Slutrapport

Futurum Support Application

Uppdragsgivare:
Futurum Digital

Projektgrupp:
Molly Arhammar Andersson
Miranda Hammarstedt
Jesper Landmér Pedersen
Johan Söderlund

Sammanfattning

Futurum Digital arbetar tillsammans med verksamhetsledare och affärsutvecklare för att digitalisera verksamheter och skapa nya möjligheter för produktutveckling. I takt med en ökad kundkrets har det vuxit fram ett allt större behov för en bättre administration och samverkan kring ärendehantering gentemot klienter. Med det här som bakgrund fick utvecklingsgruppen (Grupp 0) i uppgift att bygga ett skräddarsytt ärendehanteringssystem. I följande uppsats beskrivs arbetsprocessen inom gruppen, slutresultatet som levererats till kund, funderingar som uppkommit under resans gång och lärdomar som vi tar med oss i framtiden.

Innehållsförteckning

1 Inledning/bakgrund

Futurum Digital sköter idag sin kundkontakt genom flera olika kanaler, och det är svårt att hålla koll på status för varje enskilt projekt på ett överskådligt sätt. Futurum har därför önskat sig en lösning för att samla supportärenden och all kundkontakt på en och samma plats. Bakgrunden till idén om en samlad administrationspanel för kundärenden har varit ett ökat behov av en bättre översikt och smidigare hantering av ärenden för de ansvariga hos Futurum Digital och deras kundklientel.

Målet med projektet har varit att ta fram en administrationshantering för ärenden och kommunikation med kunder via mail, och det har varit vår uppgift att utveckla en användarvänlig administrationsapplikation, vars syfte är att agera kommunikations- och hanteringskanal för nuvarande och nya kunder. Applikationen ska spegla inkorgen för mail-supporten och skapa ärenden i administrationspanelen baserat på innehållet av mailen. Vid ett nytt mail så ska avsändaren stämmas av mot en lista av registrerade kunder. Applikationen ska även tillhandahålla en hantering för oregistrerade avsändare samt hantering av oväntade fel.

Projektet i sig har hanterats i sin helhet av fyra gruppmedlemmar som alla har arbetat på distans, som en del av kursen *Mjukvaruutveckling i grupp* vid Linnéuniversitetet, vårterminen 2018.

2 Syfte och mål

Syftet med projektet har varit att samla Futurums kundkommunikation på en och samma plats, och därmed förenkla strukturen för att hålla reda på vilket ärende som hanteras av vem och var i processen varje ärende befinner sig. Det här skulle underlätta Futurum Digitals arbete då mindre tid skulle behöva läggas på administrativa uppgifter.

Målet med applikationen är att:

- Ha en NodeJS-server som:
    - Reagerar på mail som skickas till en specifik Futurum-mailadress
    - Sparar dessa i en databas
    - Kommunicerar via en Websocket med en React-applikation
- Ha en React-klient som:

- Visar Futurum inkomna ärenden så snart de kommer in
- Ett mail ska skickas tillbaka ut till kunden när status på något av de inkomna ärendena ändras

Det slutgiltiga målet har dessutom varit att leverera applikationen med en god grundfunktionalitet till Futurum som de sedan själva ska kunna fortsätta utveckla i framtiden. Det här har därför genomsyrat hela utvecklingsprocessen, för att redan från början kunna lägga en grund som gör att projektet är lätt att överta i ett senare skede.

3 Projektorganisation

3.1 Arbetsmetoder

Projektet har genomförts genom en blandning av Unified Process och Kanban som projektmetodik. Unified Process kan bland annat ses i valet och kategoriseringen av milstenar, iterationer samt inkrementellt genomförande, men har i sin omfattning skalats ner till en process som påminner om SCRUM. Kanban har använts för att konkretisera uppgifter genom Githubs egna post-it system, issues och projects. Arbetet med en uppgift har då kunnat följas från idé till slutförande.

3.2 Ansvarsfördelning

Projektet har haft en projektledare, en testansvarig, en implementationsansvarig och en kund- och kravansvarig. Det har varit varje ansvarigs uppgift att strukturera arbetet kring sitt eget ansvarsområde och se till att det är gjort i tid, och allas uppgift att bidra till slutförandet av de arbetsuppgifter som uppstår under varje område.

Kund- och kravansvarig: Jesper.
Implementationsansvarig: Johan.
Testansvarig: Miranda.
Projektledare: Molly.

Vi har också delat upp dokumentationsansvaret i gruppen, med varje områdesansvarig som huvud-ansvarstagande för dokumentationen som faller under det området, enligt följande:

Kundansvarig: Ansvar för kravspecifikationen + vision.

Implementationsansvarig: Ansvar för arkitektur + teknisk dokumentation.
Testansvarig: Ansvarig för testspecifikation.
Projektledare: Ansvarig för projektplan + iterationsplanering.
Alla: Ansvariga för risklista + testrapporter.

Gällande utveckling har vi haft en viss uppdelning mellan klient och server inom gruppen, där dessa roller däremot har varit flexibla och i stort sett alla har varit inne och arbetat med bägge delar under någon del av projektet.

Klient (frontend): Miranda och Jesper.
Server (backend): Molly och Johan.

4 Genomförande

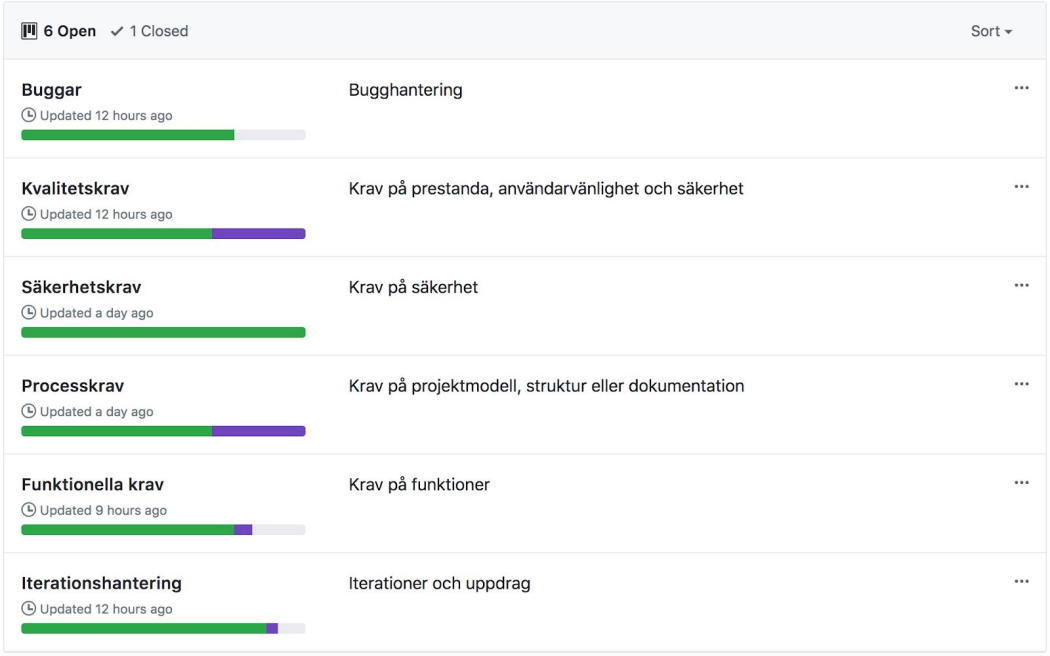
Projektets tidsresurs har varit 180 arbetstimmar per projektdeltagare, till en summa av totalt 720 timmar. Timmarna har varit tänkta att spenderas som 20 timmar per vecka, per person, under projektets gång. Projektet har genomförts som en Unified Process-process på makro-nivå, och har varit uppdelad i fyra cykler: Inception, Elaboration, Construction och Transition. Vi har dessutom inom gruppen beslutat att dessutom dela in projektet i totalt sju olika milstenar. På så sätt har vi hela tiden haft tydliga målbilder att arbeta mot inför varje iteration.

4.1 Github

Projektet har på många sätt grundat sig i användningen av Github, och i stort sett allting rörande projektet finns att hitta i dess repositorie. Härigenom har all versionshantering skötts av projektets kodbas såväl som dokumentation. Dokumentation, backlog och övriga tekniska specifikationer finns tillgängliga i den tillhörande wikin.

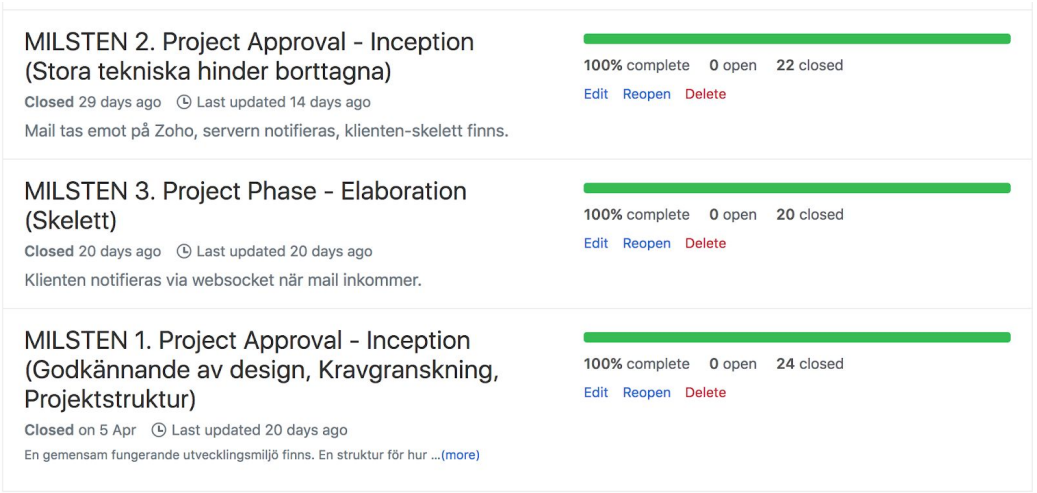
All utveckling har utgått från master-branchen där endast färdigställd eller levererad funktionalitet ligger. Utöver det här har vi arbetat med en client- och server-branch för utveckling inom respektive område. Vi har dessutom sett till att i slutet av varje iteration försöka göra en release på Github för att alltid kunna blicka tillbaka på föregående iterations arbete.

För att hantera krav från backloggen och de olika iterationerna under projektets gång har vi använt oss av Github Projects och issues. Vi har haft fem olika Projects; ett som hanterar kända buggar i applikationen, fyra som hanterar de olika typerna av krav som har definierats i kravspecifikationen (funktionella, kvalitetskrav, säkerhetskrav och processkrav), samt ett projekt som istället hanterar uppgifterna för de olika iterationerna.



Uppsatta projekt på Github där status med issues kan följas löpande.

Vi har dessutom satt upp de milstenar som vi sedan tidigare har definierat i projektplanen för issues, för att på så sätt tydligt kunna dela upp krav och uppgifter under projektets gång till olika milstenar.



Projektets tre första milstenar på Github.

4.2 Tekniker

Då kunden hade en tydlig önskan kring konsekvent skriven kod, har vi använt oss av Typescript både på klient och server kombinerat med TSLint. Tillsammans med kund valde vi även NodeJS som plattform till servern, inklusive följande bibliotek: Express för hosting och routing, IMAP för att hantera mail-trafik in från servern, Gmails API för mail-trafik ut från servern, SocketIo för Websocket-kommunikationen internt mellan klient och server, XOAuth2 för autentisering mot Gmail, Mongoose till dokumentdatabasen samt ett antal andra mindre hjälpbibliotek.

För att rendera och presentera data på klientsidan har vi valt att använda oss av ReactJS och SASS till stilsättning. Vi har även använt Webpack för att bygga ihop alla filer och Webpack-dev-server för att efterlikna kommande produktionsmiljö under utveckling och göra arbetsprocessen smidigare. I all CSS som skrivs i SASS-filerna har vi också valt att följa riktlinjerna för BEM.

4.3 Utvecklingsmiljö

Inom projektgruppen har vi suttit med två olika utvecklingsmiljöer, Mac och Linux. Det här gjorde att vi redan från början kände att det fanns ett behov av att fastställa en gemensam utvecklingsmiljö för alla, och på så sätt undvika att stöta på olika resultat hos olika gruppmedlemmar. Då vi alla sedan tidigare hade erfarenheter av Vagrant och inte ansåg att det här skulle leva upp till våra förväntningar, beslutade vi oss istället för att testa att använda Docker.

Därmed har vi under projektets gång haft en uppsatt utvecklingsmiljö i Docker med docker-compose och fyra olika containers: en för node-servern, en för React-klienten, en för den lokala MongoDB-databasen och en för Nginx som en reverse proxy. Trots att vi endast har använt Docker i utvecklingssyfte, har vi även sett över möjligheterna att produktionssätta med Docker ifall det här skulle ha varit efterfrågat av kund.

4.4 Kommunikation

4.4.1 Mot kund

Vi har under hela projektets gång haft en bra och smidig kontakt med kund. Kunden har varit tillgänglig att svara på frågor samma dag som han har kontaktats, om problemet eller

frågeställningen är liten. Mer omfattande möten har tidbokats minst några dagar i förväg. Vi har inte haft några fasta, regelbundna möten med kund men har generellt sett haft ett kundmöte varannan vecka fram till de sista tre veckorna innan slutleverans, då möten istället har legat varje vecka med regelbundna leveranser.

Alla kundmöten har skett över Google Hangouts, och övrig kommunikation via mail, telefon eller Asana, enligt kundens önskemål. Mindre problem eller frågor kring specifika krav har i huvudsak skett via Asana, där även milstolpar gentemot kund har satts upp samt mötesanteckningar att gå igenom inför kundmöten. Det har alltid varit kundansvarig som lett all kundkommunikation, även under de möten som alla i gruppen har deltagit på. Innan mötena har vi gemensamt i gruppen satt upp en agenda som sedan letts av kundansvarig under mötet, medan resterande del av gruppen varit närvarande och flikat in vid behov.

Vi har haft turen att få en väldigt engagerad kund, som redan från början gav oss väldigt tydliga riktlinjer att börja arbeta med. I och med att kunden dessutom har egen programmeringsvana och kunskap inom området så har det gjort att kommunikationen aldrig har blivit lidande på grund av begrepp eller uttryck som är utanför kundens kunskapsområde.

4.4.2 Inom utvecklingsgruppen

Under projektets gång har gruppen i första hand använt Slack som verktyg för diskussion och kommunikation. Diskussioner har i huvudsak skett i en privat grupp skriftligen, men gruppsamtal via Slack har också skett minst en gång i veckan, oftast i samband med handlednings- eller kundmöten. Det här har oftast inte varit några längre möten, men ändå effektiva och tillräckligt långa för att reda ut eventuella oklarheter och planer inför pågående iteration. Det har inte funnits några fasta dagar eller tider för när dessa avstämningar inom gruppen ska ske, utan alla medlemmar har varit så pass flexibla att de oftast har bestämts alltifrån en dag till en timme innan utsatt möte. Överlag har kommunikationen inom gruppen fungerat otroligt bra, och vi har alltid varit tydliga med varandra om när vi eventuellt inte kommer vara tillgängliga för att kunna planera därefter. Möten har alltid skett tillräckligt ofta, även om antalet har varierat mellan olika iterationer efter behov.

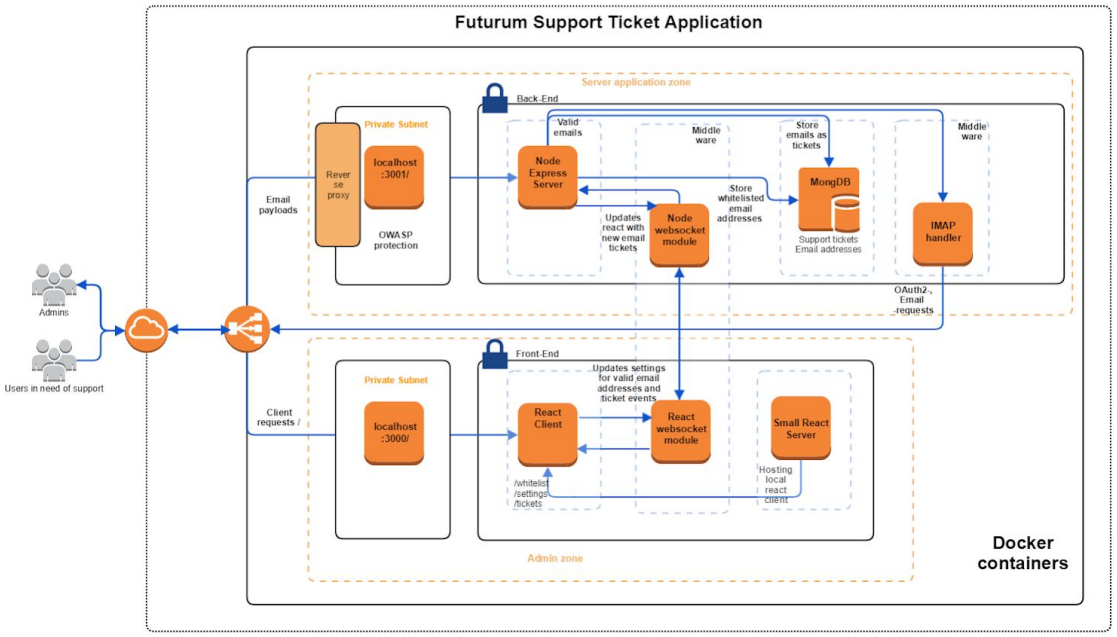
Som planeringsverktyg har Github Projects används tillsammans med issues. Därigenom har alla i gruppen löpande kunnat följa status för olika uppgifter som ska göras eller har gjorts samt haft en tydlig ansvarsfördelning för att undvika onödig förvirring.

4.4.3 Leveranser

Totalt har det gjorts fyra leveranser, inklusive slutleverans, under projektets gång, och det här har varit då faktisk implementation har levererats. Mötet innan den första leveransen och innan påbörjad implementation levererade vi istället mock-ups för arkitektur och användargränssnittet, och på så sätt fick vi bekräftat redan innan vi började skriva applikationen att det var ett utseende och en arkitektur som kund godkänt. Leveranser har sedan skett med ungefär två veckors mellanrum, vilket har varit lämpligt för att hinna utveckla så mycket som möjligt att visa inför nästa leverans. Första gången demonstrerade vi applikationen live på egen dator, och efterföljande leveranser fick kund istället själv dra ner och testa applikationen hos sig. På så sätt kunde vi tydligt se om det skulle uppstå några eventuella problem när kund själv körde projektet hos sig, och i så fall hinna felsöka och komma på en smidigare hantering inför slutleveransen.

5 Resultat

Vid projektets slutleverans har utvecklingsgruppen levererat en applikation med all den basfunktionalitet som från början efterfrågats av kund. Alla baskrav för applikationen är uppfyllda med tillhörande funktionella krav, så pass att vi även hunnit implementera en viss extrafunktionalitet, som kommer tas upp i en av följande sektioner.



Ovan visas en övergripande bild på arkitekturen för den applikation som har utvecklats. Det här är den allra första arkitekturella mockup som gjordes i projektet redan i iteration 2, vilket innebär att vissa smärre detaljer kan ha ändrats sedan dess, men som en övergripande bild är den fortfarande korrekt och demonstrerar visuellt hur klient och server fungerar tillsammans.

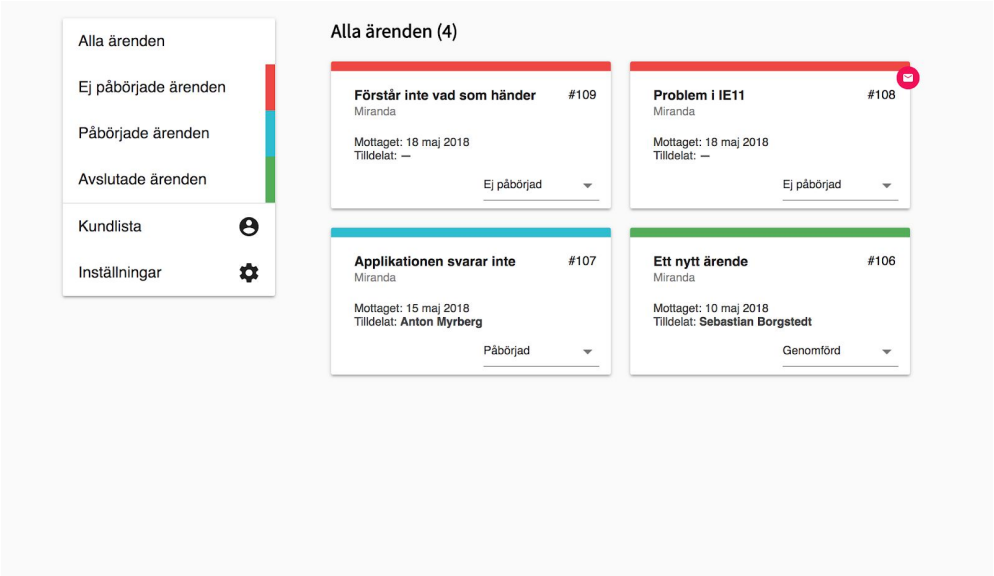
5.1 Server

På serversidan kör applikationen en NodeJS-server med Express, som i sin tur är uppkopplad mot en lokal MongoDB-databas. Servern arbetar sedan mot en IMAP-modul som är kopplad till Gmails-API för att kunna ta emot och hämta ut mail från Futurums inkorg, samt svara på ärenden direkt i applikationen. Servern och klienten kommunicerar helt och hållet genom olika uppsatta kanaler via Websockets, vilket möjliggör realtidsuppdateringar både gentemot Futurums inkorg och den egna databasen. Det är också servern som hanterar autentisering via OAuth2 mot Google, vilket är ett krav att man först gör för att kunna börja använda applikationen.

5.2 Klient

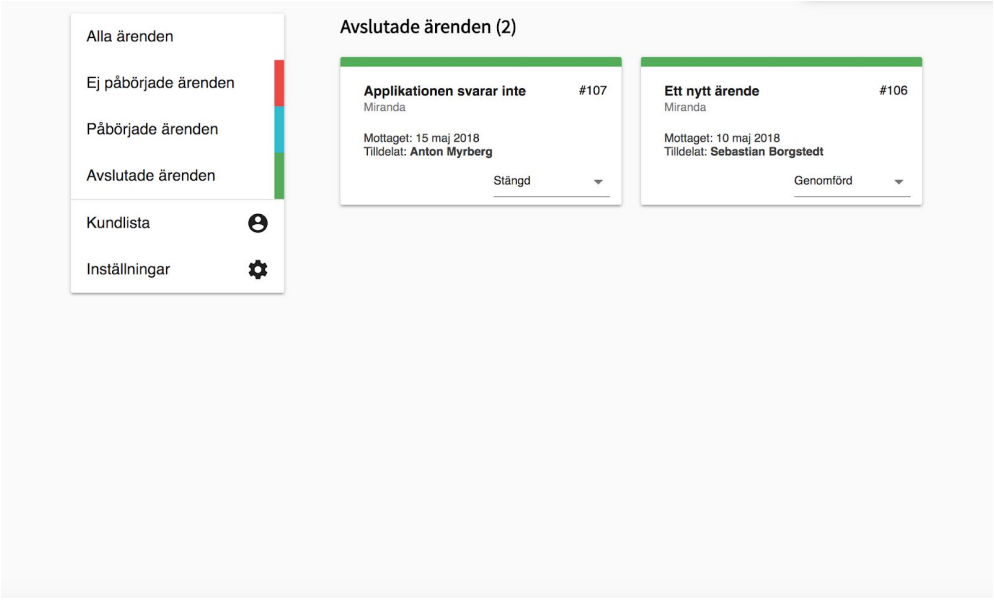
Klienten är en fristående React-applikation som fungerar som en single-page-application. Vi skapade tidigt en mock-up för att fastställa utseendet för applikationen, och nedan följer ett antal skärmdumpar på hur det slutgiltiga resultatet blev.

Ärendevy - Alla ärenden



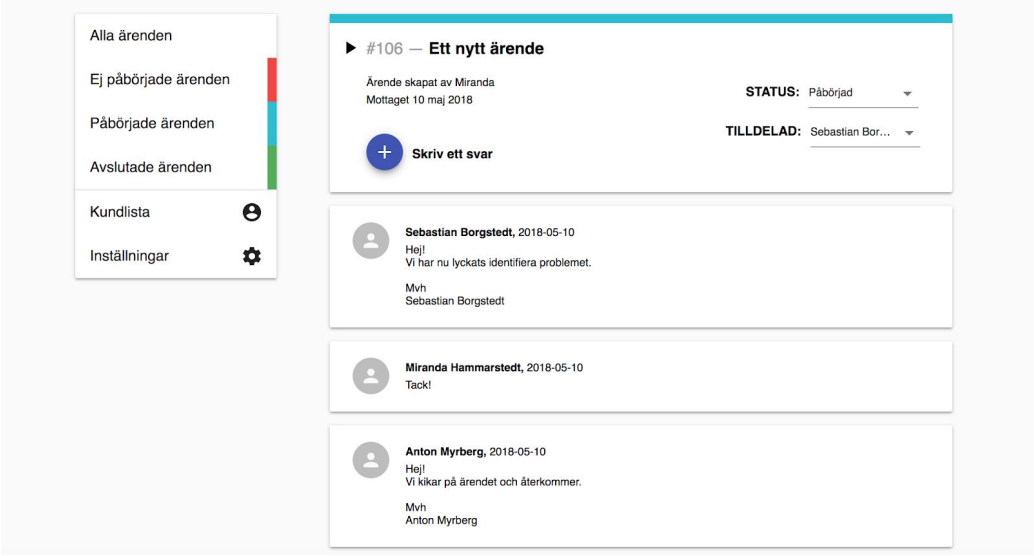
Ärendevyn är den vy som man först kommer till när man har autentiserat sig mot servern och listar alla ärenden oavsett status. Alla ärenden sorteras här efter skapelsedatum, och ärenden som innehåller nya mailsvar från kund markeras med en rund, röd notis med en mailikon, enligt bild.

Ärendevy - Filtrerade ärenden



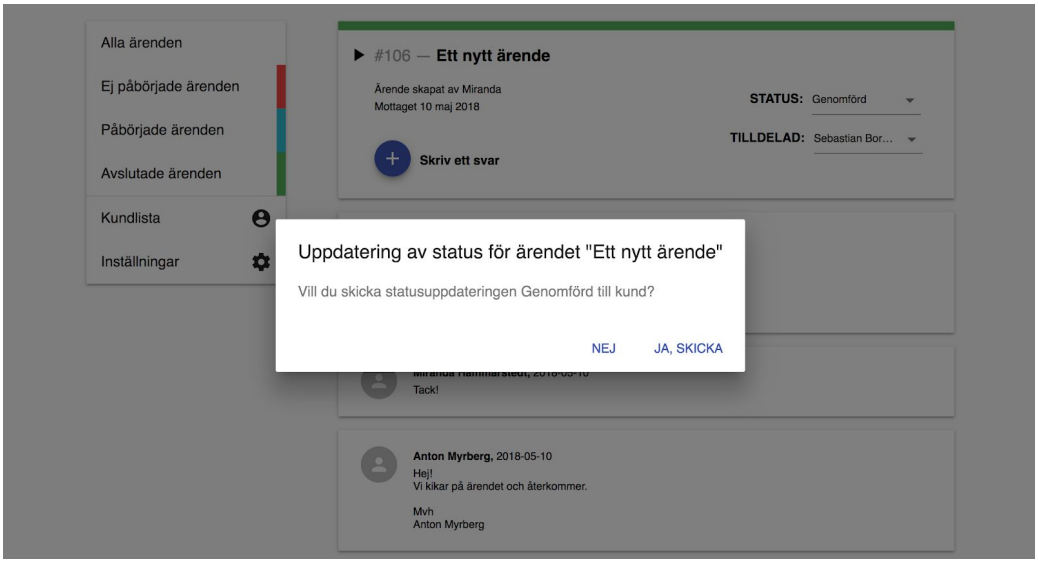
Genom att välja något av alternativen i sidomenyn markerade med färger så kan man filtrera ärenden på status. På bilden visas endast avslutade ärenden (som innefattar både status “Stängd” och “Genomförd”).

Ärendevy - Enskilt ärende



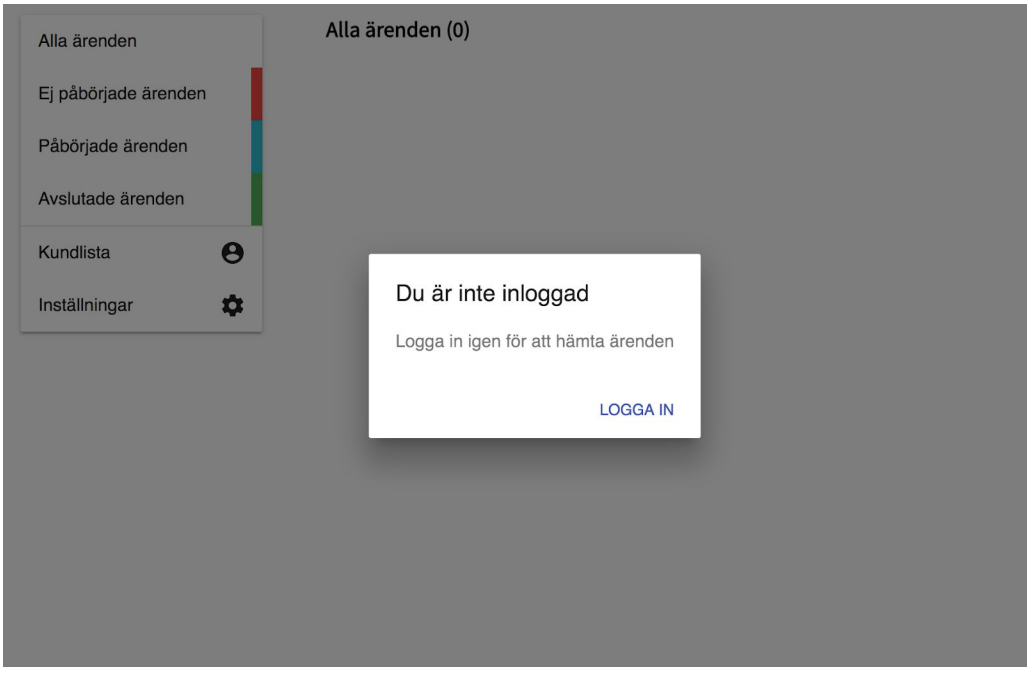
Om man från “Alla ärende”-vyn trycker på ett ärendes titel tas man vidare till det enskilda ärendet. Här hittar man all specifik information kring ärendet och även mailsvar i tråden.

Ärendevy - Enskilt ärende, uppdatering av status



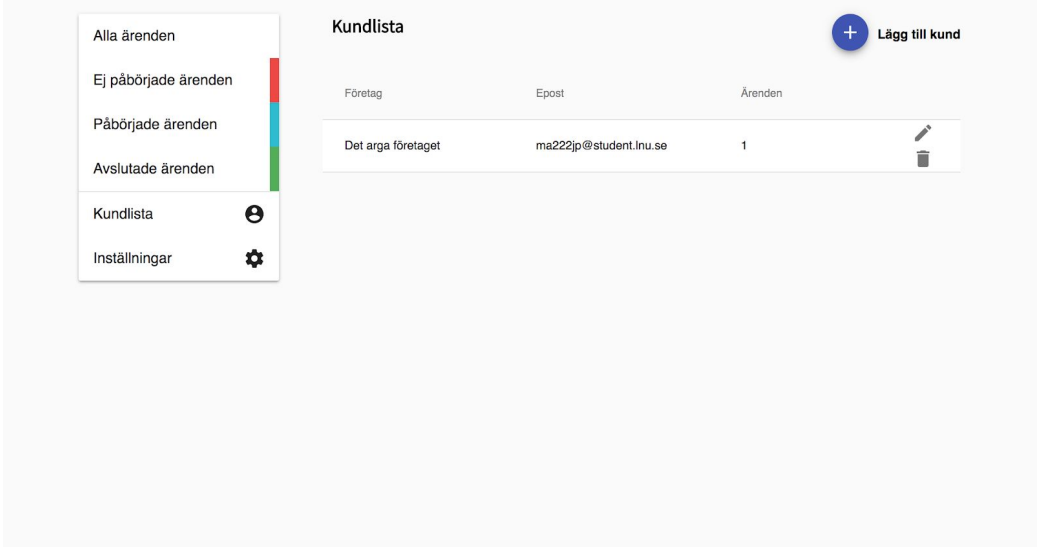
När man ändrar status på ett ärende, antingen i “Alla ärende”-vyn eller inne på det enskilda ärendet, kan man välja att även skicka uppdateringen med mail till kunden.

Ärendevy - Utloggad från servern



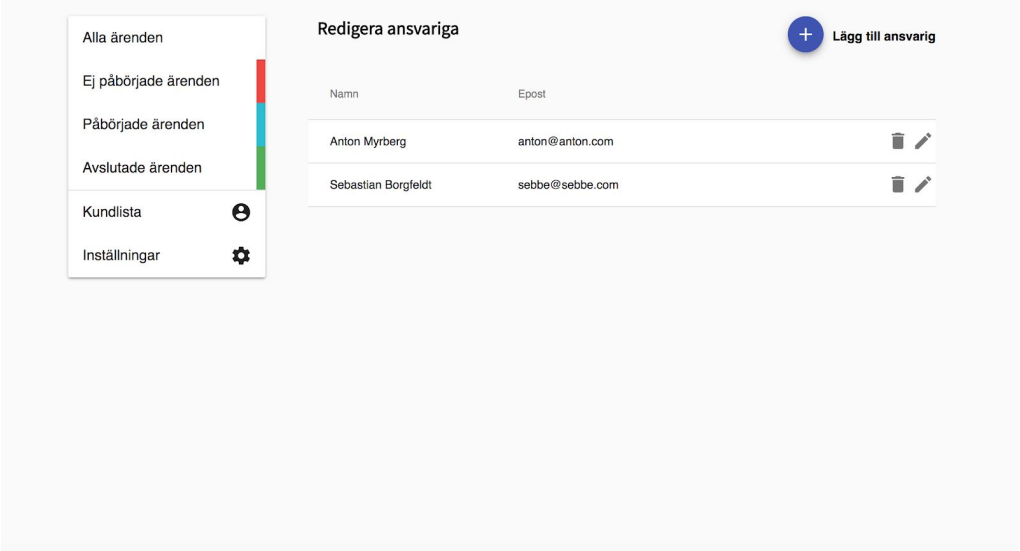
För att logga in på servern och kunna använda applikationen måste man först autentisera sig mot Gmail. Har man applikationen igång men inte är aktiv kommer man efter ett tag behöva göra om processen igen.

Kundlistan



De kunder som man kan ta emot mail från och skapa ärenden i applikationen hanteras genom en kundlista. Endast inkomna ärenden från registrerade mailadresser som finns med och tillagda i kundlistan visas på klienten.

Inställningar - Redigera ansvariga



På samma sätt som kundlistan fungerar, så kan man även lägga till och hantera ansvariga. Det här görs under inställningar och är de ansvariga som man senare kan tilldela ett ärende i applikationen.

5.3 Extrafunktionalitet

Det vi hann implementera utöver basfunktionaliteten tack vare god tidsplanering är följande:

- Möjlighet att ta bort och redigera en kund från kundlistan
- Bekräftelsemeddelanden i applikationen efter olika utförda uppgifter (till exempel ändrad tilldelning eller genomförd statusuppdatering)
- Felmeddelanden direkt på klienten från servern (till exempel vid databasfel)
- Lägga till, redigera eller ta bort ansvariga i applikationen (som används vid tilldelning)
- Notiser på ärenden i “Alla ärenden”-vyn när ett ärende har olästa mailsvar från kund

Det här är inga krav eller önskemål som har framkommit från kund utan vi har i projektgruppen själva känt att det är funktionalitet som kan vara vettig att lägga till och som är med och bidrar till en bättre användarupplevelse och administrering.

6 Avvikelser/efterkalkyl

I de allra flesta fallen har vi under projektets gång hållit oss till den plan som funnits från början, och på de punkter som vi har frångått den ursprungliga planen så har det varit förändringar som i slutändan lett till det bättre. I följande sektioner tas några av de avvikelser vi har gjort upp.

6.1 Gemensam klient/server vs. separata kontainrar

I slutresultatet landade vi med en klient/server arkitektur där klienten och servern är isolerade från varandra i olika kontainrar inuti Docker-miljön. Det här trots att den initiala idén inom gruppen var att ha en server som också hanterade klienten. Istället har vi nu en server som agerar den “riktiga” servern, och så även en mindre server på klienten som endast kör igång klientapplikationen och inte medför någon annan funktionalitet. Att vi gjorde det här valet var för att det möjliggjorde en betydligt smidigare utvecklingsprocess inom gruppen, då klient och server kunde utvecklas isolerade från varandra. Från början fick därmed båda delarna endast jobba mot testdata och inte mot varandra, vilket gjorde att vi aldrig behövde vänta in varandra i vår utveckling. Det gav dessutom en tydlig separation of concerns som vi kände gynnade projektet mer än att ha allting på en och samma server.

6.2 Socket.io vs. egen implementation av Websockets

Redan under det första kundmötet fanns det en tydlig önskan om en bra struktur för användningen av Websockets, och att den här skulle vara välplanerad. Kund sa uttryckligen “*tänk noga igenom hur ni ska hantera WebSockets*” och menade på att det kunde bli en “*hemsk soppa*” annars. De menade också på att de själva upplevt att det ibland kunde vara enklare att bygga en egen implementation kring Websockets istället för att använda ett färdigt bibliotek. Trots det här slutade det med att vi ändå använde oss av Socket.io. Det här var inte ursprungligen den plan vi hade - från början försökte vi att implementera ett betydligt mer avskalat bibliotek för Websockets, men fick däremot inte det här att fungera tillsammans med TypeScript. Det var av den här anledningen som det ändå slutade med att vi använde oss av Socket.io, då det fanns betydligt mer hjälp att hitta kring att få ihop en fungerande implementation på både klient och server med TypeScript. Vi har däremot varit väldig noggranna med att få till en bra struktur för vår Websocket-hantering i och med att kund betonade just det här lite extra, vilket vi också tycker att vi i slutändan har lyckats med.

6.3 Hantering av autentisering

I början av projektet uppgav kund att de inte ville ha någon autentisering av användare som använder applikationen, men det här ändrades dock några veckor in i projektet. Parallellt med det här implementerades till att börja med autentiseringen av vår egen applikation mot Gmails mailserver genom ett base64-kodat lösenord. Samtidigt som kunden ändrade sig om att de ville ha autentisering av användare uttryckte de också en önskan över att autentiseringen mot Gmail skulle göras över Oauth2 istället för genom lösenord. Den ursprungliga tanken var att implementera dessa delar separat - att först ha en inloggning för användaren som sköttes antingen helt av vår egen applikation eller genom en OAuth2-delegering mot användarens Google-konto, och sedan, när användaren har fått åtkomst till applikationen, be användaren att auktorisera vår applikation att använda organisationens Gmail-konto. Efter att ha arbetat med den här implementationen började den kännas både för komplex att utveckla på ett säkert sätt och onödigt omständlig för användaren som skickades genom två OAuth-flöden mot Google. Vi tog därför beslutet att slå samman autentiseringen av användaren och auktoriseringen av vår applikation, och autentisera användare genom att de får logga in på organisationens Gmail-konto och då också tillåta vår applikation att använda det. Det här förenklade inloggningsflödet både för användaren och för oss som utvecklare, med de negativa konsekvenserna att alla som använder applikationen måste ha tillgång till lösenordet till företagets mail, samt att det inte går att skilja på de olika användarna av applikationen. Vi accepterade denna konsekvens eftersom kund inte uttryckt en önskan om att kunna skilja på användare, samt att ha tillgång till lösenordet till företagets mail inte är ett orimligt scenario när man använder sig av en applikation som ger åtkomst till företagets mail.

7 Slutsats

Då kund redan från början satte upp realistiska mål kring basfunktionalitet, så motsvarade den slutliga applikationen som levererats väl kundens förväntningar. På grund av god planering och hantering av risker tidigt i projektet hann vi dessutom med att implementera en del extrafunktionalitet. Kunden har varit mycket nöjd genom hela projektet och har även varit intresserad av ett fortsatt samarbete om det hade varit aktuellt. Tack vare att vi kontinuerligt har genomfört en hel del tester från och med iteration 2, kan vi leverera ett system som vi känner stort förtroende för. Att vi dessutom har suttit på olika operativsystem och genomfört

omfattande explorativ testning varje vecka samt tryckt på att få testrapporter gjorda för varje iteration, har det här resulterat i att vi har fått ett ytterst vältestat system.

I det stora hela, förutom att ha lyckats fullfölja projektet inom de givna tidsramarna, har vi alla fått testa på att arbeta tillsammans i grupp och därmed lärt oss att skriva kod på ett tydligare sätt med syftet att den lättare ska kunna tolkas av varandra. Vi har dessutom fått testa på och lära oss nya tekniker, vilket innefattar i stort sett alla de tekniker som används, då varje gruppmedlem har haft olika mycket erfarenhet av olika områden. På så sätt har vi också lärt oss att ta hjälp och råd av varandra, samt att kunna föra diskussioner och argumentationer på ett bra sätt kring vad som lämpar sig bäst för applikationen gällande till exempel arkitekturella mönster.

8 Förslag på vidareutveckling

8.1 Inkorg för mail från oregistrerade mailadresser på klienten

Under de sista mötena vi hade med kund framkom det ett önskemål på hur man skulle kunna bättre hantera mail inkomna från oregistrerade mailadresser, som med andra ord inte finns med i kundlistan. I nuläget hanteras dessa endast genom att vidarebefordras till den mailadress som man har satt upp för det här ändamålet, och når därmed aldrig klienten. Kund önskade då att lägga till ytterligare en flik i sidomenyn där man skulle kunna se en lista med inkomna mail från oregistrerade mailadresser, som en slags inkorg för de mail som inte anses komma från godkända kunder. I den här listan skulle man direkt på klienten kunna administrera dessa och välja att godkänna mailet och lägga till avsändaren som en ny kund, alternativt avvisa mailet, och i så fall kanske skicka ett fördefinierat svar till avsändaren eller inte svara överhuvudtaget. Tyvärr kände vi inte att det här var någon funktionalitet vi skulle hunnit med att implementera när önskan las fram, men ser det som ett gott förslag för framtida vidareutveckling.

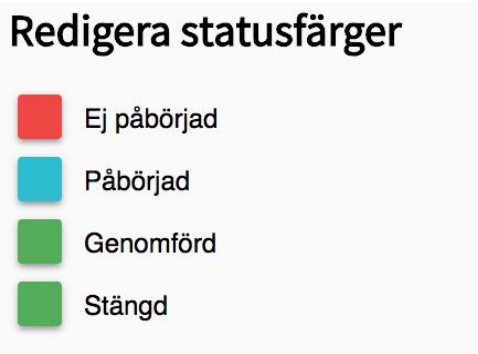
8.2 Hantera bifogade bilder/filer i mailsvar

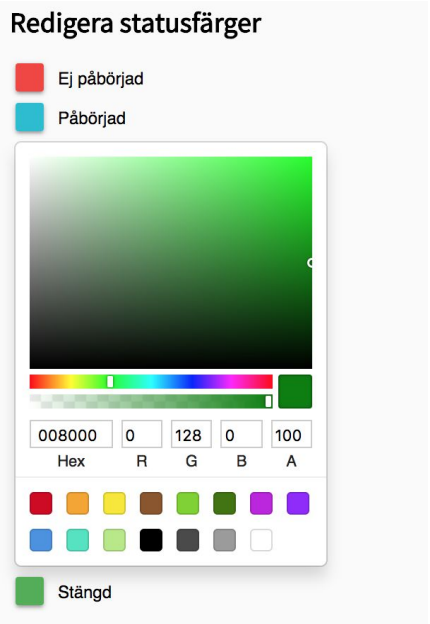
I nuläget så hanterar klient och server mailsvar till och från kund i ett ärende utan problem. Det här gäller dock endast svar i text, och skulle en kund bifoga en bild eller fil till ett svar på ett mail, skulle det här inte dyka upp i applikationen. Just nu plockar servern helt enkelt inte ut den datan från mailet, och inte heller någon påbörjad implementation för hantering av det här finns med på klienten. Även om önskemålet inte har framkommit av kund så känns det

som ett behov som skulle kunna uppstå, ifall kunder till exempel skulle behöva skicka in skärmdumpar för felsökning.

8.3 Anpassningsbara statusfärger

Ytterligare ett förslag på vidareutveckling som egentligen inte har någonting att göra med funktionalitet för ärendehanteringen utan snarare en förbättrad användarupplevelse och högre anpassningsgrad av applikationen, är att tillåta användaren att själv välja vilka statusfärger som ska användas i applikationen. I nuläget används röd (Ej påbörjat ärende), blå (Påbörjat ärende) och grön (Stängt/Genomfört ärende). Trots att vi inte hann med att implementera funktionaliteten för det här så hann vi med att bygga ett Proof of Concept som vi även har visat och kommer att lämna över till kund för eventuell vidareutveckling. Tanken är att användaren då själv ska kunna välja färg på en skala som sparas undan och sedan används genomgående i applikationen.

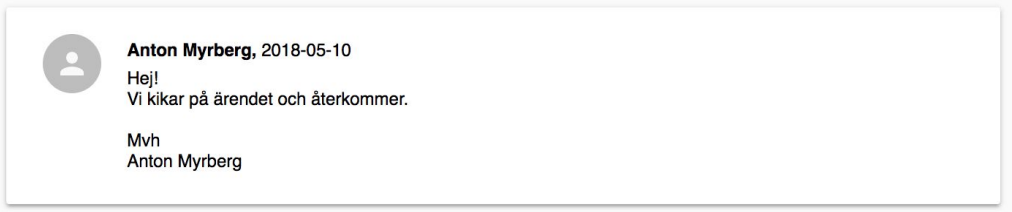




8.4 Avatarsbilder för meddelanden i ärenden

För extra förfining på klienten och en bättre användarupplevelse är ytterligare ett förslag på vidareutveckling att visa en avsändares avatarsbild bredvid meddelanden i en mailtråd i den specifika ärendevyn. I nuläget visar vi alltid en cirkel med en användarikon för att representera en användare, och därmed finns det till viss del redan stöd för det här på

klienten. En viss justering på klienten behöver ändå göras här för att det ska fungera, samt att servern också behöver ändras till att hämta ut den data som krävs för att kunna visa bilden.



8.5 Övertagande organisation

Futum Digital står ensamma som övertagande organisation, och har som målbild att kunna fortsätta utvecklingen av applikationen själva i framtiden. Därmed har alla tankar kring vidareutveckling även vidarebefordrats till Futurum och ingått i överlämnandet.

9 Förslag på förbättringar

I det stora hela är vi alla i gruppen överens om att projektet har flutit på bra och ingen av oss har upplevt några större negativa aspekter under projektets gång. Man skulle kanske kunna tänka sig att det kunde ha varit bra att planera in tätare kundmöten för att veckovis stämma av med kund och försäkra sig om att allting fortfarande är på rätt spår, men däremot är det här ingenting vi själva har känt ett behov av och det har inte heller varit en önskan från kund. Tack vare att vi har fått chansen att arbeta med en väldigt tacksam kund redan från början så har varannan vecka varit lämpligare för oss; däremot kan det vara värt att tänka på inför andra projekt som kanske är mer krävande att man kan behöva stämma av lite oftare.

Tidsfördelningen internt i gruppen har varit något flytande mellan Inception och Elaboration då den gemensamma utvecklingsmiljön var svår att få att fungera för en av medlemmarna i utvecklingsteamet som ursprungligen satt på Windows. Konsekvensen blev då att en annan gruppmedlem fick ta ett större ansvar initialt än önskvärt. När det här sedan ändrades och alla istället satt på Mac eller Linux så övergick tidsfördelningen återigen till som det varit tänkt från början.

Vi började projektet med att göra en väldigt detaljerad tidsplan i projektets wiki där vi försökte skatta exakt hur lång tid varje enskild uppgift skulle ta. Då vi snart insåg att det här

inte fungerade och utan bara tog en massa onödig tid (då vi skulle vara tvungna att förutse exakt vilka små uppgifter som skulle göras under veckan, vilket blev i stort sett omöjligt), gick vi istället över till att gruppera uppgifter mer övergripande i vår iterationsplanering, med grupper som till exempel “Applikation”, “Testning” eller “Kundhantering”. På så sätt lyckades vi hålla våra övergripande tidsskattningar betydligt bättre, och vi gick också istället över till att använda Toggls färdiga tidsrapporteringar per vecka för att föra in verklig tid som spenderats under varje iteration. Toggl-rapporterna ger detaljerade beskrivningar av all tid som rapporterats per gruppmedlem samt vilka uppgifter tiden har rapporterats på, varpå alla uppgifter är kopplade till issues på Github. Det här har gett en väldigt tydlig och smidig tidsrapportering inom gruppen, däremot har det gjort att vi har haft vår tidsplanering på två olika ställen; den uppskattade tidsplaneringen i tabeller i wikin, och den verkliga tiden i Toggl-rapporter. För gruppens arbete har vi inte upplevt det här som något större problem, däremot har vi uppfattat att det har blivit svårt att avläsa huruvida den uppskattade tiden stämmer överens eller inte med den verkliga tiden. I framtida projekt kan det därför vara värt att se över alternativ för att kunna göra det här tydligare.

9.1 Det vi tar med oss till nästa projekt

Att arbeta med Github Projects, issues och milestones har varit oerhört smidigt, framför allt när vi alla arbetar på distans, och har gjort att vi alltid har haft en tydlig arbetsfördelning och en visuell tidsplan för varje iteration. Det arbetssätt och de riktlinjer vi har satt upp för vårt arbete med branches har också fungerat otroligt bra, och även att vi har arbetat så ingående med att få en detaljerad, omfattande dokumentation, som i slutändan underlättar många andra uppgifter.

Vi har även haft en väldigt bra riskhantering och tidigt tagit tag i alla de stora tekniska riskerna. Det här var för att vi ville märka av i god tid ifall det var någon del av systemet som inte skulle kunna fungera som tänkt och i så fall hinna komma upp med en reservplan. Sedan har vi varit flitiga med explorativ testning av varandras kod, och tack vare att vi har jobbat med både klient och servern parallellt så har det här hela tiden skett kontinuerligt. Därför fick vi heller aldrig någon stor svacka mellan klient och server. Trots att vi blev rekommenderade att inte strikt dela upp ansvaret för klient och server inom gruppen, så tyckte vi själva att det föll sig naturligt och för oss har utvecklingen på det här sättet fungerat otroligt bra. I och med att vi har haft två ansvariga för varje område har man testat och tittat igenom varandras kod regelbundet, och då klient och server också varit väldigt beroende av varandra i slutskedet av applikationen så testades applikationen ofta av alla i gruppen. Med

andra ord skulle vi snarare, utifrån våra egna erfarenheter inom gruppen, kunna rekommendera att man faktiskt delar upp gruppen i klient och server trots allt, då man tvingas samarbeta mer och inte kan snöa in sig själv på att utveckla endast någon mindre, isolerad funktionalitet.

På samma sätt har skrivandet av dokumentationen aldrig upplevts som jobbig i gruppen, tack vare att vi redan från början har haft en tydlig uppdelning oss emellan. Som beskrivet i tidigare sektion kring ansvarsfördelning så har den som varit ansvarig inom ett område också haft ett visst dokumentationsansvar. Det har varit dennes ansvar att skapa de dokumenten, och sen har vi självklart hjälpts åt om så behövts. Det här har dock gjort att ingen del av dokumentationen någonsin blivit ett problem, då vi alltid har sett till att någon ansvarar för varje dokument och på så sätt har vi fått ihop en fullständig och väldigt omfattande, bra dokumentation.

10 Litteraturförslag/dokumentationshänvisning

Projekthänvisningar:

- Futurum Digital: https://futurum.digital/
- Projektdokumentation: https://github.com/1dv611-futurum-project/futurum-project/wiki
- Github Projects: https://github.com/1dv611-futurum-project/futurum-project/projects
- Demonstrationsvideo: https://youtu.be/oF7W4ur2d_U

Dokumentation/förklaringar till tekniker och verktyg:

- Docker: https://www.docker.com/
- NodeJS: https://nodejs.org/en/about/
- ReactJS: https://reactjs.org/
- Typescript: https://www.typescriptlang.org/
- IMAP: https://en.wikipedia.org/wiki/Internet_Message_Access_Protocol
- OAuth: https://developers.google.com/gmail/imap/xoauth2-protocol
- XOAuth2: https://developers.google.com/gmail/imap/xoauth2-protocol#the_sasl_xoauth2_mechanism
- Gmail-API: https://developers.google.com/gmail/api/guides/
- BEM: http://getbem.com/introduction/