Kandidatarbete

Redaktör: Pål Kastman

Datum: 2015-04-20

Version 0.2

Dokumenthistorik

Datum	Version	Utförda ändringar	Utförda av
2015-03-13	0.1	Första utkast av gemensam och individuella rapporter	Pål Kastman
2015-04-20	0.2	Andra utkastet av gemensam och individuella rapporter	Alla

Projektidentitet

Detta dokument gäller för grupp 7 i kursen TDDD77 på Linköpings universitet

Namn	Ansvarsområde	E-post
Daniel Rapp	Teamledare	danth407@student.liu.se
Daniel Falk	Analysansvarig	danfa519@student.liu.se
1 Jonas Andersson	Arkitekt	jonan111@student.liu.se
Albert Karlsson	Kvalitetssamordnare	albka735@student.liu.se
Erik Malmberg	Testledare	erima694@student.liu.se
Pål Kastman	Dokumentansvarig	palka285@student.liu.se

Kund
Region Östergötland

Kundkontakt
Daniel Hall, daniel.hall@regionostergotland.se
Erik Sundvall, erik.sundvall@regionostergotland.se
Ingrid Hallander, ingrid.hallander@regionostergotland.se

Handledare
Lena Buffoni, lena.buffoni@liu.se

Handledare
Kristian Sandahl, kristian.sandahl@liu.se

Innehåll

Del I
Gemensamma erfarenheter och diskussion

1 Inledning

Detta avsnitt behandlar varför detta projekt utförs.

1.1 Motivering

Region Östergötland har idag ett system med handböcker som en sjuksköterska går igenom inför varje operation. I dessa handböcker finns bland annat förberedelse-uppgifter och plocklistor. Handböckerna är idag inte interaktiva på något sätt, istället skrivs plocklistan och förberedelseuppgifterna ut och bockas av för hand. Plattformen med handböcker kan heller inte återanvändas på olika avdelningar på grund utav licensproblem. Utöver detta system så finns ett annat separat system, som heter kartoteket, för uppgifter om vilka artiklar som finns och var i lagret de ligger. Detta gör att personalen som ska förbereda inför operationer behöver gå in i två olika system om de inte vet var alla artiklar ligger.

1.2 Syfte

Uppgiften som gruppen har fått är att skapa ett nytt system med handböcker som har interaktiva förberedelse-och plocklistor. Listorna ska uppdateras kontinuerligt när de bockas av så flera personer kan jobba på dem samtidigt. Plocklistan ska också innehålla uppgifter om var artiklarna ligger. Tanken är att personalen ska använda en iPad för listorna så de kan gå runt och plocka i lagret och bocka av samtidigt.
I mån av tid ska också extra funktionalitet implementeras. Till exempel sortera plocklistan med avseende på närmsta väg mellan artiklarna, lagersaldo och media i handböckerna.
Hela systemet ska ligga under en open-source licens så det kan användas fritt av alla.

1.3 Frågeställning

Rapporten ska besvara följande frågeställningar.

- Hur kan ett system för operationsförberedelser realiseras så arbetet blir lättare och mer effektivt?
- Kan man använda plattformen keystone för att bygga detta system?

1.4 Avgränsningar

Kunden ville till en början att vi skulle integrera ett lagersaldo i systemet och göra det möjligt att scanna av artiklar varefter man tar dessa. Vidare ville

kunden även att systemet skulle fungera som ett större system så att alla kliniker i regionen skulle vara anslutna.

Vi kännde att tiden inte skulle räcka till för att åtadkomma dessa implementeringar och klargjorde därför för kunden i ett tidigt skede att vi hellre ville få klart grunden i systemet, så att de skulle få ett likvärdigt men ändå till viss del förbättrat system.

En kompromiss gjordes där vi valde ett ge dessa krav prioritet 2 vilket innebar att kraven var önskvärda och skulle implementeras i mån av tid, eller prioritet 3 där kraven sågs som en framtida utbyggnad.

2 Bakgrund

Studenterna som studerar kursen TDDD77 fick i januari 2015 ett uppdrag att utföra ett kandidatarbete. Först fick gruppen rangordna flera projektdirektiv för att sedan få ett av dessa uppdrag tilldelat sig. Grupp 7 fick då projektet operationsförberedelser som skickades in av Region Östergötland.

3 Teori

Node.js Keystone.js

3.1 Wkhtmltopdf

Wkhtmltopdf är ett program som tar en hemsida eller en html-fil och skapar en pdf av den. Programmet har många olika parametrar som kan ändras t.ex. mediatyp så pdf-filen kan se ut som en utskrift av hemsidan. För att kunna använda detta programmet med nodejs finns det också en nodejs-modul med samma namn som skickar kommandon till programmet.

4 Metod

I detta avsnitt beskrivs det hur arbetet med projektet har gått till.

4.1 Kravinsamling

Gruppen utgick ifrån projektdirektivet och började tänka över hur systemet skulle byggas. Det kom fram ganska snabbt att ingen i gruppen hade koll på hur operationsförberedelser går till, vilket gjorde att utbildning inom detta krävdes. För att få mer insyn så gjordes ett studiebesök på universitetssjukhuset i Linköping. Under detta studiebesök gjordes bestämdes också hur insamlingen av krav skulle gå till. Analysansvarig utsågs till ansvarig för detta. Denne skulle med hjälp av kunden arbeta fram en kravspecifikation som båda parter vara nöjda med. Detta gjordes genom flera möten och ett gemensamt dokument på Google drive där båda parter kunde gå in för att redigera och skriva kommentarer.

4.2 Utvecklingsmetod

Projektet har utvecklats iterativt med en utvecklingsmetodik som påminner mycket om SCRUM. Aktiviteterna i projektet har visats på en gemensam SCRUM-board med hjälp av webbsidan Trello. Boarden hade kategorierna TODO, DOING och DONE. När en medlem valt en aktivitet så märktes aktiviteten med medlemens namn och flyttades till den aktuella kategorin. Varje vecka har gruppen haft ett kort SCRUM-möte på 15 minuter. På mötet har varje gruppmedlem berättat vad som har gjorts den föregående veckan och vad som ska göras nästa vecka. SCRUM-mötet har oftast utförts i samband med varje veckas handledarmöte.

Ett system med alpha state cards har använts för att gruppen ska kunna veta hur långt projektet har fortskridit. Korten har även använts till att identifiera aspekter av projektet som kan förbättras och som gruppen behöver arbeta mer med. De aspekter av projektet som kunnat förbättras har använts som mål för kommande iterationer. De olika korten har uppdaterats kontinuerligt under projektets gång.

Nästan all utveckling har skett med hela gruppen samlad i samma lokal. Det har gjort att alla frågor om andra gruppmedlemmars kod har kunnat besvaratas snabbt. Det har gjort att arbetet kunnat hålla hög fart och att inga onödiga förseningar har uppstått på grund av osäkerheter i hur en del av koden fungerar. Problem har också kunnat lösas fort eftersom alla gruppmedlemmars olika kompetens har funnits tillgänglig.

4.3 Forskningsmetod

5 Resultat

5.1 Gruppens gemensamma erfarenheter

5.2 Översikt över de inviduella utredningarna

6 Diskussion

6.1 Resultat

6.2 Metod

6.3 Arbetat i ett vidare sammanhang

7 Slutsatser

8 Fortsatt arbete

Del II
Enskilda utredningar

A Kartoteket - Daniel Rapp

A.1 Inledning

Idag är information om Region Östergötlands operationsartiklar, så som priserna och placeringen i lagret på tandborstar, tandkräm, handskar och annan medicinsk utrustning, hanterat av ett internt system. Detta system kallas ett "*kartotek*".

A.1.1 Syfte

Det största problemet med det nuvarande kartoteket är att det inte är integrerat med systemet för att hantera handböcker.

I dagsläget, om en artikel slutas säljas eller Region Östergötland väljer att inte köpa in en viss artikel längre så tar de bort artikeln från kartoteket. Problemet som då uppstår är att detta inte reflekteras i handböckerna. Så om t.ex. en "*Oral-B Pro 600 CrossAction*" tandborste används i en "*Laparoskopisk sigmoideumresektion*", och tandborsten slutas säljas så tas den bort från kartoteket, men eftersom handboken för operationen inte är kopplad till kartoteket så uppdateras det inte att denna artikel inte längre finns i lagret.

Syftet med denna del av systemet är att lösa detta problem genom att integrera systemet som hanterar handböcker tillsammans med ett nytt kartotek, allesammans byggt på webben. När en artikel ändras eller tas bort i kartoteket så ändras den även i alla handböcker för operationer som kräver denna artikel.

A.1.2 Frågeställning

Frågeställningar:

- Går det att integrera systemet för handböcker med kartoteket utan att förlora funktionalitet?

A.1.3 Avgränsningar

Förutom ett förbättrat avcheckningsssytem så är Region Östergötland också i behov av ett bättre system för att hantera deras lager på ett mer automatiserat sätt. Bland annat så skulle de behöva ett system som låter dem checka in vilka varor från lagret de hämtat ut, istället för att checka av manuellt, vilket kan vara felbenäget.

Vi har dock valt att avgränsa oss från att bygga denna lösning, på grund av tidsbrister.

B Jonas Andersson

B.1 Inledning

Tidigare när jag har utvecklat webbprogram så har jag använt mig av PHP tillsammans med HTML, css och javascript. Jag har dessutom valt att skriva mycket själv och inte förlita mig på ramverk och bibliotek. I detta projekt valde jag, som arkitekt, istället att byta ut PHP mot node.js. Dessutom har jag lagt mycket vikt på använda bibliotek så mycket som möjligt för att slippa uppfinna hjulet på nytt.

B.1.1 Syfte

Syftet med denna del av rapporten är att analysera vad det finns för fördelar och nackdelar med att använda node.js gentemot andra vanliga språk för webben. Det ska även undersökas hur inlärningskurvan beror på språk och externa ramverk och bibliotek.

B.1.2 Frågeställning

Frågeställningar

- Vad finns det för fördelar/nackdelar med node.js gentemot andra språk för webben?
- Vad finns det för fördelar/nackdelar med att använda mycket externa bibliotek?

B.1.3 Avgränsningar

Det finns många programmeringsspråk som man kan använda i samma syfte som node.js. Eftersom jag enbart har tidigare erfarenheter av PHP och Python så kommer node.js jämföras mot dessa och inga andra.

B.2 Bakgrund

B.3 Teori

Node.js är en plattform för att skapa applikationer till framförallt webbservrar. Det finns en inbyggd pakethanterare vid namn npm som gör det enkelt att inkludera både små och stora bibliotek i sina projekt. Det är därför väldigt enkelt att använda sig av ett bibliotek istället för att skriva all funktionalitet själv.

Javascript är det programmeringsspråk som används i Node.js. På klienten, d.v.s. i webbläsaren, är man tvingad att använda sig av javascript eller något programmeringsspråk som kan kompileras till javascript. Genom att använda node.js får man därav samma språk på både server och klient.

B.4 Metod

B.5 Resultat

B.6 Diskussion

B.6.1 Resultat

B.6.2 Metod

B.7 Slutsatser

B.8 Referenser

C Pål Kastman

C.1 Inledning

I detta projekt så har vi valt att inledningsvis skriva dokumentationen i Googles ordbehandlingsprogram Google Docs för att i ett senare skede då det var dags att påbörja denna rapport gå över till att använda typsättningssystemet Latex.

Vi valde att göra på detta sätt för att så snabbt som möjligt komma igång, då man bland annat i Google Docs kan se live vad andra skriver.

C.1.1 Syfte

Syftet med denna individuella del är att undersöka funktionaliteten i Latex och väga fördelar mot nackdelar.

C.1.2 Frågeställning

- Vad finns det för begränsningar i Latex
- Kommer det att vara en fördel eller en nackdel för flera gruppmedlemmar att jobba samtidigt i samma delar av rapporten.

C.1.3 Avgränsningar

Denna rapport kommer att avgränsas för att endast jämföra Latex, Microsoft Office och Google Docs, detta för att inte behöva jämföra alla olika ordbenhandlingsprogram.

C.2 Bakgrund

Dokumentering är någonting som är viktigt att göra när man arbetar i projektform, dels för egen del utifall man behöver gå tillbaka och se vad som gjorts, men även för andras skull ifall man kanske får en ny medarbetare som ska integreras i projektet. En fråga som man alltid behöver besvara är i vilket ordbehandlingsprogram dokumentationen skall skrivas. Det populäraste alternativet kan tänkas vara Microsoft Word, vilket har funnits sedan 1983 [1].

Ett annat alternativ som blir allt vanligare är Google Docs, vilket är ett web-baserat ordbehandlingsprogram som lanserades av Google 2006 [2] efter att man hade köpt upp företaget Upstartle [3].

Ett tredje och inte lika vanligt alternativ är Latex, vilket inte är ett ordbehandlingsprogram utan istället ett märkspråk så som t.ex. HTML, där man istället för med ett grafiskt gränssnitt formaterar sin text,sätter sin text inom speciella taggar så att när koden senare kompileras får rätt utseende.

Latex är egentligen bara en uppsättning makron skrivna för språket Tex, vilket skapades av den amerikanske matimatikern Donald Knuth 1978 [4]. I början av 80-talet så vidareutvecklade Leslie Lamport Tex med hjälp av dess makrospråk till det som idag är Latex [5].

Vad det gäller min egen bakgrund i Latex så hade jag när detta projektet påbörjades endast använt det i ett tidigare projekt, under det projektet blev jag dock inte så insatt i Latex utan fyllde endast i material i de redan färdiga mallarna vi hade.

Under kursens gång har jag dock skrivit en laborationsrapport i Latex och därigenom skaffat mig lite mer erfarenhet.

C.3 Teori

När Microsoft utvecklade Word på 80-talet så ville de naturligtvis att dåtidens datorer skulle klara av att ladda dokument utan att det skulle vara alltför prestandakrävande, därför konstruerade man dess filformat (.doc) binärt och gjorde konstruktionen väldigt komplex. Detta medförde att bara personer som var riktigt insatta i detta filformat klarade att ändra direkt i dessa filer. Google docs däremot använder sig av molnetför att spara filer och man behöver därför aldrig oroa sig över säkerhetskopiering. Textformatering i båda dessa program görs främst genom att använda det grafiska gränssnittet, men kan även göras genom tangentbordskommandon.

Latex är i motsats till Word och Google docs inget ordbehandlingsprogram, utan istället ett märkspråk liksom HTML. När man i Word och Google docs ändrar textformateringen genom ett grafiskt gränssnitt så markerar man istället sin text med taggar, som senare när man kompilerar koden ger det önskade utseendet. Detta gör att man som användare behöver bry sig mindre om utseendet av dokumentet och kan istället fokusera på innehållet. Eftersom inte är något program i sig utan mer ett programspråk så behöver man något program att redigera koden i, detta kan göras i vilken textredigerare som helst egentligen men det finns även alternativ som kan kompilera koden åt användaren så att man kan se direkt i programmet vilka ändringar som görs. Det har på senare år dykt upp flera webbsidor där man kan skriva Latexdokument tillsammans med andra användare, exempel på sådana sidor kan vara Overleaf [6], eller Authorea [7]. Latex har väldigt många inbyggda kommandon och det är väldigt lätt att t.ex. skriva formler vilket har gjort Latex till standard inom den vetenskapliga sektorn [8].

C.4 Metod

Vi har till en början valt att under projektets gång arbeta i endast en fil för den gemensamma delen av kandidatprojektet. Vad det gäller de individuella delarna så har vi valt att lägga dessa i separata filer och sedan importera dessa till den gemensamma. Vi sparar alla dokumentfiler i samma repository på github som vi har källkoden.

Genom att göra på detta sätt så kan vi garantera att det inte kommer att uppstå några konflikter i de individuella delarna. Det som vi inte vet, är huruvuda det kommer att uppstå problem då alla gruppmedlemmar skall skriva på den gemensamma delen.

C.5 Resultat

När vi i projektets inledning använde Google docs så märkte vi att det var väldigt svårt att hålla sig till någon sorts dokumentstandard där man t.ex. använder samma typsnitt på rubriker och text. Även om alla projektmedlemmar har försökt att hålla sig till den standard vi satt upp så blev det ändå ibland att någon del av dokumenten blev annorlunda när vi redigerar samtidigt i dessa.

Detta fungerade mycket bättre i Latex då vi skrev kandidatrapporten. Latex hjälper verkligen användaren till att hela hålla samma standard i dokumenten.

Vi har under projektet försökt att hålla isär på alla olika revisioner av dokument så att man senare kan gå tillbaka och se vilka ändringar som har gjorts mellan de olika iterationerna, detta har varit lite svårt att upprätthålla med Google docs då dessa dokument hela tiden sparas kontinuerligt och det inte finns något sätt att gå tillbaka i ett dokument för att se ändringar som gjorts, detta har gjort att när vi gjort ändringar i dokument, behöver vi spara undan en kopia av den föregående revisionen vilket inte alltid har gjorts. I kandidatrapporten däremot då vi haft alla våra Latexfiler i samma repository på github så har vi enkelt kunnat gå tillbaka i historiken för att se vilka ändringar som gjorts, och då även lätt kunnat korrekturläsa varandras texter.

C.6 Diskussion

C.6.1 Resultat

C.6.2 Metod

C.7 Slutsatser

C.8 Referenser

[1] `http://ia801406.us.archive.org/21/items/A_History_of_the_Personal_Computer/eBook12.pdf`
Sida 11-12. Hämtad 2015-04-20.

[2] `http://googlepress.blogspot.se/2006/06/google-announces-limited-test-on-google_06.html`
Hämtad 2015-04-28.

[3] `http://googleblog.blogspot.se/2006/03/writely-so.html`
Hämtad 2015-04-28.

[4] `https://gcc.gnu.org/ml/java/1999-q2/msg00419.html`
Hämtad 2015-04-20.

[5] `http://research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html#latex`.
Hämtad 2015-04-20.

[6] `https://www.overleaf.com/`
Hämtad 2015-04-28.

[7] `https://www.authorea.com/`
Hämtad 2015-04-28.

[8] `ftp://ftp.dante.de/tex-archive/info/intro-scientific/scidoc.pdf`
Sida 2. Hämtad 2015-04-28.

D Daniel Falk

D.1 Inledning

Jag har i detta projekt haft rollen som analysansvarig vilket innebär en analys av kundens behov och framställning av krav utifrån dessa. Rapporten beskriver hur kravframställningen gått till i detta projekt och hur prototyper har använts för att kommunicera idéer.

D.1.1 Syfte

Syftet med denna del av rapporten är att ta reda på hur man kan samla in krav som kunden har på produkten. Fokus ligger på användningen av prototyper med utgångspunkt i detta projekt.

D.1.2 Frågeställning

Hur kan man samla in krav som tillfredställer kundens verkliga behov? Hur kan man arbeta med prototyper för att framställa krav?

D.1.3 Avgränsningar

Rapporten har sin utgångspunkt i hur kravframställningen gått till i detta projekt. Metoden bör inte ses som ett allmänt tillvägagångssätt.

D.2 Bakgrund

Att förstå kundens verkliga behov utgör grunden för ett lyckat projekt. Ett projekt faller ofta på grund av dålig kravframställning.

D.3 Teori

Teorin beskriver kravinsamlingsmetoder, verktyg och vilka problem som kan uppstå vid kravframställning.

D.3.1 Problem

Det finns ett antal barriärer som kan uppstå vid kravframställning. Ett problem är ofta att kunden inte kan formulera vad de vill ha. De kan se problemet men inte vad som behöver göras. De kan överdriva vissa problem medan andra förbises. Ett annat problem är att kunden kan ha fastnat i ett invant mönster. De kan ha svårt att föreställa sig nya sätt att utföra en uppgift på. [1]

D.3.2 Vertktyg

Prototyper

D.3.3 Kravinsamlingsmetoder

D.4 Metod

Denna del beskriver hur kravframställningen gått till i detta projekt. Först beskrivs arbetet med att analysera kundens behov under förstudien. Vidare beskrivs mer ingående vilka metoder som använts och hur kraven valdes att representeras. Avslutningsvis beskrivs vilka användartester som genomfördes och hur dessa bedrog till utvecklingen.

D.4.1 Förstudie

Den största delen utav analysarbetet skedde under förstudien. Här identifierades de olika intressenterna och en kravspecifikation utarbetades. Våran första kontakt med projektet var en projektbeskrivning där kunden formulerade sina mål och visioner av projektet. Några vikta krav gavs också såsom att prototypdesign skulle genomföras i samarbete med kunden. Ett första möte utav fyra under förstudien gav sedan mer information och vi började våran kravinsamling. Vi kunde konstatera att vi hade två olika intressenter att arbeta med. Dels sjuksköterskorna som är användare av systemet och dels CMIT, Centrum för medicinsk teknik och IT, som ansvarar för sjukhusets IT-miljöer. För att förstå användarnas behov hölls ett studiebesök där vi fick en visning av nuvarande system och hur det används. Detta var nödvändigt för att verkligen förstå vad det var som behövde göras och vad som kunde förbättras. Från CMIT:s sida hölls mer tekniska möten där teknikval diskuterades. Här var det viktigt att ta reda på vilka begränsningar som fanns och vilka val som passade våra och deras erfarenheter.

D.4.2 Kravinsamlingsmetoder

För att samla in krav under möten med kund har vi dels fört anteckningar på papper eller dator och dels spelat in på mobil. När vi varit flera personer på möten har vi gått igenom och diskuterat våra anteckningar i efterhand. Vid oklarheter har vi antecknat dessa för att förtydliga med kund. Inspelning användes främst vid första mötet och var användbart då vi fick mycket information att sätta oss in i. Kravspecifikationen skrev i Google docs. Detta valdes eftersom kraven utarbetades tillsammans med kund. Ett gemensamt redigerbart dokument gav en möjlighet för oss att arbeta på olika platser under kravframställningen vilket var effektivt. En kommentarsfunktion gav oss möjligheten att kommentera krav och föreslå förbättringar. Under interna möten och kundmöten var den gemensamma redigeringen också till nytta då kravformuleringar snabbt kunde genomföras.

D.4.3 Kravrepresentation

Kraven gavs en prioritetsordning för att kunna urskilja de mest väsentliga kraven. Detta kändes nödvändigt då vi var begränsade av en tidsbudget. En prioritering av kraven gav utrymme för vidareutveckling i mån av tid. Krav med prioritet 1 var att betrakta som grundkrav som skulle genomföras för att projektet skulle ses som godkänt. Krav med prioritet 2 var att betrakta som önskvärda

och som skulle genomföras om då grundkraven var genomförda. Krav med prioritet 3 var krav som fångats upp med som skulle ses som framtida utbyggnad.

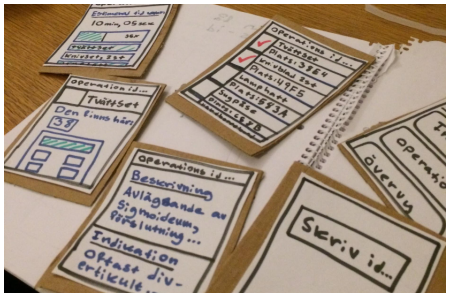
Kraven numrerades för att lätt kunna refereras till under projektets gång. De delades också in i olika sektioner efter deras del i systemet. Sektionerna var plocklistor, handböcker, kartotek, lagersystem. En extra sektion för generella krav användes. Denna uppdelning kändes naturlig för detta projekt.

Kravspecifikationen skrevs med stöd från standarden IEEE 830. Enligt standarden ska ett krav vara korrekt, otvetydigt, färdigt, konsekvent, prioriterat, verifierbart, modifierbart och spårbart. Detta eftersträvades men det kan diskuteras om alla krav passerar dessa filter. I slutändan var det ändå våran gemensamma förståelse för kravet tillsammans med kunden som accepterades.

Enligt standarden uttrycktes kraven på ska-form. Ett exempel på ett krav från projektet är följande: Plocklistor ska innehålla information om artikelns namn, förråd, sektion, hylla och fack".

D.4.4 Prototyning

Under förstudien valde vi att göra LoFi-prototyper som vi tog med till kund. Vid dessa tester kunde vi se om vi var på rätt bana när det gällde design och struktur. LoFi-protptyperna utjordes av enkla pappersskisser. Ett exempel visas i figur 1.



Figur 1: LoFi-prototyper

Vi hade vid vissa tillfällen olika förslag på design och samlade då in positiva och negativa kommentaren om de olika prototyperna.

D.4.5 Användartest

Vid varje iterationsslut hölls ett möte med kund där vi demonstrerade nya features och lät kunden testa systemet. Iterationerna gjorde att vi snabbt kunde rätta till eventuella missförstånd. Vid dessa möten diskuterade vi också vad som skulle genomföras nästa iteration. Vi gick igenom vilka krav som var genomförda och vilka som skulle prioriteras till nästa iteration.

D.5 Resultat

D.6 Diskussion

D.6.1 Resultat

D.6.2 Metod

D.7 Slutsatser

D.8 Referenser

[1] Lauesen, S. (2002) Software Requirements: Styles and Techniques. Harlow: AddisonWessly.

E Automatiserade tester med Travis CI - Erik Malmberg

E.1 Inledning

Den här enskilda utredningen är en del av kandidatrapporten i kursen TDDD77 vid Linköpings universitet. Utredningen behandlar en del av utvecklingen av ett webb-baserat system för att underlätta förberedelser inför operationer på sjukhusen i Östergötland. Systemet utvecklades på uppdrag av Region Östergötland.

E.1.1 Syfte

Syftet med den här enskilda delen av kandidatarbetet är att ge insikt i hur kontinuerlig integration och automatiserade tester kan användas för att effektivisera testandet i ett projekt som använder en agil utvecklingsmetod. Speciellt ska det undersökas hur väl det går att använda Travis CI tillsammans med ramverket Jasmine.

E.1.2 Frågeställning

De frågeställningar som ska besvaras i den här enskilda delen av rapporten är:

- Hur kan man använda Travis CI tillsammans med Jasmine för att testa en webbapplikation byggd på javascript och node.js.
- Hur många tester hinner Travis CI köra på en sekund?
- Vilka typer av tester är svåra att utföra?

I svaret på den andra frågeställningen ska testfallen specifieras noggrant så att svaret inte blir tvetydigt.

E.1.3 Avgränsningar

Inga undersökningar kommer att utföras om hur andra lösningar än Travis CI kan användas för kontinuerlig integration. De testfall som kommer användas kommer uteslutande att vara skrivna med ramverket Jasmine.

E.2 Teori

Här beskrivs den teori som är nödvändig för att förstå rapporten.

E.2.1 Vattenfallsmodellen

I vattenfallsmodellen genomförs all integration och alla tester efter att implementeringen är slutförd. Om ett problem då identifieras under integrationen så är det krångligt att gå tillbaka och åtgärda problemet. Det kan leda till förseningar av projektet. Om felet som upptäcks är så allvarligt att en betydande omdesign måste ske så kommer utvecklingen i stort sett att börja om från början och man kan räkna med en hundraprocentig ökning av budgeten, både vad gäller pengar och tid [1].

E.2.2 Kontinuerlig integration och automatiserade tester

Kontinuerlig integration kan leda till att problemen identifieras tidigare i utvecklingsprocessen. Problemen blir då lättare att åtgärda. Automatiserade tester kan effektivisera testprocessen och det finns många tillgängliga lösningar för att köra automatiserade tester [2]. Några av de vanligaste är Travis CI, Codeship och Drone.

E.2.3 Travis CI

Travis CI är en webb-baserad tjänst för att köra automatiserade enhetstester och integrationstester på projekt som finns på GitHub. Travis CI är byggt på öppen källkod och är gratis att använda. Tjänsten har stöd för många olika programmeringsspråk, men det som är relevant för innehållet i den här rapporten är javascript med node.js. För att konfigurera Travis CI används filen .travis.yml som placeras i det aktuella projektets repository på GitHub.

E.2.4 Javascript

Javascript är ett programmeringsspråk som i första hand används på klientsidan på webbsidor. Javascript exekveras av webbläsaren och arbetar mot ett gränssnitt som heter Document Object Model (DOM).

E.2.5 Node.js

Node.js är en runtime environment för internetapplikationer. Det kan till exempel användas för att skapa webbservrar. Node.js är baserat på öppen källkod och det är enkelt att lägga till nya moduler för att anpassa det system man vill använda. För att lägga till nya moduler används node package manager (npm).

E.2.6 Jasmine

Jasmine är ett ramverk för testning av Javascript. Den node-modul som används är grunt-contrib-jasmine som använder task runnern Grunt för att köra testfall som skrivits med Jasmine. Grunt kunfigureras med filen Gruntfile.js.

E.3 Metod

Arbetet inleddes genom att Travis CI kopplades till projektets repository på GitHub. Kopplingen utfördes genom att administratören för repositoryn loggade in på travis-ci.org med sitt GitHub-konto och aktiverade en webhook för repositoryn.

Inställningarna för Travis CI konfigurerades med filen .travis.yml i projektets repository. Språket valdes till javascript med node.js med inställningen: *language: node_js*. Versionen av node.js valdes till version 0.10 med inställningen: *node_js: "0.10"*.

De nödvändiga node-modulerna installerades med hjälp av node package manager (npm). Grunt installerades med kommandot: *npm install -g grunt-cli*.

Grunt-contrib-jasmine installerades med kommandot: *npm install grunt-contrib-jasmine*.

Task runnern Grunt konfigurerades med filen Gruntfile.js i projektets repository. En task för Jasmine laddedes in med inställningen: *grunt.loadNpmTasks('grunt-contrib-jasmine');*. Tasken konfigurerades med följande kod i Gruntfile.js.

```
grunt.initConfig({
  pkg: grunt.file.readJSON('package.json'),
    jasmine: {
      test: {
        src: './public/js/*.js',
        options: {
          vendor: [
            'public/js/lib/jquery/jquery-2.1.1.js',
            'node_modules/jasmine-jquery/lib/jasmine-jquery.js'
          ],
          keepRunner: true,
          specs: 'test/*-spec.js',
          template: 'test/template/spec-template.tmpl'
        }
      }
    }
  });
```

Med *src: './public/js/*.js'* valdes de filer som som skulle testas. Med vendor valdes andra filer som var nödvändiga för att köra testerna. Raden *keepRunner: true* gör att filen _SpecRunner.html sparas efter att testerna körts. Filen kan sedan öppnas i en webbläsare och innehåller detaljerad information om utfallet av testerna. Med *specs: 'test/*-spec.js'* valdes de testfall som skulle köras. Alla filer som slutar med -spec.js i mappen test anses alltså vara testfall som ska köras. Eftersom Travis CI använder npm för att starta testerna så definierades testskriptet för npm med raden *"test": "grunt jasmine –verbose"* i filen package.json i projektets repository.

E.4 Resultat

E.5 Diskussion

E.5.1 Resultat

E.5.2 Metod

E.6 Slutsatser

E.7 Referenser

[1] W.W. Royce, "Managing the development of large software systems," *Proceedings of IEEE WESCON*, pp. 2, aug, 1970. [Online]. Tillgänglig (nytryckt med annan sidnumrering): `http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf`. [Hämtad april 28, 2015].

[2] O. Karlsson, "Automatiserad testning av webbapplikationer," Linköpings univ., Linköping, Sverige, 2014, pp. 43. [Online]. Tillgänglig: `http://www.diva-portal.org/smash/get/diva2:727654/FULLTEXT01.pdf`. [Hämtad april 19, 2015].

F Checkning av checklistor - Robin Andersson

F.1 Inledning

Vårt system ska innehålla olika typer av checklistor på olika webbsidor. Om flera användare är inne på samma sida samtidigt och en person checkar en checkruta så ska den checkrutan bli checkad för alla användare som är inne på den webbsidan.

Checklistan kommer att implementeras med hjälp av html, javascript och jquery samt Socket.IO.

F.1.1 Syfte

Syftet med denna del av projektet är att flera sjuksköterskor samtidigt ska kunna förbereda operationer genom att plocka olika artiklar samt förbereda operationssalen och checka av det som är utfört utan att det ska bli några konflikter med att flera sjuksköterskor plockar samma artikel eller liknande.

F.1.2 Frågeställning

- Går det att anpassa checklistan för en surfplatta medan den samtidigt innehåller information om var artiklar befinner sig samt hur många av varje artikel som behövs?
- Kommer Socket.IO vara tillräckligt snabbt för att flera personer ska kunna checka av artiklar samtidigt utan förvirring?

F.1.3 Avgränsningar

Eftersom denna del av projektet endast innehåller checkande av checklistor så saknas etiska aspekter.

F.2 Teori

Huvuddelen i implementeringen av checklistan är kommunikationen med Socket.IO som är en modul till Node.JS. Socket.IO använder sig av websockets för att kommunicera mellan front-end och back-end. Med hjälp av socket.IO så kan man skicka data från en klient till alla andra anslutna klienter och visa datan som skickades utan att någon sida behöver laddas om. Socket.IO har olika komponenter för front-end och back-end. Händelser som skickas från den ena sidan hanteras av motsvarande händelsehanterare på den andra sidan. Varje händelse identifieras med hjälp av en sträng. Det finns några färdiga händelser i socket.io exempelvis händelsen ”connectionpå serversidan fås då en klient har anslutit till socket.io. [1]

För mer information om Socket.IO se deras officiella hemsida: `http://socket.io/`

HTML står för Hypertext Markup Language och är ett sidbeskrivnings språk. [2] Jag kommer att behöva HTML till att ge en grafisk representation av

checklistan till användarna.

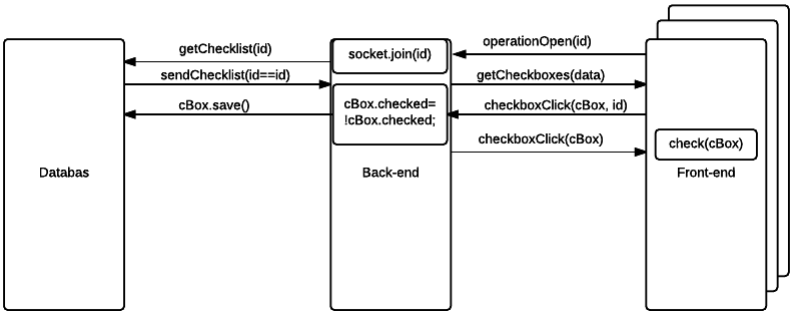
F.3 Metod

Jag började med att fundera på hur kommunikationen skulle fungera på för sätt. Jag skissade ner olika förslag på ett papper och kom på det sättet fram till hur jag skulle implementera kommunikationen. Sedan implementerade jag den och fick den att fungera. Därefter så refaktoriserade jag koden för att få den kortare och mer lättläst.

F.4 Resultat

Jag kom fram till att när en användare går in på en operationsförberedelse så kommer denne in i ett rum. Varje gång en person sedan checkar en checkbox så skickas ett Socket.IO meddelande till servern som innehåller information om vilken checkruta som ska checkas samt vilket rum checkboxen ska checkas i. Servern skickar sedan ett meddelande till det givna rummet vilken checkruta som ska checkas och alla klienter som är anslutna till det rummet checkar den givna checkrutan.

Figuren nedan visar detta flöde i ett sekvens liknande box-and-line-diagram.



När två klienter går in på en operation och en klient checkar en checkruta för första gången tar det strax under en sekund innan checkrutan checkas för den andra klienten. Därefter när någon klient checkar en checkruta så kan jag inte se någon fördröjning alls från det att en klient checkar en checkruta och en annan klient får den checkrutan checkad.

Kunden har prövat att ha flera personer inne på samma plocklista samtidigt och checka av olika artiklar. Kunden tyckte att det fungerade bra och påpekade inte någon fördröjning.

All information som krävs för plocklistorna fick plats utan att det blev för plottrigt.

F.5 Diskussion

F.5.1 Resultat

Eftersom jag endast skickar data om vilken checkruta som ska checkas till de klienter som är inne på den operation som checkrutan blev checkad på så uppdateras checkningar snabbare än att göra den enkla lösningen att bara skicka datat till alla anslutna klienter. Att det tar nästan en sekund för en checkning att uppdateras på andra klienter för första gången är långsammare än förväntat. Men eftersom det endast gäller just första artikeln och att kunden har testat checkning med flera personer samtidig utan att märka några problem så verkar detta inte vara något praktiskt problem. Att en checkning sedan kan uppdateras nästan helt utan fördrdröjning var bättre än vad jag hade förväntat mig.

F.5.2 Metod

Den metod jag använde mig av fungerade bra, men jag tror att jag skulle kunnat komma fram till samma resultat snabbare genom att göra kortare funktioner och vettigare namn redan från början istället för att göra något som funkar så snabbt som möjligt och sedan refaktorisera. För nu blev det väldigt förvirrande kod från början och jag var tvungen att sitta och tänka på vad kod jag skrivit faktiskt gjorde. Men att skissa olika förslag på ett papper först tror jag var en väldigt bra idé, det gjorde att jag fick några möjliga lösningar och sedan kunde jag överväga fördelar och nackdelar med de olika lösningarna för att sedan välja den som verkade bäst.

F.6 Slutsatser

Eftersom det fungerar bra med checkning av checklistor med hjälp av socket.io och all nödvändig information får plats utan att det upplevs som plottrigt så uppfylls syftet med denna del av projektet och min frågeställning har blivit besvarad.

F.7 Referenser

[1] Rai, Rohit. Socket. IO Real-Time Web Application Development. Birmingham: Packt Publishing Ltd., 2013.

[2] `http://tools.ietf.org/html/draft-ietf-iiir-html-00`

G Vidareutveckling av applikation för Region Östergötland - Albert Karlsson

G.1 Inledning

Denna del i rapporten behandlar vad som ska utredas och varför.

G.1.1 Syfte

Syftet med denna enskilda utredningen är att underlätta för fortsatt utveckling av webbapplikationen som projektgruppen har skapat. Alla delar i applikationen får inte användas i Region Östergötlands intranät, vilket leder till att en del måste bytas ut.

G.1.2 Frågeställning

- Vad används Keystone till i applikationen?
- Vad behövs för att ersätta Keystone?
- Vad behöver ändras för att byta ut databasen från MongoDB till MSSQL?
- Vilka moduler eller bibliotek i applikationen kräver en licens för kommersiell användning?

G.1.3 Avgränsningar

Denna rapport gäller endast för vidareutveckling för användning av Region Östergötland. Andra användare kan ha andra krav på applikationen som leder till att denna rapport är ofullständig eller felaktig.

G.2 Bakgrund

Region Östergötland ska ta över arbetet med utvecklingen av webbapplikationen efter att projektgruppen slutfört sitt arbete. För att applikationen ska kunna tas i bruk på riktigt så måste databasen bytas ut till MSSQL då Region Östergötland inte tillåter MongoDB på sina servrar. Då utvecklingen av applikationen har fortgått har Keystone fått en mindre och mindre roll i applikationen. En av Keystones största fördelar är dess enkla och smidiga administratörssystem. Detta används inte alls i applikationen längre och då databsen ska bytas till en annan typ försvinner också en annan stor del av Keystone. Detta ledde till tankar om att Keystone kanske skulle kunna bytas ut mot egenskriven kod eller mindre och mer lättförståliga moduler utan jättemycket arbete.

G.3 Teori

En beskrivning och förklaring för många av modulerna som kommer tas upp finns att läsa i avsnitt 3.

G.3.1 MSSQL

G.4 Metod

För att få en bättre förståelse för Keystones roll i applikationen så kommer först och främst Keystones tekniska dokumentation läsas.

G.5 Resultat

G.6 Diskussion

G.6.1 Resultat

G.6.2 Metod

G.7 Slutsatser

G.8 Referenser