Team reflection 2 - Week 4

Customer Value and Scope

Eftersom denna sektion har förändrats väldigt lite eller inget alls sedan förra reflektionen ansågs det mer relevant att fokusera på andra delar.

Social Contract and Effort

Design decisions and product structure:

Webapplikation

Kunden vill ha en klientapplikation som instruktörer kan använda på sin mobil. Här finns det flera bra alternativ. Ett alternativet är att skapa “native” applikationer för både IOS och Android. Därav krävs det två kodbaser i två olika programmeringsspråk (Swift och Java/Kotlin) vilket bidrar till negativt kundvärde då mycket utvecklingstid spenderas på att översätta funktionalitet och gränssnittsdesign. Dessutom är det två kodbaser som måste underhållas och uppdateras. Istället har teamet valt att skapa en webbapplikation eftersom det fungerar på alla enheter med en webbläsare.

React

Webbappen använder React, vilket är ett populärt ramverk/bibliotek utvecklat av Meta (Facebook). För kunden är React ett bra val eftersom det är modernt och industribeprövat.

React är ett relativt litet ramverk i jämförelse med t.ex. Google’s Angular. För viss funktionalitet används mindre funktionsspecifika bibliotek såsom react-router, react-awesome-calendar, och react-icons. Detta ger en större flexibilitet.

Eftersom applikation kommer att användas främst på mobiler krävs det att gränssnittet är anpassat för små skärmar. Dessutom krävs stora och tydliga inmatningsfält. För att göra detta enklare används Tailwind Css, vilket är ett modulärt CSS ramverk. Tailwind är skapat för att skriva flexibel och standardiserad CSS med kort utvecklingstid. Det bidrar till kundnytta eftersom det ger enhetligt design, och mer tid kan användas till mer funktionalitet.

REST

Kommunikation mellan frontend och backend använder REST-arkitekturen. Det främsta anledning är att servern och klientapplikation är löst kopplade. Det bidrar till kundvärde eftersom

det går att använda koppla flera olika klienter till samma server. Exempelvis går det att skapa en IOS app och använda samma server.

Spring-boot ramverk

Applikationen består av en java server som använder ramverket Spring-boot för att möjliggöra enklare webbkommjnikation mellan React gränssnittet, applikationens databas, och kundens databas. Till följd av att Spring-boot används får server applikationen en viss struktur, nämligen används “Inversion of Control”. Detta innebär att en bakgrundsprocess initierar alla klasser och sköter kommunikationen mellan dem. Denna strukturen möjliggör väldigt modular kod, då varje REST api som server har kan skapas som en egen, oberoende microservice.

Ytterligare möjliggör Spring en MVC (model-view-controller) struktur, där REST api:erna agerar som controller, interna “services” som model, och själva React gränssnittet som view.

Modulariteten tillför kundvärde eftersom det tillåter teamet att enkelt lägga till ny funktionalitet, liksom en ny REST api. Teamet anser att Spring-boot är ett mycket passande ramverk för applikationen och ser ingen anledning att byta. Däremot innebär ett ramverk som Spring, som skiljer sig mycket från normal Javakod, att det är en viss inlärningskurva och vad teamets kan göra de första inlärnings veckorna är mindre än när ramverkets struktur har förståtts.

Alltså behöver teamet fortsätta att lära sig använda ramverket under de kommande sprint veckorna.

Wix webbplats API

Applikationen använder den befintliga wix webbplatsen som kunden redan har. Det har skapats ett gränssnitt på wix webbplatsen som tillåter applikationens server att direkt kommunicera och hämta data från kundens databas. Kunden behöver därmed inte ändra sin befintliga databas eller datahantering för att applikationen skall fungera, liksom att manuellt lägga in nya användare i applikationens databas.

För tillfället fungerar denna wix api som skapats och det ses enbart som positivt att kunden inte behöver göra något extra arbeta för att applikationen skall fungera. Alltså anses inga förändringar behövas.

Which technical documentation you use and why (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents)

För databasen bestämde vi oss för att skapa så omfattande dokumentation som möjligt över designen. Vi skapade en domän för databasen, ER-diagram samt ett relational schema. Detta var för att vi i början av projektet ej visste exakt vilken data vi skulle behöva använda. Genom

att ha ett ER-diagram kunde hela teamet komma med synpunkter och förslag allteftersom nya upptäckter gjordes. För utvecklarna som utvecklade databasen utgjorde även relational schema viktig dokumentation då de kunde implementera databasen på ett effektivt sätt genom detta.

För att dokumentera själva koden har vi valt att använda oss av Javadoc för att möjliggöra framtida maintainability av koden. Dessutom ska mindre kodkommentarer användas för kod som vi utvecklare bedömer vara svårförståelig. Anledningen till denna typ av dokumentation är för att vi utvecklare lätt ska kunna förstå kod som andra har skrivit samt för att koden ska kunna byggas vidare på i framtiden efter det att produkten har lämnats till vår stakeholder.

I och med att vi använder Spring-boot så programmerar vi inte traditionellt objektorienterat i Java, och gör därför inga UML-diagram till vår kod. Tanken är att skapa en Readme framöver som förklarar hur man kör varje del av projektet, och hur strukturen för projektet ser ut. I nuläget har vi endast preliminära readmes för specifika delar av projektet. Vi ser dock inget värde i en väldigt formell dokumentation som exempelvis SDD(software design description), då projektet än så länge är väldigt litet och troligtvis inte kommer vara särskilt stort i sitt slutskede heller.

How you use and update your documentation throughout the sprints

Vi har lagt till JavaDoc-dokumentation till vissa klasser och metoder i serverkällkoden och några kommentarer för att förtydliga viss kod, men inte helt fullständig dokumentation för alla klasser och metoder. Det finns än så länge inte mycket dokumentation eller kommentarer i webgränssnittets källkod.

Vi vill ha åtminstone fullständig dokumentation för alla klasser och metoder som används ofta, eller där användandet inte helt och hållet framkommer av koden i sig. Vi vill också ha förtydligande kommentarer i kod där det behövs.

För att förbättra projektets dokumentation bör vi skriva fullständig JavaDoc-dokumentation i serverprojektet, samt skriva förtydligande kommentarer vid otydlig kod om det uppstår behov, till exempel om någon lagmedlem upptäcker att koden inte är tillräckligt tydlig. Vi bör också skriva dokumentation i webgränssnittskoden för att visa vad olika funktioner används för, samt förtydliga vad vissa delar av koden gör vid behov.

How you ensure code quality and enforce coding standards

För att se till att kodkvalitén försäkras har vi bestämt att man ska ha en aktör som inte är den som lagt en pull request för att kontrollera kodkvalitén. Samma aktör ser dessutom till att front-end funktionalitet är testat och att det ser bra ut. När detta är klart kan aktören godkänna en pull request.

I kommande sprints vill vi skapa ett effektivt flöde för denna process som ger värde samtidigt som processtiden hålls inom rimliga gränser. För att uppnå detta behöver man arbetserfarenhet

och komma på ett standardiserat sätt att utföra kontrollen. Olika typer av test såsom unit test kan bidra till standardiseringen av processen.

Användningen av DoD gör just nu att samtliga medlemmar kan gå tillbaka till denna definition för att arbeta enhetligt så att alla förstår vad som behöver göras. Varje Scrum-medlem hade annars kunnat ha olika definitioner av när deras “User stories” skulle vara klara vilket troligtvis skulle skapa konflikter och extra arbete för vissa medlemmar. Koden, kommentarer och arbetet blir därmed enklare genomgående. DoD säkerställer även kvalitén på arbetssätt och kod. Just nu ser laget inga problem med DoD vilket verkar ha fungerat bra i allas fall. För att se till att DoD fortsätter användas korrekt kommer vi ta upp detta på veckomötena och se till att lagmedlemmarna fortsätter arbeta enligt DoD.

Application of Scrum

The roles you have used within the team and their impact on your work

Renato (Scrum Master) har tagit på sig rollen som scrum master bra under de första sprintarna. Han har tagit ledning under sprintplanning och sprint reviews samt försökt se till att teamet förhåller sig till Scrum på rätt sätt. Genom att ställa frågor och liknande får vi team members fundera kring om det vi planerar att göra stämmer överens med arbetssättet som Scrum föreslår.

Jonas (Product Owner) har under gjorda sprints haft en bra kommunikation med kunden och har framfört synpunkter som kunden anser som värde. Vid förslag om förändringar eller implementationer har Jonas kollat med kunden innan det säkerställts.

I nuläget har lagmedlemmar tagit user stories inom samma tema dvs frontend eller backend vilket har gjort att informella roller och lag har bildats inom scrumlaget. Detta har även lett till att sub-lagen har specifik kunskap inom sina informella område. I framtiden kan detta möjligtvis vara dåligt för projektet som stort eftersom att alla inte har kunskap kring alla delar. Inom scrumlaget kommer vi i kommande sprints försöka fördela user stories så att alla får arbeta med alla delar för att tidigt sprida ut kunskapen innan projektet pågått för långt. Ytterligare ett sätt att minska påverkan av de informella rollerna är att vid sprint reviews gå igenom mer noggrant vad som skapats under sprinten och hur det fungerar för att alla ska ha möjlighet att lära sig.

The agile practices you have used and their impact on your work

Scrum board

- Hjälper att hålla saker organiserade genom att visuellt visa vad som behöver göras, vem som arbetar med vad, och hur långt olika delar är klara. I fortsättningen hoppas laget att

scrum board inte blir ostrukturerad. Åtgärden till detta är att kontrollera att allt som finns med fortfarande är relevant vid kommande sprints planeringsmöten.

User stories

- Delar upp arbetet i icke-beroende delar vilket gör att arbete kan ske samtidigt utan “conflicts”. Detta har fungerat bra även om det ibland har varit svårt att skapa dessa icke beroende user-stories. I jämförelse med tidigare veckor har arbetet genom väl indelade user-stories kunnat ske mer individuellt.

Sprint review/retrospective

- Veckovis samlas laget för sprint reviews vilket har lett till att det har varit mycket enklare att arbeta med SCRUM och olika agile practices. Genom att lagets medlemmar påminner varandra om arbetssätten som används inom dessa ökar kunskapen och arbetssättet blir enklare vecka för vecka. Att sprint review har skett under slutet av veckan gör det möjligt för oss att kunna vara retrospektiv kring hela veckans arbetsprocess vilket gynnar det agila arbetssättet.

Sprint planning

- Samlar teamet inför en sprint och ger en plan för hur vi kommer att arbeta och med vad. Vi kör även planning poker vilket förbättrar vår effor estimering som en enhetlig grupp.

DoD

- DoD har bidragit till att principer av scrum och andra krav säkerställs vid varje färdiggjord user story.

The sprint review and how it relates to your scope and customer value (Did you have a PO, if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?)

Frontenddelen av projektet är den enda delen som för tillfället kan ge en bild av appen och ge värde till stakeholdern, medan backenden och databasen senare kommer sammankopplas med denna och bidra med data och funktionalitet. Detta visades upp under sprint review för PO genom att ansvariga skärmdelade och diskuterade läget.

Under sprint review diskuteras även egenskaper av projektet i förhållande till vad stakeholdern behöver och hur vårt project scope ser ut. Som exempel diskuterades i denna veckans review om instruktörer behöver en inlogg till appen, något som inte definierats i vårt project scope.

För närvarande har inte vår DoD någon koppling till sprint reviewen, utan vår form av review för user stories hanteras under sprintens gång med GitHub pull requests.

Best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

Under projektet kommer vi använda oss av ett par verktyg, varav samtliga är nya för någon inom teamet.

Trello

Trello används som scrum board så att det har kunnats skapa ett tydligt arbetsflöde för varje user story. Teamet har delat in scrum board:et i ett par sektioner, liksom “backlog”, “in progress”, “reviewed”, “done”, etc. Alla var inte nödvändigtvis bekanta med trello i början av projektet, men i med att Trello inte är ett avancerat verktyg krävdes ingen särskild inlärning, utan alla förstod hur de fungerade så fort de började titta på webbplatsen för Trello.

Git och Github

Git och Github är respektive versionshanterings verktyget och webbhotellet som används för projektet. I och med att Git anses vara klurigt för många bestämde sig teamet för att de som var mest bekanta med Git skulle gå igenom hur man använde Git. De som var mindre bekanta fick testa sina kunskaper genom att “pusha” sin individuella reflection till Github webbhotellet. Ytterligare ansågs det vara enklare med ett grafiskt gränssnitt för de som var mindre bekanta med Github och Git, vilket underlättade användningen av verktyget.

Alla är fortfarande inte helt säkra på alla nyanser med Git men det mesta blev klart efter första sprinten. Sannolikt hjälpte det att teamet bestämde sig för en tydligt struktur i hur man använder git: varje user story har en ny branch, och man mergar inte till main utan pull requests.

Teamet har funderat på att lägga till Github build automation och branch protection, vilket är därmed något som behöver inlärning i de kommande veckorna. För detta finns det gott om dokumentation på Github:s webbplats och det har redan skapats user stories för att få igång buil automation.

Npm och Maven

Npm och Maven är viktiga verktyg för att bygga och köra respektive webbgränssnittet och javaservern. Alltså är det väsentligt att alla kan använda dessa två verktyg. I början av projektet

var det bara några som var bekanta med verktygen, och på grund av detta ordnades “workshops” den första veckan där det säkerställdes att alla kunde använda dessa verktyg.

Alla är inte helt säkra på hur man arbetar med dependencies i Maven, men de som har arbetat på javaservern verkar och fått ett grepp och kan hjälpa till andra teammedlemmar om det uppstår problem.