. Dessa önskemål beskriver vad som ska möjliggöras, vem som ansvarar för uppgiften och när det ska vara klart. Varje önskemål bryts vidare ned till mindre uppgifter som helst ska motsvara en dags arbete eller mindre. Under *sprint 0* undersöktes även möjliga tekniska lösningar och arbetsgruppen satte sig in i de programmeringsspråk som skulle komma att användas.

En *sprint* inleds med ett scrummöte då gruppen bestämmer vilka önskemål som ska behandlas. I slutet av varje *sprint* hålls ett återkopplingsmöte för att utvärdera det som utvecklarna har arbetat med under den senaste *sprinten*. Då genomförs även en sammanslagning av den skrivna koden. Utöver dessa möten ingår även dagliga scrummöten under pågående *sprint*. Dessa möten är till för att gruppen ska få en kort sammanfattning av vad alla gruppmedlemmar har gjort sedan föregående scrummöte. De är även till för att avgöra om några förändringar måste ske.

Till denna agila process finns många olika verktyg att tillämpa. Efter en kort undersökning av olika verktyg valdes *Trello* att arbeta efter då det är ett gratis verktyg som uppfyller arbetsgruppens behov (Fog Creek Software 2000). *Trello* är ett agilt utvecklingsverktyg som hanterar *sprint backlog* och produktens *backlog* på ett enkelt men kraftfullt sätt online. I *Trello* skapades en ny *board*, en digital motsvarighet till anslagstavla, för varje *sprint*. Gruppen bestämmer gemensamt vilka önskemål från produktens *backlog* som ska flyttas till den aktuella *sprintens backlog*. Varje *board* innehåller i sig fyra listor: att göra, pågående, för kodgranskning och klart att lägga till, där varje uppgift placeras

i ”att göra”-listan och tilldelas en eller flera gruppmedlemmar. Därefter flyttas uppgiften utefter dess status.

3.2 Rutiner

För säkerställande av systemets kvalitet är rutiner för testning, kodgranskning och dokumentation framtagna. Rutinerna följs genom hela projektet och ska vidhållas vid vidareutveckling och underhåll.

3.2.1 Dokumentation

Då systemet ska kunna vidareutvecklas och underhållas läggs vikt vid att all programkod ska vara väl kommenterad och dokumenterad. Detta görs genom tydlig extern och intern dokumentation.

Den interna dokumentationen består av kommentarer i programkoden och beskriver mer i detalj delar av systemet så som klasser och funktioner. All kommentering av programkod ska följa *Google Style Guide* för HTML5, CSS3 och JavaScript (Google Style Guide, version 2.23 & 2.93). Syftet är att all programkod ska vara enhetlig och enkel att sätta sig in i. Det ligger även till grund för hur dokumentationen ska se ut och kontrolleras även vid testning. Intern dokumentation skapas för utvecklare och kan därför innehålla mer tekniskt svåra termer.

Den externa dokumentationen innehåller en övergripande bild av hur systemet och dess komponenter fungerar. Den externa dokumentationen ska vara tillgänglig för en bredare grupp än bara utvecklare och är anpassad efter dessa premisser. Exempel på extern dokumentation är systemarkitektur, programdesign och dokumentation av programkod. I projektet används dokumentationsverktyget *Natural Docs* för att skapa en automatiskt genererad dokumentation av programkod (Natural Docs 2011).

Den webbaserade dokumentationen beskriver strukturen och de olika klasserna systemet består av, och ger användaren möjlighet att navigera mellan de olika funktionerna och klasserna för att få en övergripande bild över systemet. Till systemarkitekturen används *Astah Professional* som verktyg för att skapa olika diagram som representerar systemet (Astah 2014).

För versionshantering används verktyget git med Github som basis. De rutiner som används kring versionshantering för systemet är att:

- För varje nytt område eller ny funktion skapas en förgrening utifrån stammen, den så kallade *master*. På så sätt utgår alla nya förgreningar från ett fullt fungerande program och utvecklingen påverkar i sin tur inte huvudversionen.
- Allt som läggs in i stammen ska vara fullt fungerande.
- För att få slå ihop en förgrening med stammen ska den nya koden vara granskad av en annan utvecklare enligt projektets rutiner för kodgranskning. Den nya koden ska ha välformulerade och förklarande dokumentationskommentarer.
- Vid tillägg till en förgrening eller stammen bifogas en kommentar som beskriver status för uppgiften eller förgreningen, vad som nu fungerar, inte fungerar eller vad som är nästa steg.

3.2.2 Testning och kodgranskning

På rekommendation av handledare har testning under utvecklingen av systemet nedprioriterats till förmån för mer omfattande kodgranskningsmoment. Verifiering av ny funktionalitet har utgjorts av

granskning av både programkod och funktionaliteten i sig vid tillägg. Till projektet väljs en testningsansvarig som är ytterst ansvarig för att uppsatta rutiner för testning följs. Utgångspunkten för denna testning är att ett tillägg eller en ändring ska fungera både i den aktuella utvecklingsmiljön (förgrening) och integrerat i det riktiga systemet (stammen). Kodgranskning utförs av en utvecklare som inte har varit delaktig i framtagandet av tillägget.

Rutiner för projektet är att kodgranskaren ska kontrollera följande:

- Att kod och dokumentation följer gällande kodstruktur, exempelvis gällande kommentarer, indentering osv. Kommentarererna för att skapa dokumentation ska kontrolleras mot det gällande dokumentationsverktygets syntax.
- Att koden är begriplig, överskådlig och lätt går att sätta sig in i för en utomstående.
- Att det inte förekommer direkta felaktigheter i koden som inte uppfattas av kompilatorn.
- Om koden kan optimeras utan att förminska dess tydlighet.
- Om koden oavsiktligt påverkar andra delar av systemet.
- Om ny kod använder eller skriver över tidigare skriven kod. Då ska påverkan av den tidigare koden undersökas på nytt.
- Att funktioner returnerar korrekta värden.
- Vad som händer vid felanrop av en funktion.
- Kontrollera hur koden hanterar mänskligt fel, exempelvis felaktig input.

Efter kodgranskningen ska eventuella justeringar genomföras innan koden sammanfogas med slutprodukten och slutligen testas med de andra komponenterna i ett integrationstest.

3.3 Projekthantering

Under arbetets gång har olika riktlinjer för dokumentation av arbetet, mötesrutiner och tidshantering satts upp. Dessa har blivit utgångspunkt för arbetet i projektgruppen.

3.3.1 Projektdokumentation

Dokumentation för projektets utveckling och projektgruppen finns i form av mötesprotokoll och bilder från mindmapping och skisser från modellering av systemet. Det åligger dokumentationsansvarig att dokumentera alla tillfällen då projektgruppen skissar, diskuterar eller uppdaterar systemets struktur, arkitektur eller någon annan central del av systemet. Syftet med denna typ av dokumentation är att kunna följa utvecklingen på en mer abstrakt nivå än källkodsnivå. Utvecklingen av systemets arkitektur och liknande är primärt intressant för projektgruppen, till skillnad från slutliga versionen av de olika skisserna som ingår som extern dokumentation.

3.3.2 Möten

Varje vecka har inletts med ett veckomöte där hela arbetsgruppen samlas för att gå igenom en dagordning i form av en punktlista. Huvudpunkterna består av det nuvarande läget i projektet samt hur arbetet bör gå vidare. Dagordningen finns öppen i arbetsgruppens Google Drive vilket gör det möjligt för alla att påverka vad som ska diskuteras under mötet. Dokumentation förs direkt i dokumentet av dokumentansvarig. Vanligtvis fastslås även vilka tider som arbetsgruppen ska träffas och arbeta under kommande vecka på detta möte. Inför kommande kundmöten diskuteras frågor att ta upp med kund. Kundmötena hålls regelbundet varannan vecka där produktägaren samt minst en annan utvecklare deltar.

3.3.3 Tidshantering

Det finns olika metoder för att få hjälp med att få en överblick på projektprocessen. Ett verktyg som har använts i detta projekt är Gantt-schema, som visar alla aktiviteter i ett parallellt schema vilket gör det lätt att se vilka aktiviteter som är pågående respektive avslutade. I Gantt-schemat planeras alla projektets moment och huvudområden in, med en start- och en slutpunkt. De olika huvudområdena är bland annat förundersökning, projektplan, projektets sprints, seminarium, deadlines och kundmöten.

För att effektivisera arbetet har även arbetsroller delats ut i projektgruppen. Arbetsrollerna består av en *scrummaster*, produktägare, dokumentansvarig, koordinator, testnings-ansvarig samt att alla i gruppen har rollen som utvecklare. *Scrummaster* och produktägare är roller som ingår i den agila utvecklingsmetoden *scrum*, de övriga tre rollerna infördes av gruppen själv utifrån projektet och gruppens behov. För varje roll författades en rollbeskrivning, som skrevs in i projektplanen för att undvika missförstånd och att se till att ansvarsområden varken förbises eller ingår i två olika poster.

En projektplan framställdes som en sammanfattning av projekts omfattning, tidsestimering, organisation av arbetsgruppen, kvalitetsplan, rutiner för dokumentation, resursplan, testplan, säkerhetsplan, riskhantering samt underhållsplan, och som innehåller kundens behov och krav. På så sätt har den också fungerat som en kommunikation med kund, och gör det även möjligt för kunden att följa projektets process.

BESKRIVNING AV SCHEMALÄGGNINGSSYSTEMET

I arbetets inledande skede tas en övergripande struktur för det tänkta systemet fram. Initialt beaktas möjliga tekniker och plattformar för systemet och beslut fattas att utveckla en webbapplikation. För att skapa ett bra system och undvika att begå samma misstag som redan har begåtts och dokumenterats i andra projekt tas teorier kring systemutveckling i beaktning. Syftet med att bygga upp en övergripande struktur för systemet är att kunna bryta ner utvecklingsprocessen till delprocesser och för att tidigt lokalisera behov och kopplingar mellan systemets komponenter. Under det agila arbetets gång utvecklas och förändras strukturen utifrån nya krav och förutsättningar. Nedan beskrivs några av de teorier inom systemarkitektur och programdesign som har beaktats och beskrivningar av det slutgiltiga systemets struktur.

4.1 Systemarkitektur

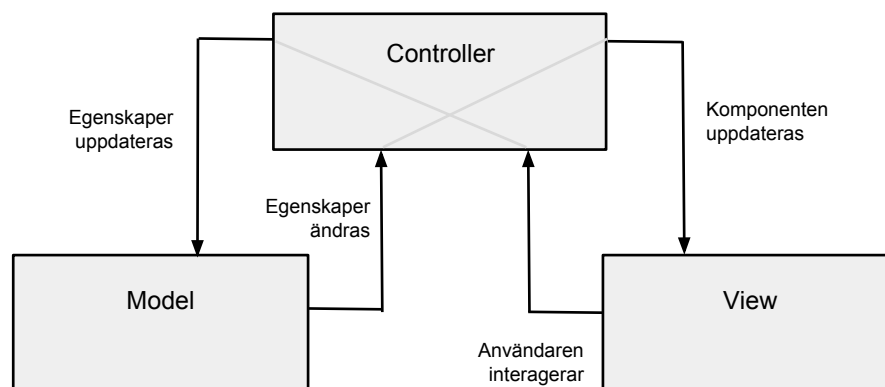
Systemarkitekturen ska fungera som en ritning över det system som ska skapas. I liknelse med ett husbygge, är det av stor vikt att en ritning finns innan ett projekt startas. På så vis kan arbetsteamet börja arbeta i olika delar av systemet men ändå veta att och hur just den byggstenen kommer att passa in i den övriga delen av systemet. I systemarkitekturen definieras vanligtvis de moduler som systemet ska bestå av och hur de kommunicerar med varandra. Varje del av modellen bör vara kopplad till något av kraven som ställts. Systemarkitekturen visar på så vis i stort vilka bjälkar i bygget som ska vara sammanfogade, vilka rum som ska ligga i anslutning till varandra och i vilket material allt ska byggas. Det specificerar däremot inte vilken typ av skruvar som ska användas eller vilken färg väggarna ska ha. Arkitekturen är föränderlig och kan behöva göras om, exempelvis då nya krav ställs på systemet. Vid varje förändring måste en ommodellering göras. Då analyseras hur de nya kraven och dess lösningar passar in i den befintliga arkitekturen. Det är i systemarkitekturen det blir tydligt var i systemet olika krav möts och hur systemet bör byggas upp för att möta kommande krav, exempelvis i form av underhållsmöjligheter. Dokumentet ska vara det stöd som krävs för att samtliga i arbetsgruppen ska kunna arbeta och vari de stora problemen ligger. Arkitekturen kan med fördel uppdateras och refaktoreras under utvecklingens gång.

Vid utveckling av systemarkitektur kan designmönster användas (Gamma et al. 2009). Designmönster utgör ett slags uppslagsverk för kända problem och deras lösningar, där ett designmönster generellt

sett består av fyra delar: ett namn, ett problem, en lösning och konsekvenser. Genom att identifiera problemet för ett system kan ett designmönster väljas, som sedan underlättar utvecklingen.

Systemarkitekturen som projektet har utgått från är designmönstret *Model-View-Controller*, MVC. Designmönstret består av tre undersystem (Lott & Pattersson 2007). I den första av dessa, *Model*, lagras all data som används i MVC-systemet. Datan används i det andra undersystemet, *View*, för att rita ut de komponenter som systemet består av. Användargränssnittet för mjukvaran är i *View*, och det är via denna som användaren interagerar med systemet. I det tredje och sista undersystemet, *Controller*, kontrolleras flödet mellan *View* och *Model*. Ett MVC-system kan bestå av en eller flera av varje undersystem, där varje del kan ansvara för ett särskilt område.

I Figur 4.1 illustreras händelseförloppet i ett MVC-system då en användare interagerar med systemet. Informationen skickas till *Controller* som i sin tur kräver uppdatering av aktuell modell. Efter att all data har uppdaterats skickas den tillbaka till *Controller* som manipulerar och vidarebefordrar den till vyerna. Där uppdateras de komponenter som visas för användaren.

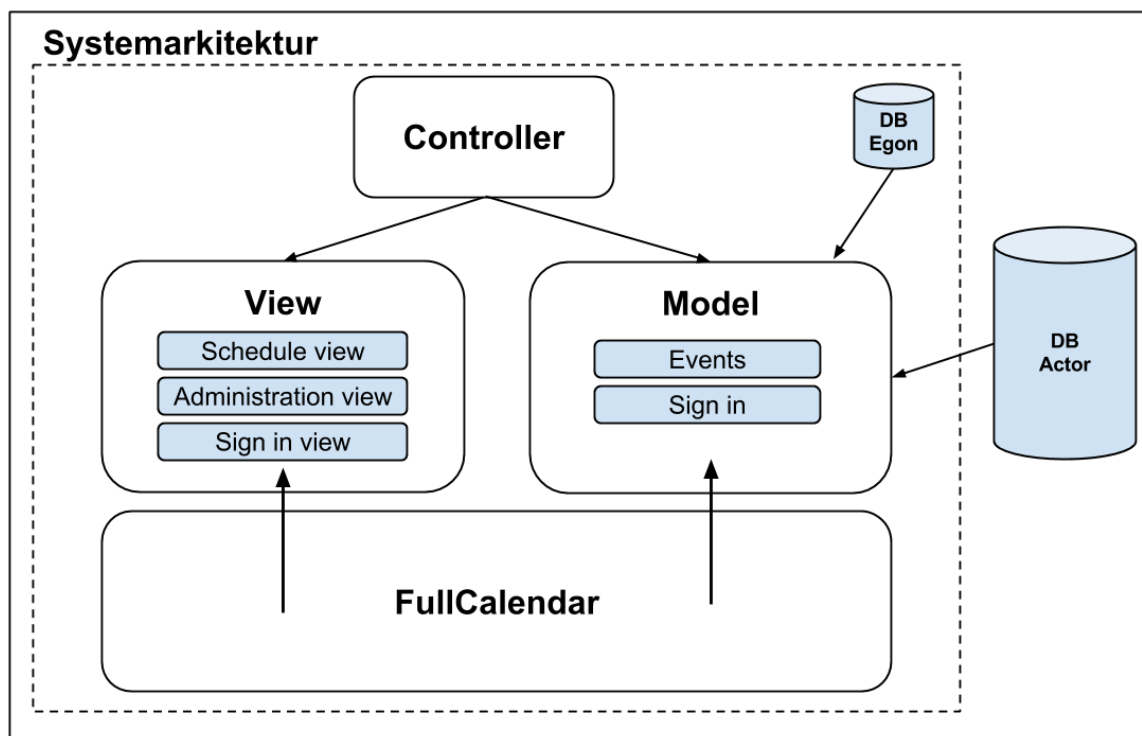


Figur 4.1: Illustration av ett MVC-system och hur dess tre undersystem interagerar med varandra.

Projektets systemarkitektur visas i Figur 4.2. Systemet består i grunden av en MVC-struktur med en insticksmodul, så kallad *plugin*, för att hantera visning av aktiviteter i en kalendermiljö. Data lagras i en lokal och en extern databas. Hemsidan är en så kallad *single page*-applikation, en typ av struktur som kan användas för just webbapplikationer. Single-page innebär att sidan dynamiskt hämtar data som ska visas vilken sedan laddas in i indexfilen, utan att sidan laddas om på nytt. En single page-applikation ökar behovet av en central styrande del, något som *huvud-controllern* sköter i systemet. Genom att utveckla enligt MVC skapas en mer dynamisk mjukvara, som är enklare att modifiera och vidareutveckla. Strukturen innebär att de olika modulerna påverkar varandra minimalt, vilket förenklar utvecklingsprocessen, då det är möjligt att arbeta på olika komponenter i systemet samtidigt.

Det finns två modeller i systemet, en för lagring av data rörande schemalagda aktiviteter och en för hantering av inloggning. Data rörande aktiviteter lagras på klientsidan som JavaScript-objekt av typen *event*, och importeras vid start av applikationen som ett så kallat JSON-objekt från en intern och en extern databas. Modellen för inloggning ligger på serversidan och kontrollerar inloggningen som utförs mot den lokala databasen.

Controllern ligger på serversidan och skickar vid start av applikationen användaren till inloggnings-



Figur 4.2: Illustration av systemets arkitektur.

sidan. När användaren är validerad visar sidan schemat och möjlig navigering i menyn beroende på användarens befogenheter. Övrig funktionalitet och uppdatering av data sker i nämnda *plugin* som används, samt lokala tillägg till denna.

Då användaren skickar en begäran till systemet, genom exempelvis användning av knappar, begär *controllern* först en uppdatering av modellerna, därefter en uppdatering av vyerna på klientsidan. Vyerna uppdateras således med den nya data som tagits fram och användaren får fram resultatet av sin begäran.

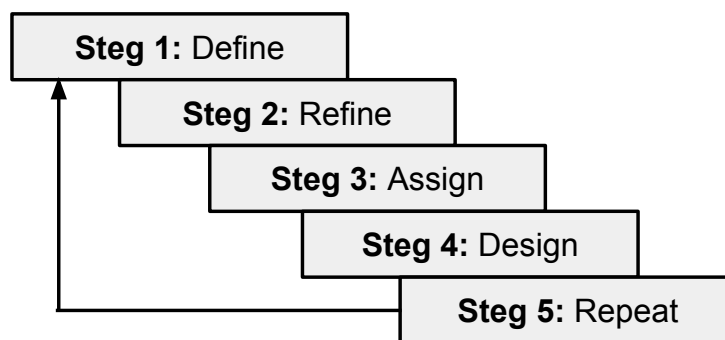
4.1.1 Programmeringsspråk

I applikationen skrivs serversidan i programmeringsspråket PHP, medan klientsidan är skriven i JavaScript och HTML. Den *plugin* som används för schemavisning använder jQuery. I övrigt är den visuella sidan uppbyggd i HTML och CSS. Modellerna är PHP-klasser, vilket även gäller för *controllern*. Vyerna är skrivna i HTML och CSS och kan också genereras från JavaScript-funktioner.

4.2 Programdesign

Programdesignen är den något mer detaljerade ritningen av systemet, där varje moduls innehåll överskådliggörs. Genom att arbeta fram en programdesign skapas även en gemensam struktur för hur systemet ska se ut i de olika delarna. En uppenbar fördel med detta är att det förenklar möjligheten för utvecklarna i arbetsteamet att arbeta i olika delar av systemet vid olika tillfällen, och på så vis undvika en längre startsträcka för att sätta sig in i strukturen. Genom att arbeta på detta vis ökar även möjligheterna till återanvändning av koden.

Programdesignen som projektet har utgått från är *Five-step-UML*. Denna programdesign är en förenklad OOAD, *Object Oriented Analysis and Design*, som går ut på att analysera vilka delar produkten bör bestå av samt dess hållbarhet i avseende på exempelvis plattform (Shoemaker 2004). *Five-step-UML* är enligt Shoemaker även ett exempel på *Model-Driven Development* där modeller skapas för att reflektera systemet och dess egenskaper. De fem steg som utgör denna metod är *Define*, *Refine*, *Assign*, *Design* och *Repeat*, se Figur 4.3.



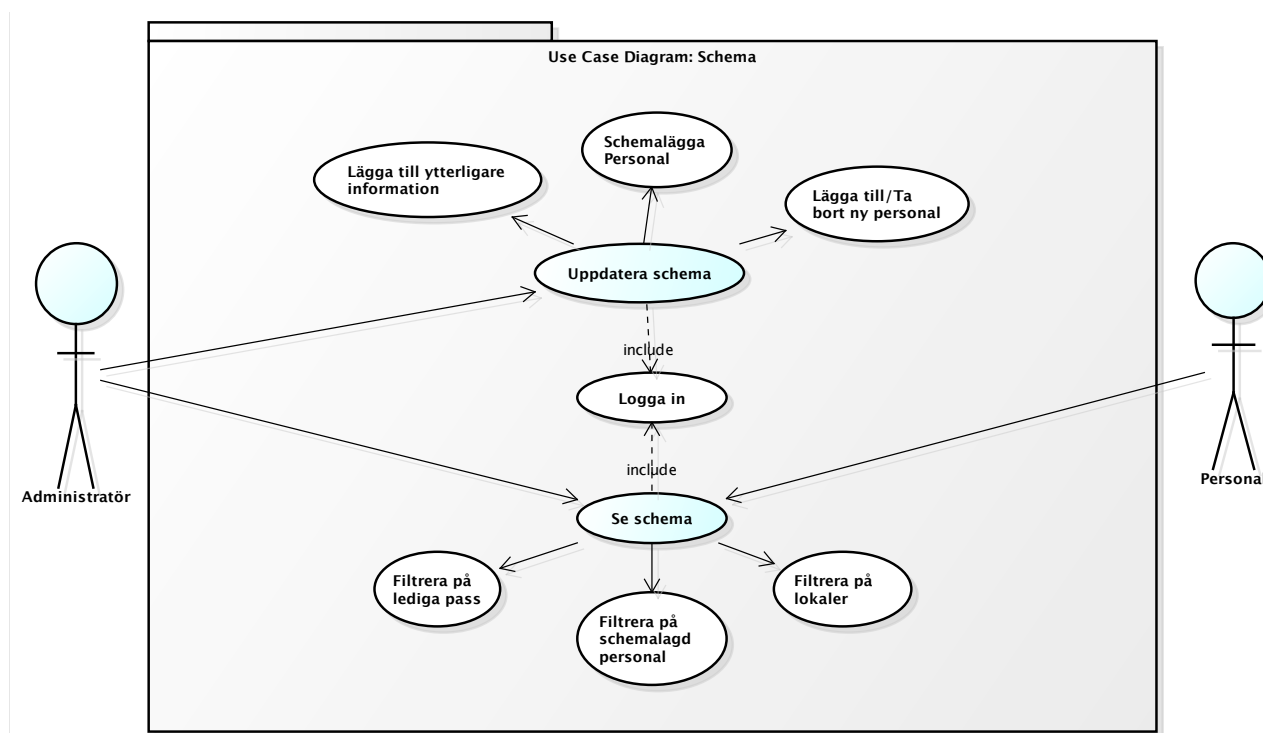
Figur 4.3: Illustration av de fem stegen i *Five-step-UML*.

I det första steget skapas ett så kallat *Use Case Diagram*, där de grundläggande kraven för systemet identifieras. I denna modell skildras aktiviteter av *aktörer* och *domänobjekt* samt vilka relationer de har till varandra. *Aktörer* och *domänobjekt* representerar personer respektive system utanför det egna systemet. I det andra steget skapas ett aktivitetsdiagram där varje krav från föregående steg beskrivs mer utförligt. Detta görs genom att beskriva det önskade beteendet för att hantera kravet samt beskriva andra scenarier som kan uppstå. I det tredje steget vidareutvecklas aktivitetsdiagrammet med element, så kallade *swimlanes*. Varje aktivitet blir då tilldelad ett element som i sin tur beskriver vilken del av systemet de olika aktiviteterna kommer höra till. Exempel på element kan vara klient, databas och server. I det fjärde steget tydliggörs relationerna mellan element och komponenter genom att komponentdiagram skapas. Komponentdiagramen skildrar arkitekturen av hela systemet. I det femte steget undersöks enskilda element och ett klassdiagram skapas. Klassdiagram beskriver de rent tekniska sambanden i koden. I det femte steget kan den interna objektorienterade strukturen på koden skildras. En tydligare bild om de olika klassernas funktioner kan därmed framhävas och i sin tur underlätta implementeringen av koden. Efter att klassdiagram har skapats startas *Five-Step-UML*-processen om för de enskilda klasserna där problemet bryts ned ytterligare.

Shoemaker beskriver att där han väljer att använda aktivitetsdiagram med *swimlanes* kan andra utvecklare tänkas använda sekvensdiagram. Sekvensdiagram kan skapas för att få en övergripande bild av vad systemet ska göra och i vilken ordning. Genom att se förloppet av de olika processerna kan utvecklaren stega genom funktionerna och se var brister och behov kan uppträda. I inledningen av systemdesignprocessen kan sekvensdiagram därmed göra nytta både för att skapa övrig design samt

att sammankoppla arbetsgruppens bild av vad systemet ska åstadkomma.

I projektet tillämpades framtagning av *Use Case*-Diagram och aktivitetsdiagram för att skapa systemets design. I Figur 4.4 illustreras *Use Case*-diagramet för systemet. Produkten är skapad för att användas av två typer av användare: administratör och personal. För att begränsa användarnas möjliga funktioner och förhindra att obehöriga använder systemet krävs inloggning för sidan. Båda typer av användare kan se det aktuella schemat, men endast administratör kan göra förändringar. Förändringar som administratör kan utföra rör personalbokning och tillägg av övrig information om evenemangen.



Figur 4.4: Systemets Use Case diagram.

I Figur 4.5 visas systemets aktivitetsdiagram. Diagrammet består av fyra element: användargränssnittet (*GUI*), FullCalendar, den interna databasen Egon samt kundens externa databas Actor. Diagrammet ger en bild av vilken kommunikation som krävs mellan de olika elementen och hur ofta exempelvis databasanrop behöver utföras. Även säkerhetsaspekter som var kontroll av inloggningsuppgifter sker blir tydligt för utvecklare.

Vid start av applikationen kräver sidan inloggning för att verifiera vilken befogenhet användaren har. Vid felaktiga uppgifter skickas återkoppling med information, och användaren ombeds försöka på nytt. Därefter hämtas data från den lokala och den externa databasen, vilka lagras i modellen i FullCalendar. FullCalendar genererar i sin tur schemat. Användargränssnittet visar de möjliga vyerna samt valmöjligheter så som filtrering och redigering. Beroende på begäran skickas denna vidare till FullCalendar för uppdatering av vyn, alternativt till den lokala databasen för uppdatering av personalbokning eller uppdatering av personallista. Den avslutande aktiviteten är användarens utloggning.

4.3 Verktyg

I utvecklingsprocessen i projektet används ramverket *Bootstrap* för delar av den visuella framställningen (Twitter 2014). I övrigt används inget ramverk för webbutveckling. Istället är strukturen för systemet anpassad utifrån det designmönster som används samt de förutsättningar den *plugin* som används för schemavisning, utgör.

4.3.1 FullCalendar

För funktioner och vyer av schemat används en *jQuery-plugin* kallad *FullCalendar* (Shaw 2013). *FullCalendar* är släppt under en MIT-licens, vilket innebär att den är fri att använda och modifiera och får användas i kommersiella syften. *FullCalendar* erbjuder ett flertal funktioner i form av visning av schema i olika vyer och redigering genom exempelvis *drag-and-drop*.

Inom sina tänkta ramar är verktyget flexibelt med många möjligheter att lägga in egna parametrar. Däremot finns en svaghet i möjligheterna att skapa funktionalitet utanför de ramar som är satta av *FullCalendar*. Exempel på detta är nya vyer och filtrering av evenemang, även popup-fönster har lagts till för att visa information om valt evenemang. För att hantera utökade funktioner skapas två separata instanser, en för den vy som enbart visar schemat och en för de funktioner som används av administratör. *FullCalendar* kompletteras således med filtreringsknappar, popup-fönster med mer information om evenemangen i schemavyn. I administratörsvyn är *FullCalendar* kompletterad med olika schemaläggningsfunktioner som: skapa ny personal, koppla personal till ett evenemang samt lägga till extra text i ett evenemang. Dokumentation för *FullCalendar* finns för befintliga funktioner, dock är den inte till större hjälp ur ett bredare utvecklingsperspektiv då dokumentation för implementation av egna funktioner saknas.

4.4 Designmässiga val gällande interaktion

För att användandet av applikationen ska vara så intuitivt och enkelt som möjligt, är användargränssnittet omsorgsfullt utformat.

Stor vikt har lagts vid att mängden innehåll på sidan ska hållas till det minsta möjliga, för att inte överlasta en användare med information. Den information som presenteras är därför begränsad till enbart det som är relevant för tillfället. Detta görs genom att informationen för bokningar presenteras i två lager, varav enbart det första är synligt från början. Ytterligare information är därefter tillgänglig endast för en vald bokning. På detta sätt förses en användare med lite information i taget, och endast vid begäran om sådan. Detta gör användargränssnittet tydligare, eftersom ingen överflödig information visas. Vidare är verktyg för att manipulera schemat reducerade till att bara innefatta sådana som är aktuella för de aktiviteter som visas. På detta sätt förenklas användandet av applikationen, då alla val som presenteras för en användare alltid har en funktion.

På administratörssidan är upplägget något annorlunda från på schemasidan. För att effektivt nyttja denna funktionalitet behöver en användare en god överblick över schemat. På grund av detta öppnas här inget extrafönster med information om varje bokning, utan denna visas istället i sidopanelen. På detta sätt förloras inte översikten över schemat när en enskild bokning markeras. Av samma anledning finns det på administratörssidan inga filtreringsverktyg, då sådana skulle ta upp för stor del av den totala ytan, och alltså tvinga schemat att vara mindre.

All utformning av sidupplägg och interaktionsverktyg är gjord med tanke på att applikationen ska

vara lätthanterlig både på en datorskärm och på mobila enheter. En önskan från kund är att kunna använda applikationen på en pekskärm på kontoret. Med detta som grund är alla interaktionsverktyg utformade som stora och tydliga knappar. Verktygen för filtrering fungerar som kryssrutor, men eftersom sådana är små och kräver hög precision och alltså inte är lämpliga för enheter med pekskärm, är utseende och storlek för dessa omgjorda. Istället för rullistor för att hantera personalen används av samma anledning knappar även för detta. Likaså är flikarna för att navigera mellan de två delsidorna och kontroller för att styra schemat stora och väldefinierade.

KAPITEL

5

RESULTAT

Resultatet för projektet är en webbaserad tjänst med namnet TildeWeb. Systemet är utformat för två typer av användare, administratör och personal och såväl struktur som användargränssnitt är anpassat därefter. Lösningen är även utformad att vara ett system möjligt att vidareutveckla för att uppnå vidare behov från kund. Systemets struktur och dess grafiska utformning beskrivs nedan.

5.1 Systemets övergripande delar

Systemet arbetar över tre olika delar: databas, klient och server. De ingående komponenterna sköter olika delar av den funktionalitet som applikationen erbjuder. I databasen lagras all information om personal och bokningar, på klientsidan sköts all interaktion med en användare och på serversidan sköts kommunikationen mellan de olika delarna.

5.1.1 Databas

Systemet läser in aktiviteter från kundens bokningssystem, vilket systemet enbart har läsrättigheter till. Eftersom systemet kan lägga till personal och extra text på en aktivitet har en ny databas skapats för att lagra den nya informationen.

5.1.1.1 Extern databas

Den externa databasen är kopplad till kundens bokningssystem och innehåller därför många tabeller och mycket information. Systemet läser från tre olika tabeller, nedan kallade: *bokningar*, *lokaler* och *kundregister*. Från den externa databasen läses aktiviteter in till modellen *events* från *bokningar*. Varje bokning har ett bokningsnummer, en starttid och en sluttid. Det kan även finnas mer information för bokningen så som lokal-ID, text för personalen, kund-ID eller uppgift om bokningsgrupp. Bokningsnumret är unikt för varje aktivitet. Uppgifter som lokal och kund finns som lokal- eller kund-ID i *bokningar* och slås därför upp i sina respektive tabeller för att ett namn på lokalen eller kunden ska erhållas. Tabellen *lokaler* innehåller förutom lokaler även andra typer av bokningar som behöver personal så som ”guide” för guidade turer och ”tekniker” för domen. För att minimera risken att data förloras och för att underlätta utvecklingsarbetet skapas en lokal kopia av databasen som gruppen

arbetar mot. För att implementera systemet till organisationen krävs uppkoppling mot den aktuella databasen. För detta krävs förutom tillgång till denna även uppdatering av inställningar för apache samt server för applikationen.

5.1.1.2 Intern databas

Utöver den befintliga databas som innehåller information om inbokade aktiviteter, lokaler och kunder har en ny databas skapats. Denna databas lagrar information om personalen, kopplingen mellan personal och bokning, samt extra bokningstext. Denna information finns i tre olika tabeller i databasen: *personalinformation*, *personalbokning* och *extra bokningstext*. *Personalinformation* innehåller information om personalen: fullständigt namn, personal-ID, initialer och lösenord för inloggning. I *personalinformation* är både personal-ID och initialer unika. Den andra tabellen, *personalbokning*, lagrar kopplingar mellan en aktivitets bokningsnummer och ID-nummer för personalen, vilket utgör själva schemaläggningen. Den tredje tabellen ger möjligheten att lägga till extra information till bokningarna, utan att behöva ändra i den ursprungliga databasen. Samtliga tabeller uppdateras och hanteras i systemet av användare med administratörsstatus.

5.1.2 Klient

På klientsidan möjliggörs och behandlas all interaktion med en användare av systemet, för att relevant information ska kunna presenteras. Detta görs genom att alla handlingar som utförs av en användare antingen ändrar parametrar direkt i den nuvarande vyn eller skickar information till serversidan. Det som görs direkt på klientsidan använder information som har genererats på servern, och utgörs av handlingar som att markera bokningar, skapa knappar för att hantera personal och att navigera i schemat.

5.1.3 Server

På serversidan görs den övergripande styrningen av systemet. Här finns klasser som hanterar applikationens yttre navigering och filer som sköter all koppling mellan klient och databas. Utifrån en användares handlingar som tas emot via klienten anropas olika objekt och funktioner som används för att uppdatera den aktuella vyn eller för att ändra i den lokala databasen. De bokningar som ska presenteras i schemat sammanställs också på servern. Detta görs med hjälp av förfrågningar till båda databaser och genom att lagra de erhållna svaren i en variabel som är åtkomlig för klienten.

5.2 Funktionalitet

Applikationen är uppdelad efter två olika användningsområden. Det första är att visa ett schema och presentera all relevant information, och det andra är att schemalägga personal för de bokningar som finns. För dessa två användningsområden finns det två olika vyer, kallade *schema* och *administrera*, se Bilaga B.1, Figur B.1. Förknippade med dessa är två olika användare, *personal* och *administratör*, som har olika tillgång till vyerna. Personalen når endast schemat, medan administratören har åtkomst till båda vyer. Kontroll för detta görs på inloggningssidan, se Bilaga B.2, Figur B.2.

5.2.1 Schemavy

Den totala ytan i schemavyn är uppdelad i två delar. Den första utgörs av själva schemat och den andra är en sidopanel för att välja vilket innehåll som visas. All bokningsinformation presenteras i två lager, där det första är vad som visas direkt i schemat. Detta är namn på den lokal som är bokad, initialer för schemalagd personal och bokningens varaktighet. All övrig information presenteras för

en bokning i taget, då en sådan aktiveras genom att en användare klickar på den. Detta görs genom att en informationsruta öppnas över den nuvarande vyn. Här visas, utöver den redan presenterade informationen, namn på den kund som har gjort bokningen, eventuell bokningstext och även sådan text som kan ha lagts till av en administratör för applikationen, se Bilaga B.3, Figur B.3. Schemat är också styrbart för en användare på så sätt att det går att välja mellan tre olika vyer, som representerar olika tidsintervall. Förvalsläget är en tre-dagarsvy, som visar den aktuella dagen i mitten och föregående och nästkommande dag till vänster respektive höger om denna. Utöver denna vy erbjuder applikationen en veckovy och en dagsvy, se Bilaga B.5, Figur B.5. I alla dessa vyer är det också möjligt för en användare att bläddra framåt och bakåt i tiden.

5.2.2 Administratörsvy

Likt schemavyn är administratörsvyn uppdelad i två delar, med skillnaden att filtreringsfunktionaliteten i sidopanelen har ersatts med administratörsverktyg. Dessa används för att schemalägga personal på de bokningar som finns i den externa databasen. Här presenteras också all information som i schemavyn visas i en informationsruta.

Den funktionalitet som tillhandahålls av administratörsverktygen är att schemalägga personal, lägga till extra information på bokningar och att lägga till och ta bort tillgänglig personal, se Bilaga B.4, Figur B.4. Schemaläggning av personal sker genom att en användare markerar en bokning i schemat och använder de knappar som representerar personalen. Dessa knappar genereras beroende på vilken bokning som är vald, och är uppdelade i *inbokad personal* och *tillgänglig personal*. Inbokad personal är sådan som sedan tidigare är schemalagd på den aktuella bokningen, och tillgänglig personal är resterande delar av personalen. Med hjälp av fältet *övrig information* kan fritext läggas till på varje bokning, utan att den externa databasen behöver manipuleras, se Bilaga B.6, Figur B.6.

Vid tillägg av personal på en bokning görs en kontroll av eventuell dubbelbokning för aktuell medarbetare. Detta görs genom en jämförelse av bokningars start- och sluttider för att hitta sådana som överlappar varandra. Därefter görs en kontroll av huruvida aktuell personal är bokad på en överlappande aktivitet. I detta fall presenteras en dialogruta som informerar användaren om att det gjorda valet medför en dubbelbokning. Med denna dialogruta ges också användaren valet att trots detta slutföra bokningen eller att avbryta den.

5.2.2.1 Filtreringsfunktionalitet

För att göra schemat överskådligt används filtrering, där användaren själv kan påverka vilka aktiviteter som ska visas. De parametrar som filtreringen erbjuder och hanterar är olika lokaler och personal. Varje parameter presenteras som en knapp i GUI:t. För att minimera antalet knappar så visas bara knappar rörande de lokaler och den personal som har inbokade aktiviteter. Knapparna är uppdelade i två grupper: personal och lokaler. Om flera parametrar ur samma grupp är aktiverade visas unionen av de valda parametrarna, till skillnad från om parametrarna är från olika grupper, då visas snittet av de olika grupperna. En demonstration av filtreringsalgoritmen visas i Bilaga B.7, Figur B.7

I filtreringsfunktionen ingår det två andra knappar, *obemannade pass* och *rensa*. Obemannade pass fungerar som personalknappar, förutom att den visar enbart de aktiviteter utan personal. Denna knapp går således att kombinera med filtrering på lokalerna, men är designad så att den inte går att använda i samband med filtrering av personal. Rensa-knappen tömmer alla filtreringsknappar och laddar om schemat med alla aktiviteter. Rensa-knappen är aktiv om det finns något att rensa, det vill säga om någon filtreringsknapp eller obemannade pass-knappen är aktiverad. Om ingen filtreringsknapp är aktiverad är rensa-knappen inaktiverad.

5.2.2.2 Funktionalitet för att följa en grupps alla aktiviteter

En önskan från kunden är att i schemat kunna se alla aktiviteter som en kund bokat. Ofta har en kund flera aktiviteter och det är därför värdefullt för personalen att veta var gruppen är före och efter en aktivitet för att kunna ge så bra bemötande som möjligt. På informationsrutan som visas då användaren klickat på en aktivitet finns knappen *Följ grupp*. Samtliga aktiviteter som inte tillhör samma grupp visas då något transparent, vilket gör att de efterfrågade aktiviteterna framträder tydligare.

DISKUSSION OCH FRAMTIDA ARBETE

Produkten som är framtagen täcker de mål som satts upp för att skapa en minsta godkända produkt. I kapitlet diskuteras positiva och negativa aspekter av arbetets upplägg samt möjliga vidareutvecklingar av systemet.

6.1 Diskussion

Vid uppstarten av projektet tog det tid att komma in i den agila arbetsmetoden. Det lades stor vikt vid de dokument som skulle upprättas och ligga som grund för arbetet. Detta gjorde att själva utvecklingsarbetet med produkten inleddes senare än önskat. Först när den agila processen kommit till rätta började utvecklingsarbetet att flyta på för gruppen. Det främsta misstaget var formulering och storlek på uppgifterna som direkt kan kopplas till brist på kunskap inom området.

För att göra systemet lättillgängligt och plattformsoberoende valdes en webbapplikation som lösning. Något ramverk för webbutveckling användes inte, endast *Bootstrap* för den visuella delen av sidan. I övrigt valdes användande av ramverk bort, delvis på grund av det var svårt att avgöra vad som behövdes innan strukturen var satt. I efterhand hade det varit bra att använda något ramverk för klientsidan, exempelvis Angular.js eller Prototype, då det blev mycket fokus på sådana saker som ett ramverk enkelt skulle lösa.

Insticksmodulen FullCalendar har haft både för- och nackdelar. Det har fungerat som en bra utgångspunkt för att visa schemat och har haft de grundläggande funktioner som krävs för detta. Dock har det varit komplicerat att lägga till extra funktioner som FullCalendar har saknat. Andra insticksmoduler undersöktes först då svagheter i FullCalendar uppbådats, då var arbetet tyvärr redan långt gånget i FullCalendar och det var svårt att motivera ett byte av insticksmodul i det läget. En annan aspekt värd att nämna är att modulen är väl dokumenterad men svår att navigera i, vilket försvårat oerhört för utvecklingsarbetet då det tagit mycket tid till att enbart förstå struktur och händelseförlopp i modulen. FullCalendar kan med fördel användas vid enklare applikationer med begränsade behov utanför de ramar som finns, däremot bör andra hjälpmedel övervägas vid bredare utveckling.

Designmönster var något som diskuterades mycket om det skulle implementeras eller inte. När det

väl skulle appliceras hade utvecklingsarbetet redan börjat och det blev således några komplikationer vilket fördröjde implementationen. MVC applicerades på serversidan och då FullCalendar som utgör majoriteten av systemet ligger på klientsidan blev det svårt att inkludera hela systemet i designmönstret. Det gjordes då ett aktivt val att gå ifrån designmönstret för FullCalendar med motiveringen att det inte är värt den tiden det skulle ta att modifiera och anpassa modulen till designmönstret. MVC fungerade slutligen bra för att underlätta strukturen för systemet, även om det inte är en fullvärdig MVC-struktur.

Då projektet är webbaserat har körtester inte genomförts under testning och kodgranskning. Det har medfört att rutinerna för kodgranskning varit många och omfattande för att ersätta de klassiska testningsmetoder som inte kunnat tillämpas. Det är svårt att ersätta körtester med mänsklig granskning då det ställer höga krav på analytiskt tänkande och djupgående kunskap inom programmeringsspråket.

Stor vikt har lagts på att göra ett system som är användarvänligt och anpassat för de behov som användarna har. Kunden berättade tidigt att de önskade att använda systemet på en touchskärm och att personalen ska kunna nå schemat snabbt. Det är något som genomsyrat hela utvecklingsprocessen och legat till grund för många designmässiga val. Ramverket *Bootstrap* har använts primärt för att skapa en responsiv tjänst som fungerar även på mobila enheter. Tyvärr har inga genuina användartester genomförts, men däremot har nuvarande administratören, tillika administratör för systemet testat systemet vid kundmöte. En stor utmaning var att göra schemavyn överskådligt, vilket har försvårats då den databas kunden gav var utdaterad och innehöll bokningar från 2010 då centret öppnade. Då det var många annorlunda bokningar och bokningssystemet användes på ett annat sätt än vad det görs idag, innebar det att det var svårt att utläsa och analysera hur en normal vecka ser ut. Därför har det varit svårt att avgöra hur många bokningar som normalt ligger parallella och liknande. Fokus har istället lagts på att skapa en bra och tydlig filtreringsfunktionalitet för att underlätta för användare snarare än att förändra renderingsmetoden och hanteringen av parallella aktiviteter i FullCalendar.

I retrospekt kan det konstateras att en teknisk handledning hade varit av stor hjälp i inledningen av projektet, främst på grund av den bristande tekniska kompetens som gruppen innehade inom just området webbutveckling. Däremot har just detta faktum inneburit att projektgruppen haft möjlighet att utvecklas inom systemutveckling och förmåga att ta fram den teori och de praktiska lösningar som krävs. Detta har dock skett på viss bekostnad av kvaliteten på produkten.

6.2 Framtida arbete

Följande krav skulle kunna behandlas vid en eventuell framtida utveckling:

- Att öppna koppling mellan systemet och aktuell databas.
- Att ge möjlighet att prenumerera på schemat, med vald filtrering, till sin Google Kalender.
- Att en anställds schemalagda tid sammanställs för tidsrapportering.
- Att skapa en kompetenslista för personal och utgå från denna för att ge förslag på personal vid schemaläggning.
- Att skapa möjlighet till arkivering av scheman.

KAPITEL

7

SLUTSATS

Att ta fram ett system för hantering av schemaläggning är möjligt och kan förenkla arbetet för samtliga anställda på Visualiseringscenter C. Den optimala plattformen anses avslutningsvis vara webben, främst eftersom det innebär hög lättillgänglighet. Det ökar även möjligheterna att bygga ett självständigt system. Det ger även en lägre inlärningströskel för användare av systemet, då de flesta är vana att använda internetapplikationer i någon utsträckning. Nedan presenteras målen för projektet samt hur de har fullföljts.

Visa ett lättåtkomligt schema som all personal kan ta del av (minsta godkända produkt).

Resultatet är en webbaserad tjänst för att visa det aktuella schemat. Samtlig personal med befogenhet har därmed tillgång till schemat, som innehåller samma information som det ersatta systemet.

Möjlighet att filtrera schemat i olika vyer och genom olika parametrar.

Vyn för schemat kan skräddarsys utifrån behov, tidsintervall, lokaler, personal eller bokningar.

Möjlighet att länka personal till en aktivitet i kalendern.

Personal med administrationsbehörighet har möjlighet att schemalägga personal från specifika bokningar. Administratörer kan även själva lägga till och ta bort personal ur systemet.

Sammankoppling med anställdas Google Kalender.

Målet är inte uppnått och prioriterades ned tidigt i processen i samråd med kund. Funktionaliteten är inte bedömd som nödvändig för det färdiga systemet.

Semiautomatisk beräkning av schema.

Ett färdigberäknat förslag på möjligt schema är mycket eftertraktat av kund, men kräver både kompetenslista för personal samt algoritmer för att sammanställa de två databaserna och beräkna bästa möjliga schema. Övriga aspekter som tidsåtgång hos personal, maximal arbetstid inom samma dygn m.m. behöver också beaktas. Magnituden av arbetet är orsaken till att arbetet inte inletts under projektets gång.

Skapa personalinlogg med kompetenslista.

Kompetenslistor för personal skulle i viss mån kunna förenkla arbetet för administratör. Dock är det

i dagsläget inte nödvändigt att ha en sådan då personalstyrkan inte är större än att det går att hantera utan en digital lista. Det ligger även en nackdel i det behov av att uppdatera listorna som uppstår.

Möjlighet att prenumerera på scheman.

Den förenklade versionen av schemat underlättar åtkomsten till eget schema. Möjlighet att prenumerera på sitt eget schema har därmed på grund av tidsbrist bortprioriterats.

Arkivering av tidigare schema.

De uppdateringar som utförs av administratör lagras i den interna databasen, vilket innebär att ingen data går förlorad. Vid hämtning av valt tidsintervall för bokningar visas således den schemaläggning som gjorts. I projektet betraktas inte detta som en fullgod lösning för arkivering, dels eftersom det är möjligt för administratör att redigera tidigare scheman och dels för att tidsintervall är satt redan i utvecklingsstadiet. För att betrakta målet som uppnått bedöms att vidareutveckling krävs för denna funktionalitet.

LITTERATURFÖRTECKNING

- Astah. (2014). *Astah Professional* (Version 6.8). [UML-verktyg] Tillgänglig: <http://astah.net/>
- Fog Creek Software. (2000). Trello [Programvara]. Tillgänglig: <https://trello.com/>
- Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (2009). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Google Style Guide (Version 2.23). Google HTML/CSS Style Guide.
<http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml> [2014-05-13].
- Google Style Guide (Version 2.93). Google JavaScript Style Guide.
<http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml> [2014-05-13].
- Lott, J. & Patterson, D. (2007). *Advanced ActionScript with Design Patterns*. Peachpit Press.
- Natural Docs. (2011). *Natural Docs 1.52* (Version 1.52) [Dokumentationsverktyg] Tillgänglig: <http://www.natindocs.org/>
- Rubin, K. S. (2013). *The Essential Scrum - A Practical Guide to the Most Popular Agile Process*. Addison Wesley.
- Shaw, A. (2013). *Fullcalendar* (Version 2.0.0) [Ramverk]. Tillgänglig: <http://arshaw.com/fullcalendar/download/>
- Shoemaker, M. L. (2004). *UML Applied: A .NET Perspective*. Apress.
- Twitter. (2014). *Bootstrap* (Version 3.1.1) [Ramverk]. Tillgänglig: <http://getbootstrap.com/getting-started/#download>

BILAGA

A

PROJEKTGRUPPEN - TILDE CREW

Lovisa Dahl

Dokumentationsansvarig

Hanterade stories från backlog:

- Som utvecklare vill jag ha en databas för hantering av systemets funktionalitet.
- Som användare vill jag kunna filtrera schemat för en så enkel hantering som möjligt.
- Som kund vill jag att systemet ska kunna kopplas upp för att kunna användas.
- Som ägare av systemet vill jag att all kod ska vara väl dokumenterad och testad.
- Som ägare vill jag ha information och rätt konfigurerings för att hantera min databas.
- Projektplan
- Kravhantering
- Rapport

Emma Forsling Parborg

Koordinator

Hanterade stories från backlog:

- Som administratör vill jag kunna schemalägga personal utifrån bokade aktiviteter.
- Som användare vill jag ha en lättnavigerad och visuellt tilltalande sida.
- Som ägare av systemet vill jag att användare ska avkrävas inloggning.
- Som administratör vill jag kunna lägga till och ta bort personal ur personallistan.
- Som ägare av systemet vill jag att all kod ska vara väl dokumenterad och testad.
- Projektplan

- Kravhantering
- Rapport

Martin Gråd

Dokumentansvarig

Hanterade stories från backlog:

- Som administratör vill jag kunna schemalägga personal utifrån bokade aktiviteter.
- Som utvecklare vill jag att sidan ska vara en single-page applikation.
- Som administratör vill jag få en varning om jag är på väg att dubbelboka personal.
- Som användare vill jag ha en lättnavigerad och visuellt tilltalande sida.
- Som ägare av systemet vill jag att användare ska avkrävas inloggning.
- Som användare vill jag ha enkel åtkomst till övrig information om bokade aktiviteter.
- Som administratör vill jag kunna lägga till och ta bort personal ur personallistan.
- Som administratör vill jag kunna lägga in extra text på bokade aktiviteter.
- Som ägare av systemet vill jag att all kod ska vara väl dokumenterad och testad.
- Projektplan
- Kravhantering
- Rapport

Linnea Malcherek

Projektledare och Scrum master

Hanterade stories från backlog:

- Som personal vill jag kunna följa en bokningsgrupp
- Som utvecklare vill jag ha en databas för hantering av systemets funktionalitet.
- Som användare av systemet vill jag att olika grupper av bokade aktiviteter ska vara färgkodade för enklare navigering.
- Som ägare av systemet vill jag att all kod ska vara väl dokumenterad och testad.
- Projektplan
- Kravhantering
- Rapport

Julia Nilsson

Kundkontakt och produktägare

Hanterade stories från backlog:

- Som utvecklare vill jag ha en databas för hantering av systemets funktionalitet.
- Som användare vill jag ha enkel åtkomst till övrig information om bokade aktiviteter.
- Som kund vill jag ha kontinuerlig kontakt med utvecklingsteamet och ha löpande avstämningar.
- Som ägare av systemet vill jag att all kod ska vara väl dokumenterad och testad.
- Projektplan
- Kravhantering
- Rapport

Linnéa Nåbo

Testningsansvarig

Hanterade stories från backlog:

- Som personal vill jag enkelt kunna navigera mellan olika kalendervyer.
- Som användare vill jag kunna filtrera schemat för en så enkel hantering som möjligt.
- Som utvecklare och underhållare av systemet vill jag ha ett verktyg för hantering av schema-visning.
- Som ägare av systemet vill jag att all kod ska vara väl dokumenterad och testad.
- Projektplan
- Kravhantering
- Rapport

BILAGA

B

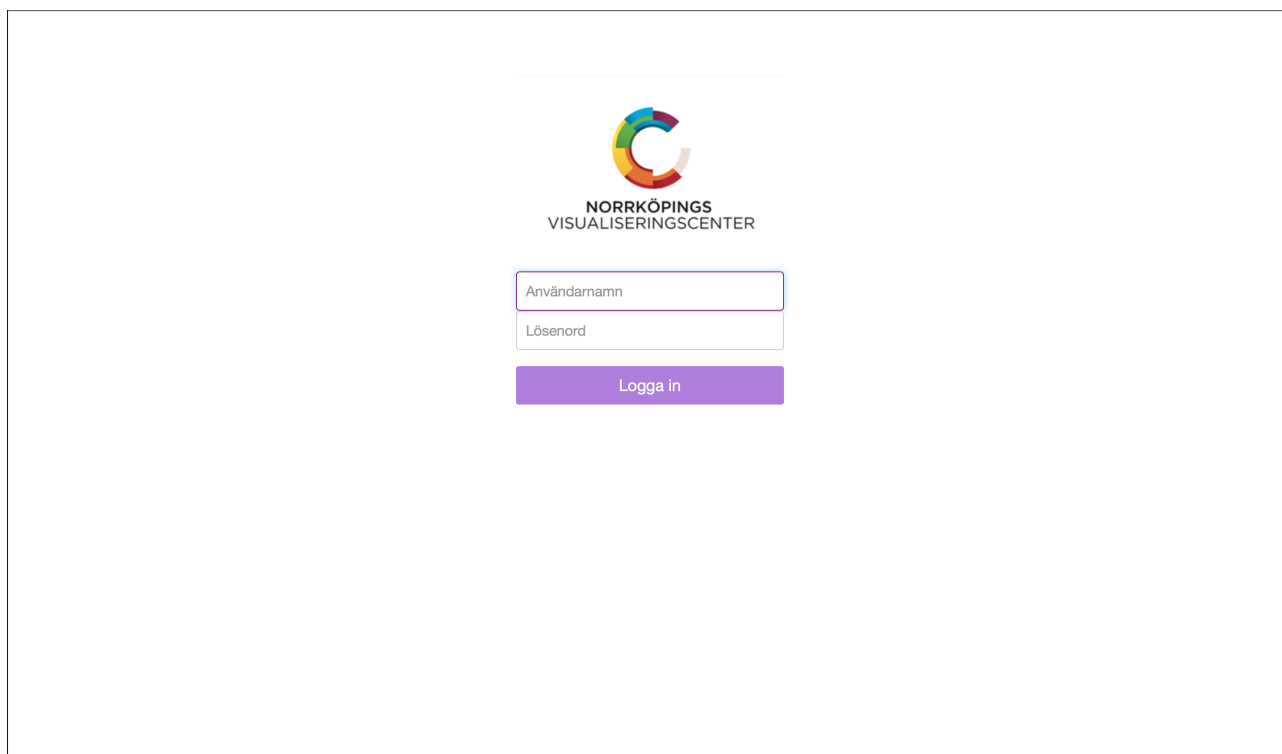
BILDER PÅ SYSTEMET FUNKTIONALITET

B.1 Systemets huvudmeny



Figur B.1: De två olika menyerna för applikationen. Den övre för administratören och den undre för övrig personal.

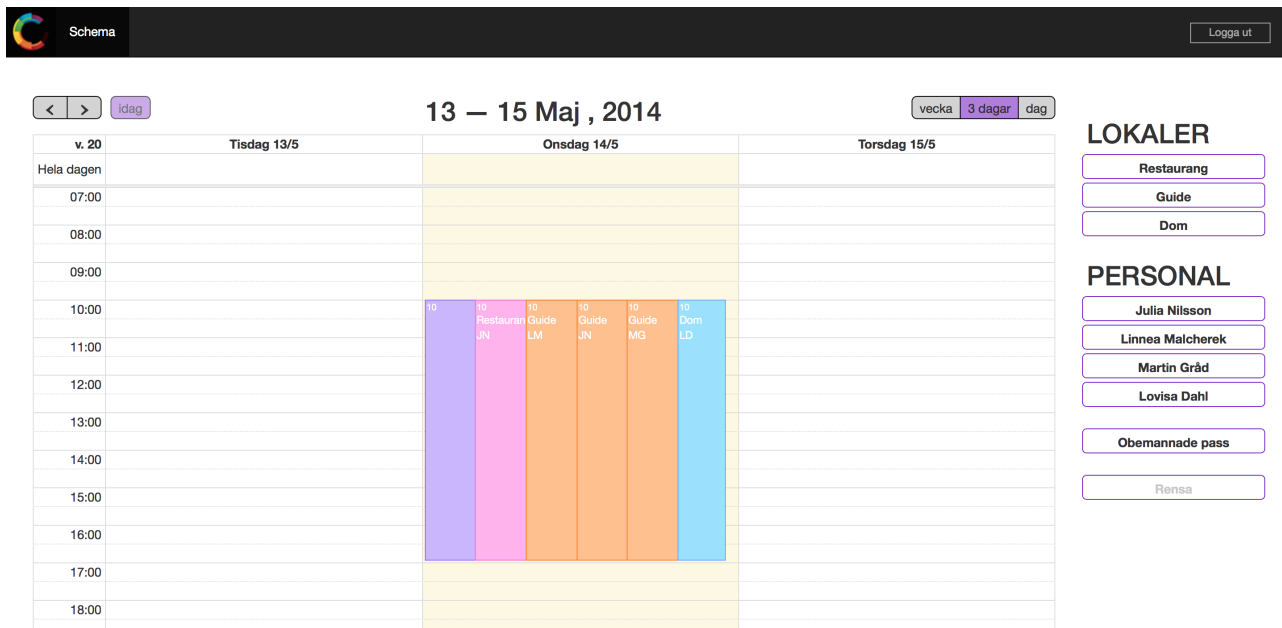
B.2 Systemets inloggningsvy



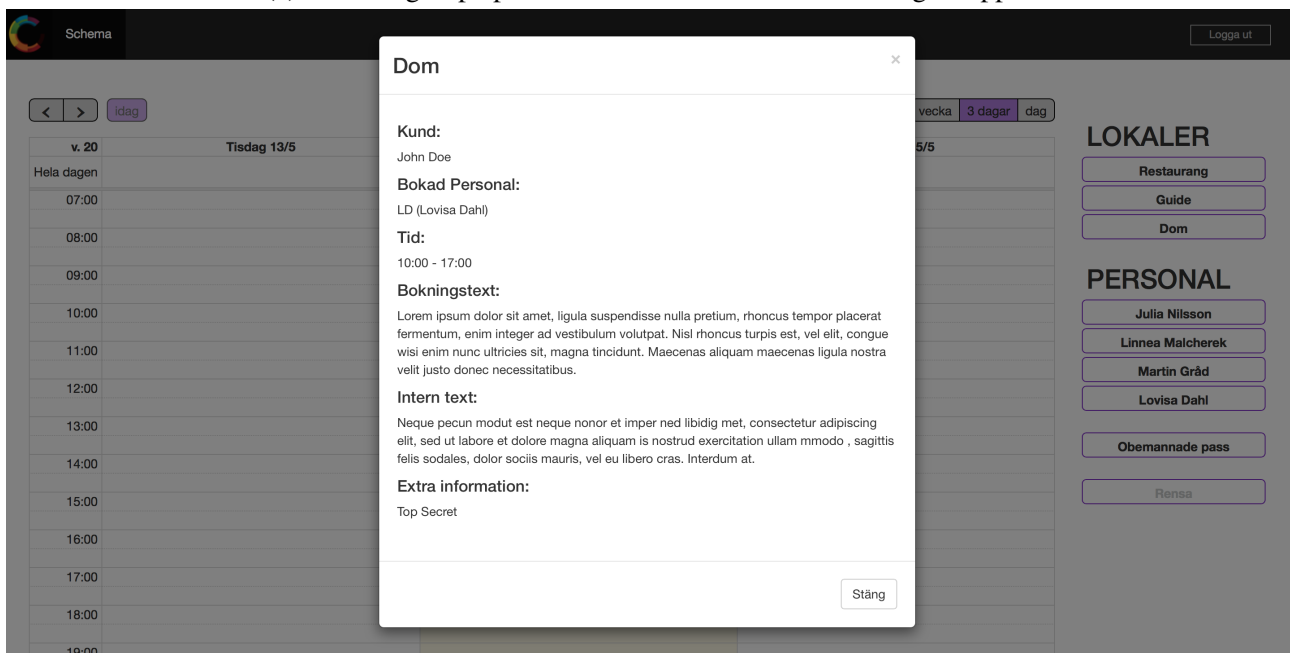
The image shows a login page for the application. At the top center is a logo consisting of a stylized 'C' made of colorful segments. Below the logo is the text "NORRKÖPINGS VISUALISERINGSCENTER". Underneath this, there are two input fields: "Användarnamn" (Username) and "Lösenord" (Password). Below the input fields is a purple button labeled "Logga in" (Login).

Figur B.2: Inloggningssidan för applikationen.

B.3 Informationsruta för en aktivitet



(a) Första lagret på personalsidan: Schemat och filtreringsknappar



(b) Andra lagret på personalsidan: Informationsruta för vald bokning

Figur B.3: Personalvyn

B.4 Demonstration av lägga till/ta bort personal

The figure consists of two screenshots of a web application interface. Both screenshots show a background calendar view for 'Tisdag 13/5' with a time slot from 07:00 to 12:00. A modal dialog box titled 'Lägg till/ta bort personal' is open in the center of the screen. In the top screenshot, the 'Lägg till' tab is selected, and the input fields for 'Namn' and 'Initialer' are empty. In the bottom screenshot, the 'Ta bort' tab is selected, and the input fields contain the text 'Pippi Långben' and 'PL' respectively. A 'Lägg till' button is visible in both screenshots.

(a) Lägg till ny personal

The figure consists of two screenshots of a web application interface. Both screenshots show a background calendar view for 'Tisdag 13/5' with a time slot from 07:00 to 14:00. A modal dialog box titled 'Lägg till/ta bort personal' is open in the center of the screen. In the top screenshot, the 'Ta bort' tab is selected, and a list of names is displayed: Emma Forsling Parborg, Julia Nilsson, Linnea Malcherek, Linnéa Nåbo, Lovisa Dahl, Martin Gråd, and Pippi Långben. Pippi Långben is highlighted. In the bottom screenshot, the same list is displayed, and Pippi Långben is highlighted.

(b) Ta bort personal

Figur B.4: Administratörsvyn: Lägga till och ta bort personal

B.5 Demonstration av de olika vyerna

< > idag 12 – 14 Maj , 2014 vecka 3 dagar dag

v. 20	Måndag 12/5	Tisdag 13/5	Onsdag 14/5
Hela dagen			
07:00			
08:00			
09:00			

(a) Tredagarsvy

< > idag Tisdag, 13 Maj, 2014 vecka 3 dagar dag

v. 20	Tisdag 13/5
Hela dagen	
07:00	
08:00	
09:00	

(b) Dagsvy

< > idag 12 – 18 Maj , 2014 vecka 3 dagar dag

v. 20	Mån 12/5	Tis 13/5	Ons 14/5	Tors 15/5	Fre 16/5	Lör 17/5	Sön 18/5
Hela dagen							
07:00							
08:00							
09:00							

(c) Veckovy

Figur B.5: Schemapresentation: Illustrering av de vyer som finns i applikationen.

B.6 Demonstration av schemaläggningsmomentet

Lägg till/ta bort personal

(Välj en bokning)

Lägg till/ta bort personal

Vald bokning

Inbokad personal:
Personal saknas

Tillgänglig personal:
Emma Forsling Parborg
Julia Nilsson
Linnea Malcherek
Linnéa Nåbo
Lovisa Dahl
Martin Gråd

Kund:
Saknas

Bokningstext:
Saknas

Intern text:
Saknas

Övrig information:
Klicka här för att skriva.

Lägg till information

Lägg till/ta bort personal

Vald bokning

Inbokad personal:
Personal saknas

Tillgänglig personal:
Emma Forsling Parborg
Julia Nilsson
Linnea Malcherek
Linnéa Nåbo
Lovisa Dahl
Martin Gråd

Kund:
Saknas

Bokningstext:
Saknas

Intern text:
Saknas

Övrig information:
Klicka här för att skriva.

Lägg till information

Lägg till/ta bort personal

Vald bokning

Inbokad personal:
Lovisa Dahl

Tillgänglig personal:
Emma Forsling Parborg
Julia Nilsson
Linnea Malcherek
Linnéa Nåbo
Martin Gråd

Kund:
Saknas

Bokningstext:
Saknas

Intern text:
Saknas

Övrig information:
Top Secret

Lägg till information

Lägg till/ta bort personal

Vald bokning

Inbokad personal:
Lovisa Dahl

Tillgänglig personal:
Emma Forsling Parborg
Julia Nilsson
Linnea Malcherek
Linnéa Nåbo
Martin Gråd

Kund:
Saknas

Bokningstext:
Saknas

Intern text:
Saknas

Övrig information:
Klicka här för att skriva.

Lägg till information

Lägg till/ta bort personal

Vald bokning

Inbokad personal:
Lovisa Dahl

Tillgänglig personal:
Emma Forsling Parborg
Julia Nilsson
Linnea Malcherek
Linnéa Nåbo
Martin Gråd

Kund:
Saknas

Bokningstext:
Saknas

Intern text:
Saknas

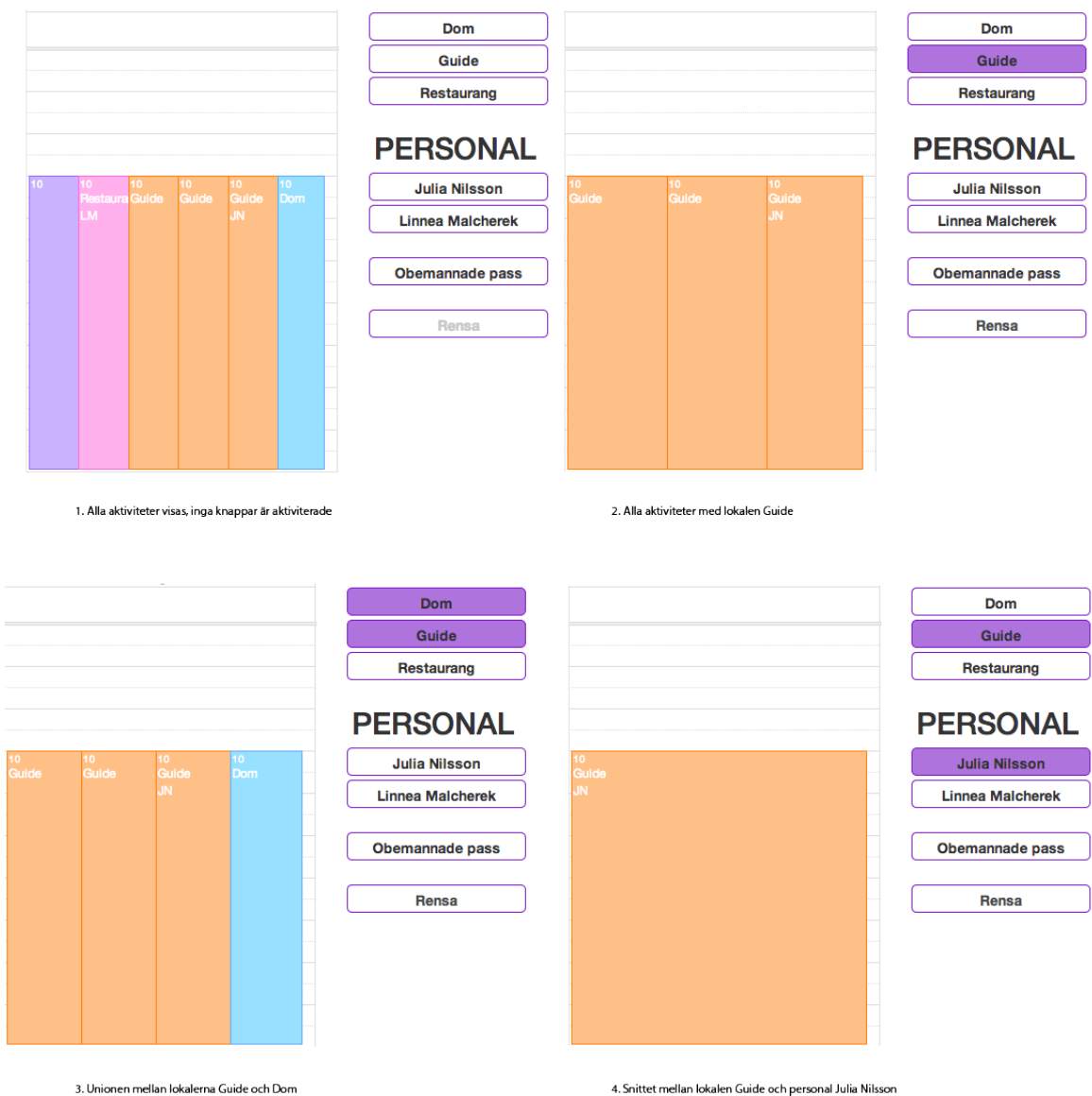
Övrig information:
Top Secret

Lägg till information

Extra information tillagd!

Figur B.6: Sidopanel på Administratörsvy

B.7 Demonstration av filtreringsalgoritmen



Figur B.7: En beskrivning av hur filtreringsknapparna agerar utifrån union och snitt.