Reflektionsrapport

DAT256 - Software Engineering Project

Läsperiod 4 - VT 2019

Grupp Togepi

Lucas Fallqvist
Markus Pettersson
Klara Pohjanen
Mateo Raspudic

Iman Radjavi
Emil Svensson
Sofija Zdjelar
Robert Zetterlund

Introduktion

Följande rapport ämnar att presentera den slutgiltliga reflektionen från grupp Togepi i kursen DAT256 Software Engineering Project. Rapporten redogör för det mjukvaruutvecklingsprojekt som gruppen utförde samt hur Scrum användes för att strukturera arbetssättet. Först presenteras bakgrunden till projektet, sedan visionen som sattes i början av projektet och till sist de frågor som under kursens gång besvarats i gruppens gruppreflektioner.

Bakgrund

Syftet med kursen var att lösa ett öppet problem inom områdena **hållbar utveckling** och **mobilitet**. Det var upp till gruppen att själva hitta ett problem att lösa. Gruppen valde att fokusera på Chalmers campus kaféer, där det ofta kan bildas köer under rasterna eftersom alla ska köpa kaffe samtidigt. Dessutom används det i dagsläget mycket engångsmaterial till kaffemuggar och stämpelkort för köp av kaffe. Gruppen identifierade en lösning där en mobilapplikation istället kan användas för att effektivisera dagens kösystem samt reducera antalet engångsartiklar som används.

Vision

För att knyta an till **mobilitet** och ökad rörelse skapades visionen om en app som skulle minska på köbildningen runt om på campus på de ställen där det går att köpa kaffe. Med detta som bakgrund formulerades den första av projektets två epics:

"Underlätta för alla att få sitt kaffe snabbare"

Att köpa kaffe i appen hjälper till att spara tid för kunder genom att de slipper köa för att betala i kassan. Efter att en betalning slutförts genereras en QR-kod som används för att unikt identifiera varje köp som görs. Detta bidrar till att ytterligare snabba upp varje köp av kaffe eftersom kunder med hjälp av denna kod kan gå förbi kassan, bekräfta köpet inför personalen genom att skanna QR-koden för att sedan hämta sin kaffe på egen hand.

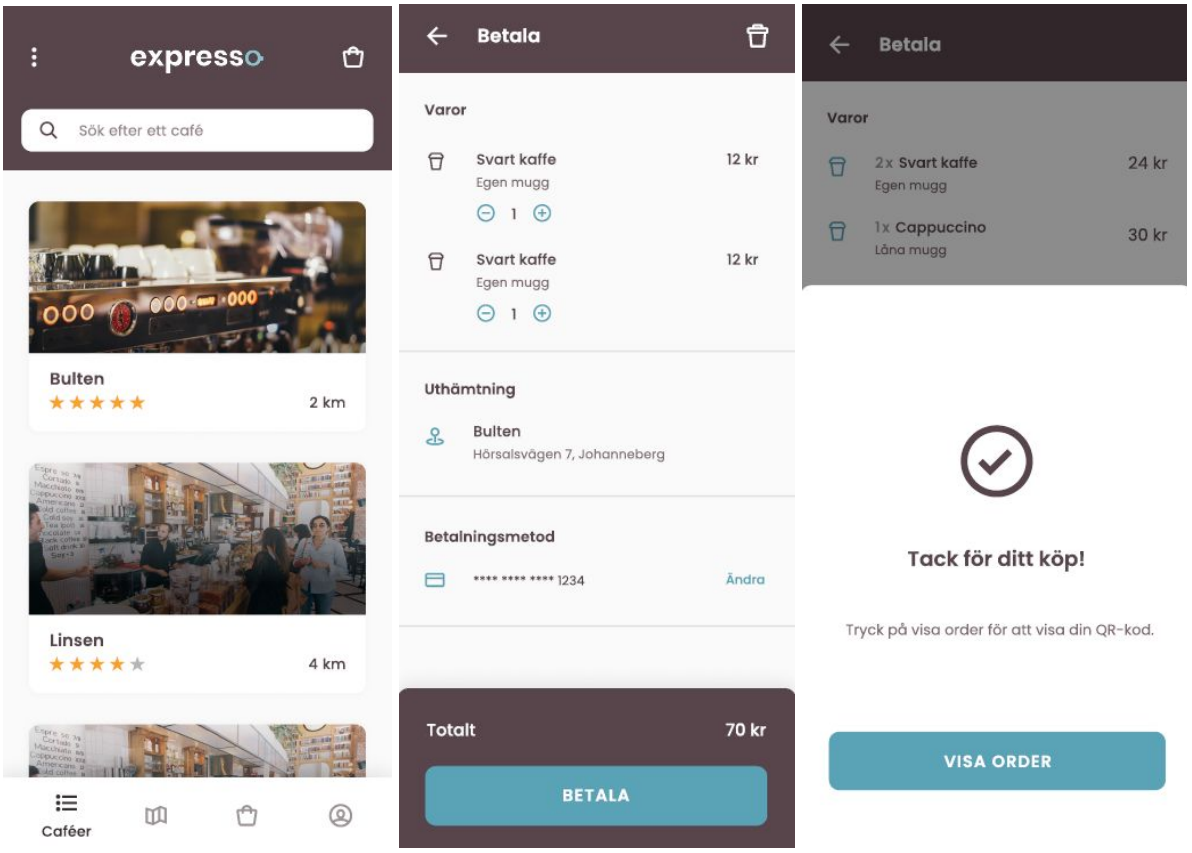
I syfte att uppfylla kursens andra målsättning, att bidra till **förbättrad hållbarhet**, valde gruppen att främst satsa på den ekologiska hållbarheten. Det var genom detta mål som projektets andra epic formulerades:

"Förbättra miljön genom att minska användandet av engångsartiklar"

Visionen med appen var således att uppmuntra användaren att ta med sig en egen mugg istället för att få en engångsmugg på kaféet, genom att erbjuda ett ekonomiskt incitament i form av en rabatt. Detta gynnar också kaféer och deras personal - dels genom att de sparar pengar på att köpa in engångsmuggar, dels eftersom hanteringen av kaffe i det fallet blir helt frikopplad från personalen vilket sparar tid även för dem. Att erbjuda kaffe till ett rabatterat pris bör således vara enkelt motiverat, speciellt med tanke på att kaffe har höga marginaler. Dessutom fanns visionen om att

med hjälp av ett digitalt system i appen möjliggöra för kaféer att låna ut muggar till kunder. Detta möjliggörs genom att spara ner information om vilka användare som väljer att låna en mugg, vilket ger kaféerna information kring hur många muggar som är utlånade och vilka personer det är som har dessa muggar.

Med ovanstående som bakgrund skapade gruppen i början av projektet en bild av vad som ansågs vara en minimum viable product (MVP). Visualisering av den slutgiltliga versionen visas i figurerna nedan i form av en mockup.



Figur 1: En första mock-up av hur gruppen ville att den slutgiltliga MVP skulle se ut, skapad i början av projektet

Reflektioner

Nedan avhandlas de olika frågor som har legat till grund för gruppens veckovisa reflektioner under projektet. Dessa frågor kommer att delas upp i 3 delar: *idag* - var vi står efter denna kurs, *mål* - var vi hoppas stå i kommande projekt och *ta sig dit* - hur vi ska ta oss till målet från där vi är idag. Den avslutande reflektionen bryter detta mönster och en allmän avslutande reflektion förs istället.

Customer Value and Scope

The chosen scope of the application under development including the priority of features and for whom you are creating value

Idag

När arbetet med att välja mobilapplikationens omfattning började under läsvecka 2 gjordes en Business Model Canvas (BMC), se figur 2. I och med det sattes också värdepropositionen upp, som syftar till att minska både köer och framförallt användandet av engångsmuggar. Därmed lades stort fokus på hållbarhet i uppgiften, medan mobilitet behandlades indirekt genom att fokusera på människor i rörelse. Detta skapar värde för kaféer och samhället i stort till följd av färre engångsmuggar, men också för användaren som slipper stå i kö på grund av ett smidigare kundflöde.

Figur 2: Business Model Canvas som visar den första formuleringen av idén och de kringliggande aktiviteterna, aktörerna och faktorer som den berörs av

I början av processen tog vi kontakt med Chalmers Konferens & Restauranger för att se hur de ställde sig till mobilapplikationen och vad som borde prioriteras. Det blev tydligt att det var hållbarhetsaspekter som var av störst intresse vilket således cementerade hållbarhetsperspektivet som prioritet för mobilapplikationens utformning. Under mötet med Chalmers Konferens & Restauranger påpekades även kostnadsfria transaktioner som en prioritet. För oss innebar det att

betalningar i appen måste kunna göras avgiftsfria. I slutet av utvecklingen tog vi kontakt med Chalmers Konferens & Restauranger för att få respons på produkten som vi hade levererat.

I början av arbetet låg vår BMC och samtal inom gruppen till grund för vilka funktioner som skulle prioriteras i mobilapplikationen. Under arbetets gång, vid sprint planning, bestämde produktägaren vilka funktioner som skulle utvecklas under sprinten i fråga. Funktioner som skulle prioriteras under sprinten prioriterades alltså av produktägaren.

Mål

Mobilapplikationens omfattning (scope) bör i framtiden vara tydligare. I samband med detta hoppas vi få mer och kontinuerlig återkoppling från intressenter på det värde vi levererar för att förstärka det inkrementella och iterativa arbetssättet. Eftersom detta projekt har utförts fristående från några externa intressenter har det inte varit självklart vad som är värdeadderande eller vad som ingår i projektets scope.

Ta sig dit

Med en mer involverad intressent kan mobilapplikationens omfattning och värde lättare definieras från start. Genom att involvera externa intressenter under sprint plannings kan en klarare bild över vilka funktioner som bör prioriteras ges. Vidare kan även mer återkoppling på det som utvecklas ges under sprint review med hjälp av en extern intressent. För detta projekt har även rollerna som Scrum Master och Produktägare har roterat bland gruppmedlemmarna. Den som har agerat produktägare har alltså ändå varit med och utvecklat appen. Om produktägaren istället hade avlägsnats från den tekniska utvecklingen skulle denne kunna vara tydlig med förväntningarna och omfattningen på produkten utan att vara subjektiv kring sin egen medverkan till den och vad som är mer eller mindre tekniskt svårt att utföra.

Syftet med en mobilapplikation bör vara tydligt och agera som en röd tråd under användarens interaktion med den. I appen som utvecklades under projektet hade följande hjälpt till för att tydligare belysa hur appen leder till ökad **mobilitet** och **hållbarhet**:

- Inledande genomgång som kommer upp första gången användaren öppnar appen och som berättar om appen för användaren
- Förbättrad interaktionsdesign med fokus på emotionell design. Efter varje köp med egen mugg kan användaren mötas av ett uppmuntrande meddelande och påminnelse om att göra något gott för miljön

The success criteria for the team in terms of what you want to achieve within the project

Idag

Som tidigare nämnt satte två Epics upp för projektet då visionen utformades. Dessa två Epics agerade även som de övergripande målen för projektet och således även som framgångskriterier för projektet. De två Epics som sattes upp var:

- Underlätta för alla att få sitt kaffe snabbare
- Förbättra miljön genom att minska användandet av engångsartiklar

Det främsta målet med vårt projekt var att leverera en Minimum Viable Product (MVP). Tanken var att den skulle vara färdig mot slutet av projektet, inför slutredovisning av mobilapplikationen. För att uppnå detta mål var det viktigt att ha våra två epics i åtanke, speciellt när det kommer till att underlätta köpet av kaffe. Det skulle vara enkelt och smidigt att beställa kaffe genom mobilapplikationen.

Under den sista sprinten blev de kvarstående user stories färdiga tidigt vilket gjorde att tid kunde läggas på att göra mobilapplikationen ännu bättre. Appen innehöll i slutändan de förväntade funktionerna utifrån våra user stories. Detta inkluderar att göra köp snabba från start till slut genom att ha så få nödvändiga vyer som möjligt för att snabba upp betalningsprocessen, samt att inloggning inte behöver göras förrän betalning ska ske. Allt detta för att underlätta för användaren att få sitt kaffe snabbare.

För att realisera vår andra epic har två val tillämpats i mobilapplikationen, lån av mugg eller ta med sin egen mugg. Genom att helt ta avstånd från användandet av engångsartiklar kan appen leda till minskad förbrukning av engångsmuggar och således minska belastningen på miljön. I praktiken skulle det innebära att de som inte har en egen mugg får låna en, för att sedan gå tillbaka med den till kaféet. Genom att även introducera en prisreducering vid användande av egen mugg, skapas ett större incitament för att välja det alternativet. Systemet med fysiska lånemuggar implementerades dock aldrig, eftersom det ligger utanför projektets scope. I slutet av projektet var dock gruppen enig om att en lyckad MVP kunde levereras. Presentationen bestod av en demonstration av hela processen från val av kaffe till genomförande av köp.

Mål

Målet i nästa projekt kommer vara att sätta mål som är konkreta under varje sprint. De mål som sattes upp i början av projektet var mer övergripande mål såsom “Underlätta för alla att få sitt kaffe snabbare“, som endast hade en konkret betydelse i slutet av projektet. Denna Epic påverkade exempelvis projektet genom att vi tog bort onödiga vyer i routingen då de gjorde processen att köpa en kaffe långsammare. De påverkade alltså gruppens arbetsprocess men det var svårt att kvantitativt mäta några framsteg mot målen.

Det är därför att eftersträva att bryta ner de övergripande målen ner till delmål eller åtminstone till något mätbart. Syftet med detta skulle vara att kunna vägleda ett framtida Scrum team genom att

det finns ett mått som går att mäta av. På detta sätta går det att ha en tydlig kommunikation med en framtida produktägare för att se ifall projektet gör framsteg som det är tänkt. I varje sprint är det isåfall enkelt att mäta hur mycket värde inkrementet från sprinten har bidragit till.

Ta sig dit

För att göra detta är det viktigt att i nästa projekt tidigt boka möte med externa intressenter för att gemensamt måla upp en bild av vad som bedöms vara framgångskriterier. Dessa kriterier bör vara kvantitativa och entydigt mätbara så att förvirring kring vad som är bra undviks i så hög utsträckning som möjligt. Det bör även, i samspel med produktägaren, diskuteras kring vad som faktiskt är värde och nytta för slutkund och forma framgångskriterierna efter detta.

För att följa upp så att varje inkrement i sprintarna faktiskt levererar det värde som är sagt bör produktägaren involveras i sprint planning, för att sätta riktning, men även sprint review för att utvärdera ifall värde faktiskt levererades. På detta sätt kan Scrum team:et utvärdera vad som behövs göras annorlunda och bättre i sprint retrospective. Beroende på tillgänglighet och projektets omfattning kan även produktägaren involveras i Daily Scrum - alternativt att Scrum Master tar ansvar för att stämma av med produktägaren under sprinten.

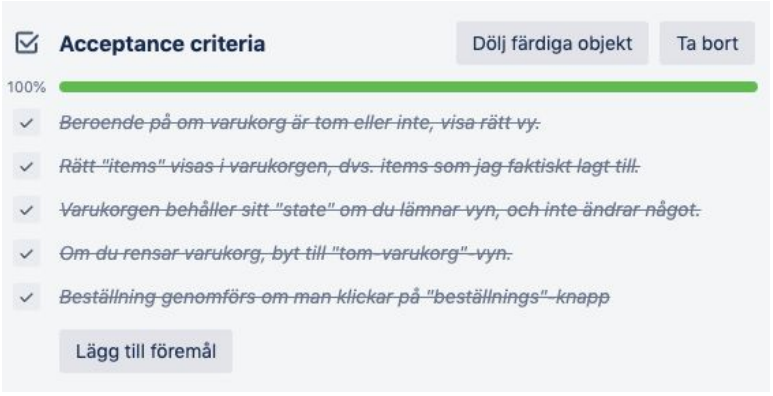
Your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value

Idag

Under uppstart var det svårt att skriva User Stories som var vertikala snarare än horisontella. Det märktes under sprint 4 som gick över påsk att många av de User Stories som avhandlades var att *visa* saker i appen snarare än att skapa funktionaliteten bakom det som visades. Den horisontella naturen belystes vid ett handledningstillfälle med Jan-Philipp där han klickade på en knapp som enbart gjorde en console.log(), vilket inte ansågs värdegivande. Utifrån handledningen förbättrades vertikaliteten av User Stories och i efterföljande sprintar försökte vi integrera arbete med både front- och backend i alla User Stories.

User Stories bröts ned i tasks i grupper om två personer. I början av projektet försökte gruppen göra det tillsammans i helgrupp. Detta var väldigt ineffektivt och ledde till att det sedan gjordes i mindre grupper om 2. Väl nedbruten estimerades User Stories genom Delfi-metoden där alla gruppmedlemmar samtidigt gav sin åsikt om vilken effort som krävdes. Efter det fick de som hade högst respektive lägst effort motivera sin åsikt, varefter samma procedur gjordes på nytt. Den här metoden fungerade bra och användes under hela projektet.

För att bestämma en User Story:s Acceptance Criteria diskuterades förväntat beteende för en User Story:s features. Det fungerade lite som ett flowchart, där beteende stakades upp, och följden av det beteende bestämdes. Ett exempel på ett acceptance criteria visas i figur 3 nedan.



Figur 3: Ett utdrag från Trello för att exemplifiera hur acceptance criterias kunde göras

En User Story:s estimering måste konkretiseras med hjälp av gruppens Velocity. Initialt valdes en fast velocity för hela gruppen 80, (det vill säga 10 per person) och därutifrån estimerades effort på olika User Stories enligt ovan. I början blev velocity ofta underskattat (eller effort överskattat) och teamet hann med mer än vad som hade satts upp från början. Under projektet hade vi ett påsklov men sträckte ut sprint 4 till två veckor. Vi behöll samma velocity vilket skapade missförstånd inom gruppen eftersom vissa hade semester utomlands medan andra förblev hemmasittande (mer detaljerat beskrivet i effort sektionen).

Mot slutet av projektet, närmare bestämt efter sprint 6 som hade en röd dag, konstaterades att vi inte kunde ha en fast velocity varje sprint. Istället fick gruppmedlemmarna säga hur mycket tid de kände att de kunde lägga kommande sprint och därefter sattes effort i hur många timmar gruppen trodde att en User Story skulle ta. Ett lämpligt beslut eftersom 5 av gruppmedlemmarna färdigställde sitt kandidatarbete och detta behövdes reflekteras i teamets velocity. Estimering i timmar var för gruppen ett mer intuitivt sätt att estimera på, men det var fortfarande inte perfekt då vi kände att gruppmedlemmarna inte visste hur mycket tid som verkligen lades på kodande (iochmed eftersökningar, oväntade bugs osv).

Mål

Det långsiktiga målet i fråga om User Stories är att bli tillräckligt bekväm med estimering av effort och sprintens velocity så att majoriteten av alla User Stories blir korrekt estimerade. Detta medför att teamet varje sprint till fullo levererar de User Stories som har arbetats på under sprinten samt att det inte finns mycket tid över när ens User Story är klar. Genom att göra det, kan det säkerställas att det varje sprint levereras värde till produktägaren.

När det kommer till att bryta ner User Stories till tasks, är ett mål att hitta en metod som fungerar för att göra detta smidigt - men ändå ha alla gruppmedlemmar på samma sida om vad som ingår i vald User Story. Att sitta alla och bryta ner User Stories var ineffektivt, men när tasks skrevs i mindre grupper blev det ibland diskussioner kring vad som faktiskt ingick - en balans mellan dessa två hade varit det optimala. För att ytterligare förbättra User Stories bör vi tydligt bemöta I.N.V.E.S.T. Criterias vid vår sprint-planning.

När det gäller att leverera värde, är målet att formulera User Stories på ett sådant sätt att det levererar värde till produktägaren. I samband med detta är ett naturligt mål att alltid leverera värde i slutet av sprinten.

Ta sig dit

Att lära sig estimera effort och uppskatta sprintens velocity är ingenting som vi kan lära oss på en dag. Det enda sättet att verkligen bli duktig på det är att arbeta enligt Scrum-metodiken och lära känna sitt team. Under sprint retrospective kan gruppen ta lärdomar om hur väl de estimerat effort och velocity för sprinten och skapa bättre förutsättningar för nästa sprint planering. Något som gjorde det svårare för oss var att alla gruppmedlemmar var på olika nivå rent erfarenhetsmässigt sett, vilket gjorde det svårt att estimera en effort som skulle gälla för samtliga gruppmedlemmar. Att utvecklas inom det området är därför något som vi tror kommer att komma med erfarenhet - men processen kan nog snabbas på, genom att tydligare diskutera hur saker som inlärning och efterforskningar ska hanteras.

Att bryta ner User Stories i tasks bör kunna fortsätta göras i par, men något som kanske kan förbättra processen är att ha någon sorts peer review med någon annan. Genom att titta på andras tasks och läsa den User Story som hör till, bör det kunna göras klarare vad som faktiskt förväntas av den. En User Story bör testas gentemot I.N.V.E.S.T Criterias genom att vid sprint-planningen som grupp reflektera över om en user-story är:

- Independent
- Negotiable
- Vertical
- Estimable
- Small
- Testable

Värde kan enklare levereras i slutet av sprinten till produktägaren om User Stories och tillhörande tasks är bättre formulerade från första början, det vill säga formulerade vertikalt istället för horisontellt. Det blir också lättare att leverera värde om en bättre estimering av effort görs, eftersom det är en förutsättning för att hinna bli klar med en User Story, men det är även viktigt att bli bättre på att i tid upptäcka om en User Story inte kommer hinna bli klar. På det sättet har gruppen större möjlighet att ändå leverera något värdefullt, även om alla tasks inte är klara.

Your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders

Idag

Gruppen använde sig av Pull Requests och peer review som acceptance tests, det vill säga för att undersöka om acceptance criteria för en User Story hade uppfyllts. Givet en Pull Request hämtades och testades koden lokalt utifrån Acceptance Criterias. Om det accepterades så merge:ades Pull Requesten in i vår dev-branch (givet att även DoD uppfylldes).

I början av projektet fanns det inget strukturerat, överenskommet arbetsflöde kring Pull Requests, utan de var bara ett krav för att överhuvudtaget kunna merge:a in kod i dev-branchen. Allteftersom projektet framskred stötte gruppen på ett par sprintar där i princip inget värde kunde levereras till produktägaren i slutet av sprinten. Med denna erfarenhet bakom oss bestämdes det att vi skulle avsätta tid för att granska Pull Requests, för att se till att koden kom in i dev-branchen innan det var dags att ha sprint review samt avstämning med sprintens produktägare. Ytterligare åtgärder som togs för att bättre utnyttja GitHubs Pull Requests för att det skulle bli lättare och mer tydligt vad som ingick i en User Story las tasks och/eller acceptance criteria in i Pull Requesten som en lista, så att den som granskade koden skulle kunna få en lättare överblick av vad som skulle finnas med. Dessutom fick den som granskade Pull Request:en avgöra om de övriga definition of done var uppfyllda (kommenterad kod med mera).

När arbetet med Pull Requests utvecklades, testade vi även att lägga till att produktägaren skulle godkänna designen på Pull Requesten innan den merge:ades in i dev-branchen. Det här ville vi göra för att säkerställa att arbetet ledde till värdeskapande för produktägaren. Det här testade vi under en sprint, men det var inget vi fick ordentlig rutin på. Istället fortsatte fokus att ligga på Pull Requests och hanteringen av dem.

Mål

Målet är att i framtiden låta acceptance tests vara en mer given och integrerad del i arbetsprocessen. Produktägaren och intressenter ska vara med och avgöra huruvida något uppfyller kriterierna snarare än gruppen, då gruppen är partisk till fördel för den utvecklade produkten i relation till målet.

Ta sig dit

För att i framtiden klara av att från början låta acceptance tests vara en mer integrerad del i arbetsprocessen bör två huvudsakliga saker göras. Dels bör en medlem i gruppen ansättas till att kontinuerligt ha ansvar för att se till att alla acceptance tests gås igenom, samt att denna person även har huvudansvar för att följa upp med produktägaren och intressenter kring ifall kriterier är avklarade.

The three KPIs you use for monitoring your progress and how you use them to improve your process

Idag

Eftersom gruppen inte började med kodning och med framsteg på appen förens sprint nummer 3 så kommer alla KPI:er att utgå från denna sprint. De sprinter som var innan dess var endast ämnade för att starta upp projektet och göra saker som var gemensamt för hela gruppen. Det går inte att mäta de uppgifter som gjorde då, varför de utelämnas från tabellerna i detta kapitel.

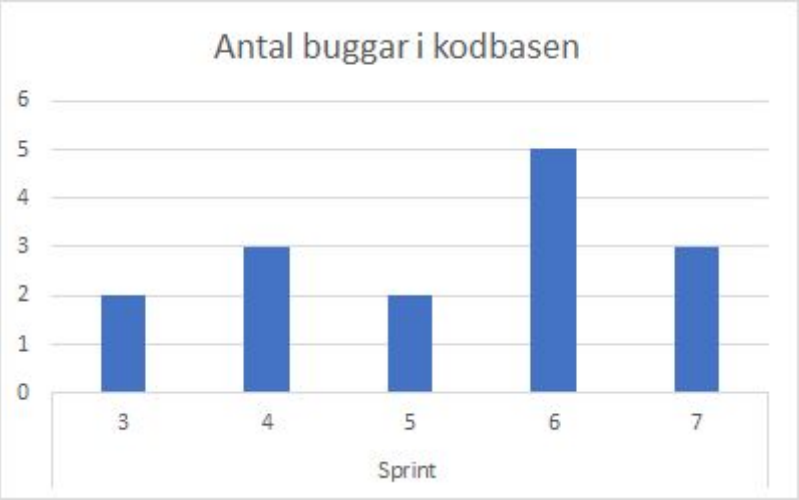
De 3 KPI:erna som vi har använt under projektets gång är:

- **Antal buggar i kodbasen**
  Antalet issues utvecklare skapat i Github.

- **Andel av User Stories som klarades av**
  Kvot mellan antalet uppskattade User Stories och de som blev klara.
- **Hur nöjd är gruppen med sprinten?**
  Uppskattat värde från gruppen för hur vi har upplevt arbetet som skett under sprinten.

Antal buggar i kodbasen

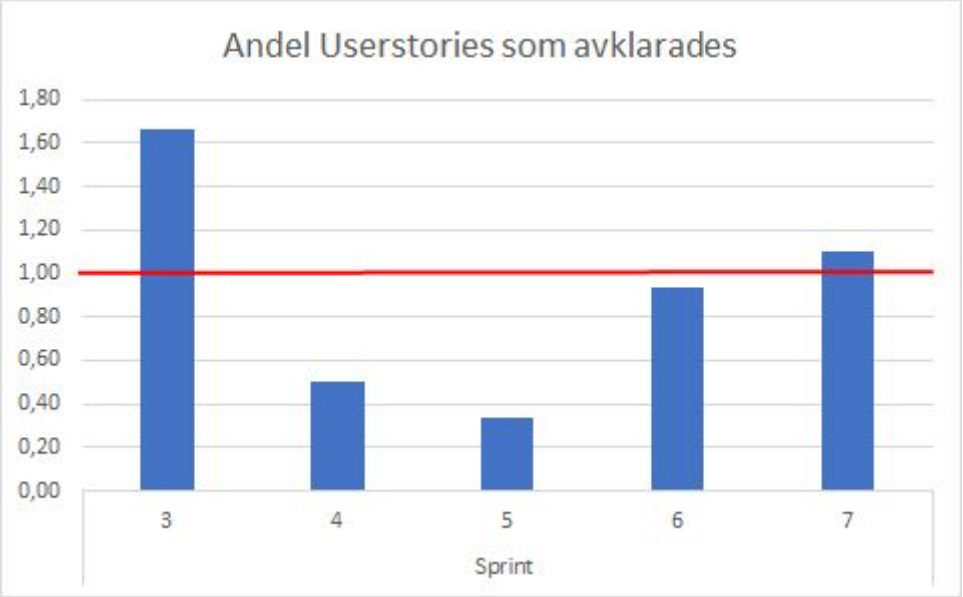
Detta mått var ett sätt för gruppen att mäta hur mycket problem det fanns i kodbasen. Detta gjordes i början av projektet genom att alla kända buggar rapporterades in genom att lägga upp en issue i GitHub vilket kollade igenom under sprint review. Tanken med detta vara att kommunicera ut till gruppen om vad som behövdes göras för att få appen och koden att fungera bättre. Målvärdet var att ha så få buggar som möjligt i hela koden och det slutgiltliga målet var självklart att få ner antalet totala buggar till 0. Detta var dock ingenting som vi lyckades med och det går att se att en stor ökning av antal buggar kom till i slutet av projektet på sprint nummer 6. Det berodde på att vi under sprinten började att jobba mer systematiskt med hur vi rapporterade in buggarna, då vi använde vår wiki om use-cases (mer om wiki senare).



Figur 4: Visualiserar resultaten från KPI:n Antal buggar i kodbasen, där 0 var det optimala fallet

Andel av User Stories som klarades av

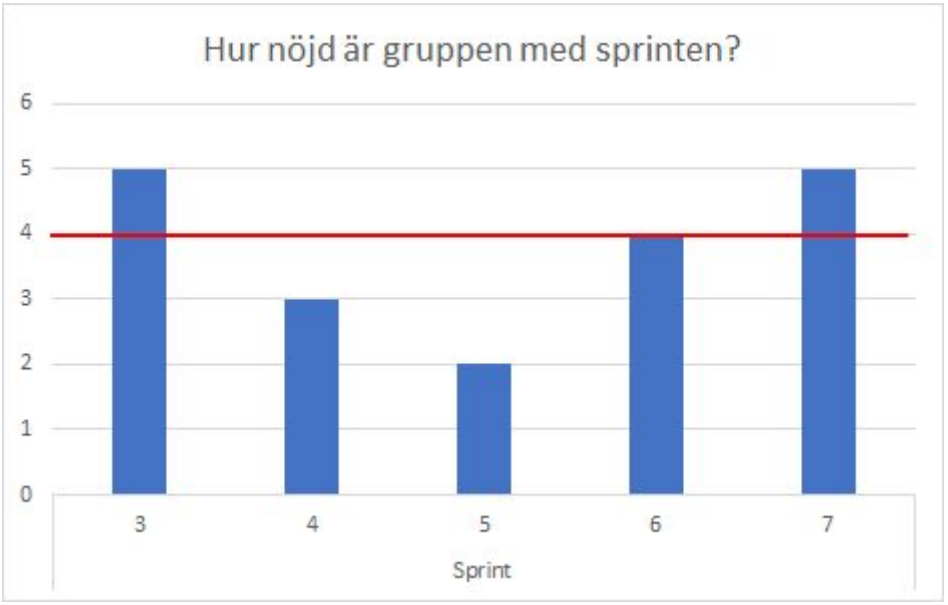
Denna KPI Skapades eftersom gruppen ville ha ett sätt att mäta hur bra och hur nära gruppen kom till att klara av de User Stories som sattes upp under sprint planning i början av sprinten. Detta mättes under projektets gång med målet att komma så nära kvoten 1 som möjligt. Det ansågs alltså inte positivt att överprestera och få en kvot som var större än 1. Om gruppen överpresterade så var detta ett tecken på att estimeringen av effort och velocity blev fel i början av sprinten under sprint planning. Ett exempel på detta var under sprint nummer 3, den första riktiga sprinten, då vi överpresterade från de mål som vi satte upp. Det motsatta, att vi kom under kvoten 1 ansågs också vara negativt på ett analogt vis som att överprestera. Detta händer under sprint nummer 4, 5 och 6 och den sista sprinten, nummer 7 lyckades vi att göra klart de User Stories som vi tog på oss.



Figur 5: Visualiserar resultatet av KPI:n Andel User Stories som avklarade där det optimala fallet är att vara precis på kvoten 1

Hur nöjd var gruppen med sprinten?

Gruppens känsla för sprinten mättes med hur nöjda vi var. KPI mäts som genomsnittet av varje gruppmedlems svar på en skala mellan 1-5. Gruppmedlemmarna har tagit i beaktning hur väl deras egna arbete har gått och hur väl de tycker att arbetet som en grupp har fungerat. Som kan ses nedan har gruppen varit nöjd de veckor som många User Stories har avklarats och mindre nöjda när det funnits mycket som inte blivit gjort.



Figur 6: Visualiserar resultatet av KPI:n Hur nöjd är gruppen med sprinten där gruppen själva satte att målet var att ligga över 4:a i nöjdhet varje vecka

Mål

Efter att ha använt de tre KPI:erna under projektet har gruppen fått insikter i vad som utgör bättre och sämre KPI:er och hur de kan användas för att mäta framsteg i ett projekt. KPI:er är bra underlag för att öppna upp för diskussioner under sprint retrospectives i syfte att förbättra arbetssättet.

Något som gruppen i efterhand har insett är en lärdom till framtida projekt som är att implementera någon typ av mätning av hur mycket tid som gruppen och gruppmedlemmar lägger på User Stories. Under projektet mätte gruppen endast andelen avklarade User Stories under sprint review i förhållande till de ansatta i sprint planning. Under sprint retrospective på sprint nummer 6 kom vi i gruppen överens om att börja logga tider för hur mycket vi satt under möten, hur mycket vi kodade och hur mycket vi satt med hantering av GitHub. Detta gjorde att arbetsmoralen gick upp något och gruppen kunde prestera bättre, men eftersom det började mätas så sent implementerades det aldrig som ett KPI.

Under projektets gång använde vi av oss utav den mjuka KPI:n “Hur nöjd är gruppen med sprinten” för att mäta hur bra gruppen tyckte att en given sprint hade gått. Måttet som det är nu mäter inte något som är direkt kopplat till värde, utan är mest en indikator för projektgruppen i sig. Något som hade varit mer optimalt hade varit att mäta hur nöjd produktägaren hade varit med resultatet av en given sprint. Detta hade varit av mest nytta om det faktiskt hade funnits en extern aktör som var produktägare under projektet. Detta hade även säkerställt att gruppen under sprint review kan vet att värde har levererats.

Något som diskuterades av gruppen var att använda sig utav ett intervall inom vilket det skulle vara godkänt för gruppen att hamna för KPI:n “Andel av User Stories som klarades av”. Det implementerades dock aldrig under projektets gång men diskuterades som lämpligt, för att inte endast tillåta för att en perfekt kvot på 1 som acceptabel. Detta skulle isåfall kunna implementeras som ett intervall med en felmarginal på +- 10% eller med andra ord inom intervall 0.9 - 1.1, vilket isåfall hade betytt att vi i de två sista sprintarna hade varit godkända enligt dessa mått. Detta är något som inför framtida projekt är att eftersträva, eftersom det inte är rimligt att anta att perfekta resultat alltid kan nås. Det kan istället resultera i sänkt arbetsmoral eftersom mål inte uppnås på kontinuerlig basis. Att använda sig utav felmarginaler bör även tas till hänsyn på KPI:n “Antal buggar i kodbasen”

Vidare anser vi att en KPI endast är så giltig som gruppen erkänner den att vara, dvs. KPI:er bör vara underlag för diskussion och konsekvent underpresterande KPI:n ska ha tydliga förbättringsåtgärder.

Ta sig dit

För att arbeta mer efter KPI:er och mätvärden behöver dessa sättas upp tidigt och vara relevanta för projektet samt värdeadderande. Genom att ha KPI:er som mäter gruppens prestationer kan dessa ligga till grund för att diskutera vad gruppen kan ta för lärdomar under sprint retrospective för att arbeta bättre nästkommande sprints.

För att kunna ta sig till målen i framtida projekt bör tider loggas ner tidigt i projektet för de aktiviteter som görs. Det gjordes väldigt sent i projektet och genom att göra detta redan från start kan KPI:er tas fram. Det blir även enklare att kunna se tillbaka i tidsloggar för att se hur mycket tid

som har lagts och hur mycket som blivit gjort i relation till detta.

För att ta sig till målet om att se till att värde levereras under varje sprint kan en produktägare tidigt involveras där den får betygsätta varje ändring i relation till hur mycket bättre denna blev från tidigare vecka. En nöjdhets-KPI skulle säkerställa att värde levereras på ett rätt sätt och förtydligar kommunikationen mellan gruppen och produktägaren.

Vidare bör även utrymme för felmarginaler tillåtas på alla KPI:er, framförallt de som mäts i "hårda" värden. Detta eftersom det inte är rimligt att gruppen alltid kommer ha perfekt kodbas eller alltid kommer att estimera perfekt gällande hur mycket Effort gruppen kan ta på sig på en given sprint under en sprint planning. För att motverka att gruppen tappar arbetsmoral till följd av oavklarade mål bör alltid en marginal tillåtas. Detta anses dock inte nödvändigt på "mjuka" KPI:er eftersom det där endast uppskattas baserat på känsla eller åsikt hur bra något har gått.

Social Contract and Effort

Your social contract, i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project

Idag

Det sociala kontraktet utformades under den första sprinten, där grundläggande regler sattes upp för vad som förväntades av gruppmedlemmarna under projektets gång. Hela gruppen samarbetade för att ta fram dokumentet och det har setts som ett levande dokument, det vill säga att det uppdaterats kontinuerligt under projektets gång.

Under mitten av projektet upplevdes problem med kommunikationen då stora delar av gruppen var bortresta över påskledigheten. Det ledde till att vissa delar av det sociala kontraktet, exempelvis möten och beslut, fick ske via gruppens Slack-kanal.

Under senare sprintar blev det uppenbart att gruppens mötestillfällen ofta var långa och att någon gruppmedlem behövde gå tidigare. Det här ledde till att vi förtydligade kontraktet ytterligare, en punkt infördes i syfte att alla medlemmar skulle informera om hur lång tid varje gruppmedlem hade för mötet - för att kunna säkerställa att de viktigaste punkterna som skulle gås igenom verkligen hann betas av ihop.

Det sociala kontraktet har uppdaterats när gruppens medlemmar har kommit överens om nya arbetssätt eller rutiner. Här har gruppen haft stor nytta av Sprint Retrospectives, speciellt när det identifierats saker som fungerat mindre bra. Exempelvis fanns det i början ingenting skrivet i det sociala kontraktet om hur gruppmedlemmar skulle arbeta kring Pull Requests. Vem låg ansvaret hos att en Pull Request var transparent, i den mån att alla gruppmedlemmar som inte arbetade på en User Story förstod syftet med den? Då sattes en rutin upp att så fort en användare blev tilldelad en User Story så öppnades en Pull Request med beskrivande namn. Den kontinuerliga utvecklingen av det sociala kontraktet har därför varit viktig för att kontinuerligt förbättra gruppens arbetssätt.

Mål

Gruppens mål är att skapa ett socialt kontrakt som bidrar till att upprätthålla ett strukturerat arbetssätt och ett enhetligt förhållningssätt till den utsatta uppgiften inom gruppen utan att begränsa individerna. Vi vill även att de regler och bestämmelser som upprättas i detta kontrakt utövas av alla gruppmedlemmar och vid eventuella förändringar så tar alla gruppmedlemmar ansvar för att göra sig införstådda med dessa ändringar. Kontraktet bör kontinuerligt följas upp och itereras på genom att lägga till och ta bort önskade respektive oönskade stycken för att reflektera gruppens optimala arbetsprocess.

Ta sig dit

Det sociala kontraktet bör vara ett levande dokument som kan anpassas efter hur gruppens medlemmar vill arbeta. Under Sprint Retrospective bör även det sociala kontraktet tas upp för diskussion för att se om gruppen anser att innehållet fortfarande är relevant för gruppens arbetssätt. Det är då även viktigt att göra förändringar där gruppen anser att de inte följer kontraktet baserat på hela gruppens åsikter. Genom att lyfta det sociala kontraktet får medlemmarna i gruppen själva en möjlighet att återkoppla till vad gruppen tidigare satt för regler om arbetet. Under Sprint Retrospective skapas då även en möjlighet att se vad gruppen kan arbeta bättre med i det sociala kontraktet.

När det uppstår problem i arbetsprocessen kan det vara bra att gå tillbaka till det sociala kontraktet och uppdatera det, detta bör tilldelas till en specifik medlem och lämpligast är förmodligen att det faller under Scrum masterns arbetsuppgifter.

För att kontinuerligt påminna om saker som gruppmedlemmar glömmer på daglig basis kan det vara ett bra tillfälle för Scrum mastern att under daily Scrums lyfta upp de saker som inte följs upp, t.ex. som att hålla en tydlig kommunikation kring vad varje gruppmedlem jobbar med inom gruppen för tillfället och uppmärksamma nya tillskott i kontraktet.

Vi ser även att det i idealfallet förs extensiv dokumentation kring ändringar i kontraktet. I detta projekt föll detta lite mellan stolarna, vilket innebar att dokumentationen i praktikten var git-loggen (som ofta undgick majoriteten av gruppmedlemmarna). Till nästa projekt föreslår vi därför att tillfället då ändringar till kontraktet sker får sin egna avsatta tid på mötesagendan och att det därmed dokumenteras i mötesprotokollet.

The time you have spent on the course and how it relates to what you delivered

Idag

Genom hela projektet har gruppen haft målet att hinna klart med innevarande sprint-leverabler. För att kunna göra det på ett bra sätt har vi haft två stycken veckovisa gruppmöten. De har tillsammans tagit ganska mycket av gruppens tid och varit grundläggande för att kunna genomföra sprint planning, review samt retrospective för varje sprint.

Från början hade vi sprint planning på måndagar och sprint review samt retrospective på fredagar, men eftersom gruppmedlemmarna gärna ville kunna arbeta på helgen flyttades även sprint review samt retrospective till måndagar efter sprint 4. Vi fortsatte ha gruppmöten på fredagar, som ett extra tillfälle att sitta tillsammans och arbeta.

Sprint 4, som gick över påsk visade stora skillnader i hur mycket tid de olika gruppmedlemmarna la på kursen, mycket på grund av variationer i hur mycket tid de hade. Gruppen hade inte heller kommit överens om vilka dagar som ansågs vara lediga, vilket gjorde att det blev svårt att sätta en standard för hur mycket arbete som förväntades. De User Stories som var påbörjade hann dock klart och mer därtill, vilket talar för att de var underestimerade i effort.

Senare under sprinten uppkom motsatt problem. Gruppmedlemmarna la mycket tid men lyckades inte leverera de User Stories som hade valts ut under sprint planning, vilket bottnar i svårigheterna kring rutiner på granskningen av Pull Requests. Det här ledde till att inget värde levererades till produktägaren, trots all tid som lagts på projektet.

Gruppen började med tidrapportering i slutet av projektet. Genom att koppla effort till antal timmar som arbetades, tvingades vi att arbeta fokuserat för att få tid och effort att stämma överens med varandra.

Mål

I framtida projekt tar gruppen med sig lärdomen att det är viktigt att projektets scope planeras utifrån den tid som är möjlig att lägga på projektet. Gruppen ska även komma överens om hur flexibla medlemmar får vara med arbetstiden under varje sprint.

Ta sig dit

Genom att vara tydligare i det sociala kontraktet kan bättre riktlinjer skapas för gruppen. Att till exempel sätta en gräns som “minst 10 h per sprint” och sedan följa upp medlemmars arbetstid med KPI:er för att se om detta efterlevs. Detta kan möjliggöras genom att tidsrapportering implementeras från början av projektet och alltid efterföljs.

För att säkerställa att alla i gruppen lägger den tid som krävs och för att se så att ingenting oväntat sker under sprinten kan det under Daily Scrum vara bra att se så att alla är med. Det bör även ligga på Scrum Masters ansvar att se till att alla gruppmedlemmar förstår vad som krävs av de har möjlighet att lägga den tid som förväntas.

Design decisions and product structure

How your design decisions support customer value

Idag

Från början valdes React Native som utvecklingsmiljö för att ge kundvärde i form av en mobilapplikation som stöds av både Android och iOS-enheter. Detta är av värde för kunden då det

inte stänger ute användare beroende på vilken mobil de använder sig av. Det fundamentala med React är att skriva deklarativ kod i form av komponenter som bygger ihop ett helhetssystem. Skapandet av komponenter möjliggör återanvändning i flera delar av mobilapplikationen samt att komponenterna enkelt kan utökas vid behov. Detta gör att fler funktioner snabbt kan läggas till i efterhand för att ge ett högre kundvärde.

Vidare valdes Google Firebase att integreras i vår mobilapplikation för att ge förhöjd kundupplevelse genom att enkelt och säkert kunna registrera sig och logga in. Det möjliggjorde att användaren kunde lagra information om användare, som t.ex. kortnummer, för att slippa skriva in detta varenda gång någon betalar. Det valdes även tidigt att implementera Redux för att kunna lagra globala states i hela mobilapplikationen, för att kunna bland annat visa antal varor i varukorgen och möjligheten att ta sig till varukorgen från alla vyer i appen. Detta innebär även att vi kan erbjuda kunden möjlighet att beställa kaffe från olika vyer, vilket ger värde. Med denna möjlighet att manipulera kundvagnen överallt med Redux har vi valt att använda React Navigation. React Navigation är standardbiblioteket för navigering på mobilapplikationer i React Native. Navigering ger värde till kunden, eftersom vi då kan visa fler kaféer och fokusera informationsflödet.

Figma användes tidigt i arbetet för att designa och strukturera olika vyer i mobilapplikationen i form av mockups. Det gjordes innan vyer implementerades i själva appen då vi som utvecklare kunde slippa tänka på designen under kodskrivningen. Detta gav ett stort kundvärde då mobilapplikationen kunde utvecklas utifrån dessa mockups och gav i slutändan en användarvänlig och estetiskt tilltalande mobilapplikation.

Vi skapade ett eget API för att få mer flexibilitet kring vår data, vilket bland annat skulle kunna ge kaféägare möjligheten att manipulera sitt utbud av kaffe själva. API:et utvecklades inkrementellt under hela arbetet allt eftersom produktägaren ställde nya krav på produkten. Exempelvis fanns det i början bara ett API för att hämta hårdkodad information om olika kaféer. Detta underlättade för oss eftersom vi i ett senare skede förväntades hosta databasen på en server. Genom att ha tagit fram API:et gradvis kunde övergången ske utan att driften av mobilapplikationen störtes. Vid senare sprints hostades vår backend på Heroku där all information om kaféer lagrades. Denna information hämtades då i vår app genom att använda vårt API som ett mellanlager. Backenden lagrade även data om kundernas tidigare köp, kårkortsnummer och aktiva ordrar som fortfarande gäller och kan scannas.

Figur 7 nedan visar hur datan är strukturerad i våra Firebase-databaser. Till vänster ses ett kvittoobjekt, och till höger syns användarobjekt som innehåller kvittot för en aktiv order samt användarens lagrade kortnummer.

Figur 7: Visar hur data strukturerades i Firebase.

Mål

I framtida projekt kommer vi huvudsakligen ta med oss användningen av Figma för skapandet av mockups då det hjälpte väldigt mycket under utvecklingsfasen. Det möjliggjorde att gruppmedlemmarna under utvecklingen inte behövde tänka på hur komponenter ska struktureras upp designmässigt, utan det var bara att följa en given mall. I vårt projekt användes inte mockups till alla vyer i mobilapplikationen, vilket är något vi i framtida projekt kommer sätta stort fokus på. Det är även viktigt att tidigt i arbetsprocessen identifiera teknologier som behöver användas och hur dessa ska samarbeta tillsammans för att bygga ihop ett helhetssystem.

Ta sig dit

För att nästa projekt ska realiseras på ett effektivt sätt så ska mockups och planering för samtliga vyer skapas tidigt i arbetet för att underlätta utvecklingsstadiet. I början av arbetsprocessen bör även projektets teknologier och utvecklingsmiljöer bestämmas utifrån vad som kommer att behövas och hur mycket kundvärde som ges av dessa. Efter detta projekt har många av gruppmedlemmarna fått en större inblick i hur en mobilapplikation kan utvecklas på ett framgångsrikt sätt, vilket innebär att det kan bli en bättre diskussion kring valet av tekniker i kommande projekt.

Which technical documentation you use and why

Idag

Ett tekniskt dokument författades tidigt i projektet. Detta underlättade för gruppen då det var en samlingsplats av information gällande projektet. I dokumentet samlades installationsguider, länkar och information om vår stack. Under projektets gång uppdaterades dokumentet med information om datamodeller och om vårt API. Det tekniska dokumentet finns att läsa på GitHub.

Vid ett senare skede insåg vi att dokumentationen kändes statisk eftersom den hostades genom en gist (som endast en team-medlem kunde ändra). Då flyttade vi all dokumentation till projektets Wiki på GitHub. Under projektets gång samlades mer information på Wiki:n. Exempelvis noterades det inom gruppen vid sprint-planning att User Stories bedömdes olika stora beroende på vem som bedömde den, vilket belyser en potentiell obalans av kunskap. Detta belystes eftersom vi använde oss av Delfi-metoden, se fråga angående estimering. Det som vi genom Wiki:n försökt åstadkomma är transparens och en fördelning av kunskap, dels för att bedömningen av User Stories ska bli enklare, men också för att undvika nischad kompetens.

Annan teknisk dokumentation, i form av en mockup, skedde genom Figma (mer om figma i tidigare avsnitt ovan). Figman tillät gruppen att ha en utgångspunkt vid designande av vyer. Detta gjorde det enklare att visuellt presentera sin idé istället för att implementera den i kod.

Beroende på ett projekts scope finns olika nivåer av dokumentation. Projektet krävde grundläggande kunskaper inom React, något som erbjöd teamet att länka till simpel dokumentation på nätet. Länkarna samlades på vår Slack.

Wiki:n bestod i slutet av projektet av följande sidor:

- API och datamodeller
    - Information om vår backend och hur kaffe och ordrar representeras i appen
- Färgschema för appen
    - För att ha en enhetlig app samlar vi alla färger som används, allt från bakgrundsfärg, till detaljfärger
- Ikoner
    - En guide om var ikoner hämtas ifrån, och hur, eftersom vi då undviker att använda ikoner som vi saknar rättigheter till
- Firebase, kvitton och Heroku
    - Information om hur vi hanterar inloggningsuppgifter genom Firebase, kvittons datamodell i backenden och information om det som är deployat genom heroku
- Mockup och wireframing på Figma
    - En länk till vår mockup
- Tekniskt dokument
    - En stor ansamling av information kring appen, bra för installering eller mindre ändringar
- Use Cases

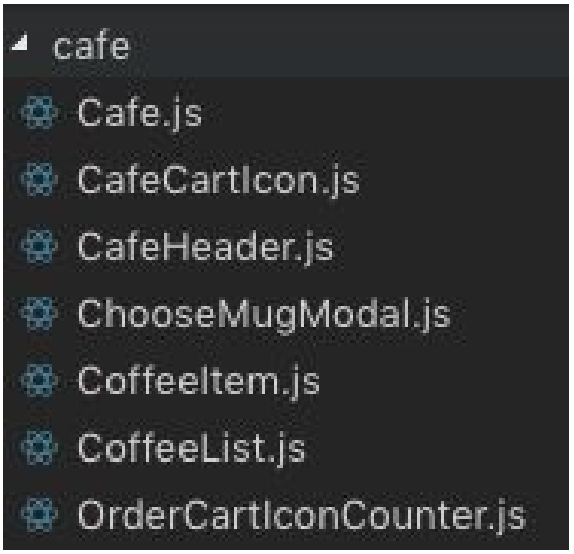
- En samling av user-cases som beskriver funktionalitet och navigering/funktioner som måste funka innan merge in till dev - här har gruppen även en möjlighet att testa “actual behaviour” gentemot “expected behaviour”

Vidare användes dokumentation i koden för att förklara vid behov. Längst upp i en komponent skrevs information om den, se exempel i figur 8 nedan.

```
6   /**
7    * @file This is a presenetational view that shows the user the current state of the cart.
8    *
9    * Since cart does not have any properties containing information about the amount of coffees
10   * or total price of all coffees, we need to calculate them using some functions related to
11   * the state of the cart. Further reading in {@link cartFunctions.js}
12   *
13   * @param cart The current cart object in redux.
14   *
15   */
16
17  const CafeCartIcon = ({ cart }) => {
```

Figur 8: Ett utdrag från kodbasen för att visa hur kommentarer utnyttjades under projektets gång

Denna typ av dokumentation var väldigt bra, då den beskriver när/hur/var den används och vad för inparameter den har, något som inte fanns för alla komponenter vilket gjorde det komplicerat vid felsökning men även vid orientering. Orienteringen bland filer försöktes förbättras genom väldöpta mappar och filnamn. I figuren 9 nedan visas mappen för Café-Vyn.



Figur 9: Ett utdrag från kodbasen för att visa hur mappar strukturerades

Vi använde även små kommentarer för att förtydliga vid behov, se figur 10 nedan.

Figur 10: Ett utdrag från kodbasen för att visa hur mindre kommentarer användes för att förtydliga löpande

Mål

Vi planerar att i framtida projekt försöka använda Wiki mer, på ett mer effektivt sätt. Wiki:n bör utvecklas i takt med kodbasen istället för efter behov och innehålla relevant information. Relevant information för oss innebär information som har enkla svar, exempelvis vilken färg som används, hur redux-state kan utnyttjas i komponenter och liknande. Ytterligare relevant information är samling av länkar, installationsguider och test-guider. Målet är i viss utsträckning att undvika meningen "kommer du ihåg när jag berättade?" och ersätta det med möjligheten att söka upp svaret på wikin.

Optimalt gällande kod-orientering vore nog att skapa ett UML-diagram och behålla en liknande approach till mapp-strukturering. Visualisering med modifierad UML som gör så att man slipper leta upp vilka inparametrar en komponent förväntar sig. Genom användning av en övergripande @file text (som beskriver vad komponenten gör och vad som använder den), och en kort @param text som beskriver vilka inparametrar som komponenten kräver blir utvecklingen lättare.

Ta sig dit

I framtiden bör ett Wiki-entry skapas/uppdateras vid införande av ny funktionalitet, för att jämna ut kunskapsfördelningen och dela kunskap inom gruppen.

Genom att tydligare dokumentera hur varje komponent används med @pre och @param samt var varje komponent används skulle koden bli enklare att förstå när kodbasen växer. Att tidigt sätta upp en standard för vad "tydlig dokumentation" innebär kan koden blir bättre dokumenterad. För att trycka på dokumentationen kan man vid slutet av en sprint se de filer som ändrats och kräva att dokumentationen avspeglar ny funktionalitet, så att man undviker daterade kommentarer (visuell code smell).

How you use and update your documentation throughout the sprints

Idag

Om en gruppmedlem behövt information om något har de kunnat söka efter det på wikin, men även kunnat reflektera över informationen för att se om det behöver finnas på wikin. Det har inte funnits ett systematiskt sätt att uppdatera wikin, utan den har vuxit efter behov.

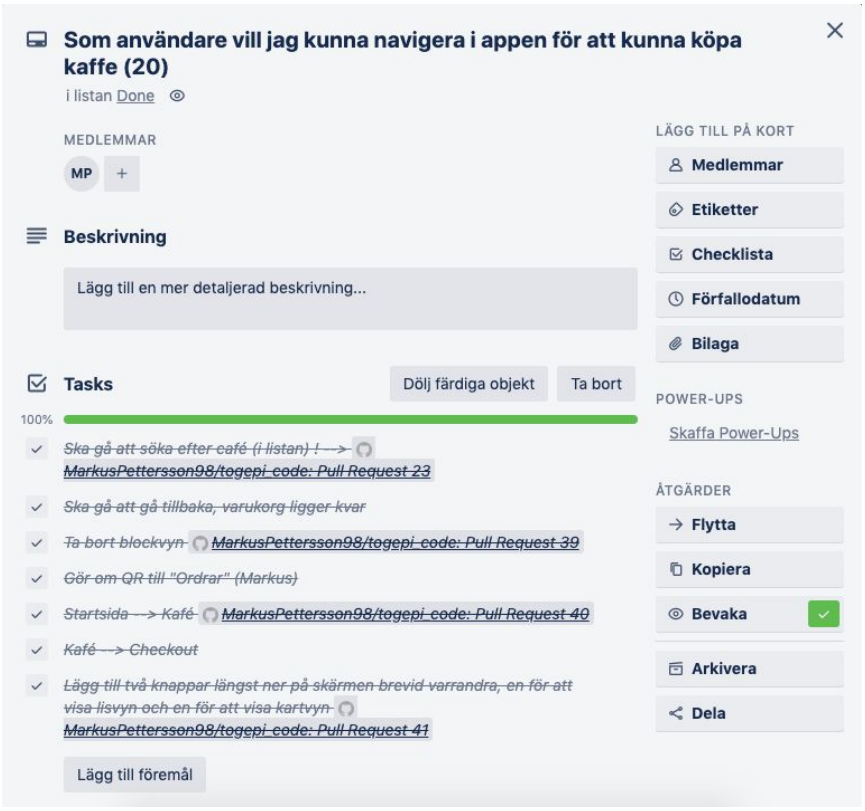
Vi har även dokumenterat koden genom commits, Pull Requests och issues på Github. Genom att dokumentera varje Pull Request som tas in i kodbasen är det enkelt att gå igenom vad som gjorts i appen och se vilka funktioner som presenterats.

Första tillvägagångssätt med sprint 1:
Vi började med att ha Trello och GitHub väldigt separerat, inga länkar sinsemellan vilket kändes problematiskt. Vid en merge fanns det ingen bestämd och/eller naturlig övergång till att flytta kortet på Trello. Det hände att Trello inte alltid uppdaterades, något som vi ville ändra på.

Andra tillvägagångssätt med sprint 4:
För att bemöta problemen med ingen koppling mellan GitHub och Trello bestämde vi oss att länka till Trello-kortet från github. Detta hade också några problem, eftersom det inte gick att se vilka tasks som var färdiga utan att gå till Trello. Figur 11 nedan visar en Pull Request där vi länkat till User Story:n på Trello.

Figur 11: Exempel på hur en Pull Request kunde se ut på GitHub när det länkades till ett Trello kort

På Trello länkade vi även till GitHubs Pull Requests, för att få mer koppling sinsemellan, se figur 12 nedan.



Figur 12: Exempel på hur ett kort på Trello kunde se ut när det länkades till en Pull Request

Gruppen upplevde att även detta kändes svårt, att hoppa mellan GitHub och Trello.

Tredje tillvägagångssätt med sprint 7:
För att förbättra processen bestämde vi oss för att föra över en User Story och dess tasks från Trello, genom att skapa en Pull Request som använder sig av checklists. Detta tillvägagångssätt beskrivs i det sociala kontraktet efter sprintar då gruppen upplevt bristfällig kommunikation kring arbetet.

"En Pull Request ska skapas när man väljer en User Story, detta ger mer transparens om vem som jobbar med vad och en inblick i vad som ska reviewas"

- Utdrag från sociala kontraktet

Då blir Trello endast något som behövs göras vid få, men tydliga stunder, exempelvis när User Storyn tas från "in-progress" till "in-analysis". Figur 13 nedan visar det slutliga utseendet på en Pull Request.

Figur 13: Det slutgiltiliga utseende på en Pull Request i GitHub

Mål

I nästa projekt kommer ett liknande tekniskt dokument författas. Samma arbete med Pull Request som i det tredje och sista tillvägagångssätt kommer att bedrivas, då det erbjöd mest flexibilitet. Därmed är målet med vår fortsatta dokumentation:

- Att göra koden lätt att förstå och sätta sig in i för samtliga utvecklare
- Att göra användandet av Scrum, med User Stories och tasks simplare att följa

Ta sig dit

Arbetsprocessen är iterativ och nästa naturliga steg för oss är att använda oss av GitHubs "Projects"-tab, som ser ut att kunna ersätta Trello. Målet för dokumentation och dess användande är flexibiliteten och möjligheten för alla teammedlemmar att ändra i den. Vi inser att författandet av ett tekniskt dokument kan (i den första veckan) minska levererat värde, men att fördelarna under resten av projektet väger upp för nackdelarna.

How you ensure code quality and enforce coding standards

Idag

Det bestämdes tidigt i utvecklingsstadiet att vi skulle utgå från Airbnb's stilguide för Javascript, en välrenommerad guide för kodstandard, för att bibehålla en hög standard på vår kod. Gruppen använde sig av Pull Requests på GitHub vid utveckling av ny funktionalitet till mobilapplikationen. Genom att blockera sammanfogning av branches in till dev-branchen utan att ha fått sin Pull Request godkänd av någon annan i gruppen kunde kodkvaliteten försäkras. Vi bestämde oss tidigt att minst en person måste gå igenom och godkänna andra personers Pull Requests genom att review:a kodkvaliteten och implementationen. Vidare använde vi oss av Travis CI, ett verktyg för "continuous integration" som agerat som en test/build-svit för oss. Travis har hjälpt oss att undvika "Det funkar på min dator"-problemet eftersom den utgår från git-repot. Vidare för att mäta hur bra våra tester täcker vår kodbas har vi använt oss av Codecov som mäter procentuellt vilken del av koden som exekveras av tester. Dessutom användes testning för att säkerställa att införandet av ny funktionalitet inte gjorde att kod slutade fungera, illustrerat i figur 14 nedan.

```
test('add 1 (one) brygg_kaffe without OWN MUG to cart with one (1) brygg_kaffe with OWN MUG', () => {
    expect(
        cart(brygg_kaffe_ownmug_in_cart, addCoffee({...brygg_kaffe, ownMug: false}))
    ).toEqual(two_brygg_kaffe_ownmug_and_borrowed_in_cart);
});
describe('Testing if getShop returns the right shop', () => {
    it('Should output Bulten store', () => {
        return getShop('Bulten').then(data => {
            const { name, id } = data;
            expect(name).toBe('Bulten');
            expect(id).toBe(1);
        });
    });
    it('Case sensitivity check', () => {
        return getShop('buLtEn').then(data =>
            expect(data.name).toEqual('Bulten'),
        );
    });
    it('Should output Wijkanders store, checking if other restaurants work too', () => {
        return getShop('Wijkanders').then(data =>
            expect(data.name).toEqual('Wijkanders'),
        );
    });
```

Figur 14: Utdrag ur kodbasen för att visa hur tester användes i projektet för att säkerställa att kod fungerade som den ska

Testerna utvecklades i takt med appen och genom Travis och CodeCov kunde vi få en inblick i appens skick.

Sammantaget säkerställer arbetssättet att vår dev-branch endast innehöll kod som fungerar och som hade god kvalitet, eller åtminstone att koden hade gåtts igenom. Gruppen använde sig även av kodformateringsverktyget Prettier för att säkerställa att koden var konsekvent formaterad och lättläslig.

Mål

Målet i framtiden är att fortsätta med liknande strategi då detta var ett bra sätt att garantera fungerade och kvalitativ kod. Continuous integration anses oerhört viktig eftersom man aldrig kan försätta sig i en obyggbar-miljö, en garanti som definitivt ger värde. CodeCov är ett bra verktyg, men utan några tydliga riktlinjer fanns det inget incitament för gruppmedlemmarna att behålla en hög kvot på CodeCov genom att skriva tester. I ett projekt bör CodeCov resultat tas med en nypa salt, då det finns kod som inte behövs testas, samt eventuella komplikationer vid testning av UI.

Vidare krävs välkommenterad kod samt förklarande funktions- och variabelnamn. Ofta utvecklas kod som är lättförståelig för personen som skrivit den, men är svårtolkad för andra i teamet. Målet i framtida projekt är att alla i gruppen ska ha en något gemensam bild över hur man skriver kod som går att standardisera och som det finns olika alternativ för hur man kan skriva. Det är även ett mål att i efterhand jobba på ett strukturerat sätt för att kolla igenom koden.

Ta sig dit

För att ta sig dit ska det tidigt sättas upp verktyg och dokumentation för att sätta en standard gällande projektets kodkvalitet. För att använda sig av CodeCov optimalt krävs regler kring det, exempelvis att 50% av koden är testad. Att inse nyanserna med CodeCov är värdefullt då andelen kod testad är ett lika abstrakt mått som rader kod skriven. Tester som skrivs behöver vara mer generella och vi noterade att hårdkodade tester är suboptimalt. Faktorer som inte direkt berör koden kan strula till det, såsom förändringar i representationer av data. För att göra testsviten mer flexibel bör egenskaper och utvalda attribut på ett dataobjekt testas. Det tillåter utveckling utan att testsviten behöver skrivas om till förmån för att enbart växa i takt med kodbasen.

För att säkerställa kodkvalitet och att koden är läsbar för samtliga gruppmedlemmar och för eventuella externa aktörer som vill kolla igenom koden bör följande ge störst effekt. För det första bör gruppen komma överens om en kodstandard kring hur man skriver kod som ofta upprepas och som det finns alternativ kring hur man skriver. Förutom det kan även anonym peer-to-peer kod-review tillämpas där man efter en sprint låter gruppmedlemmar gå igenom varandras kod. Anonymitet tillåter för en så objektiv granskning som möjligt.

Application of Scrum

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

Idag

I början av projektet delade vi inte upp rollerna eftersom vi inte visste exakt hur vi skulle lägga upp projektet. De flesta arbetsuppgifterna var saker som alla i gruppen var tvungna att gå igenom, varför vi började med att dela upp och specificera arbetsuppgifter istället för roller. Detta gällde mellan läsvecka 1-2 varefter vi började dela upp ansvar till olika personer i form av de klassiska rollerna i Scrum. Under projektets gång har vi delat upp ansvar efter 3 olika roller i projektet, dessa 3 har varit Scrum Master, produktägare och Scrum team. Detta är de klassiska rollerna som ofta framhävs som de centrala delarna i en Scrumprocess.

Figur 15: Traditionella rollerna i Scrum

I vår grupp så har vi roterat dessa roller varje sprint så att att en ny person har varit Scrum Master varje sprint och en ny person har varit produktägare. Resten av medlemmarna i gruppen har då varit del av Scrum team. Detta gjorde vi eftersom vi satte som ett mål där vi ville att alla i gruppen skulle lära sig så mycket som möjligt under projektets gång och vi ansåg att det därför skulle vara bra ifall alla fick testa på alla roller som ingår.

Dessa roller anammades under den första sprinten, som varade mellan läsvecka 3-4, så fick produktägaren rollen att ställa krav på vad som var "värdefullt" att leverera. Produktägaren kunde även tillfrågas för att förtydliga kring vad som behövdes för att uppfylla de krav som sattes och i slutet av sprinten fick produktägaren bestämma om kraven uppfyllts. Scrum Mastern fick under denna första sprint ansvar för att agera ordförande på möten, vara tillgänglig för frågor från gruppen samt hålla koll på GitHub och Pull Requests från gruppen. Efter den första sprinten med dessa roller utvärderade vi hur det hade gått och kom fram till att Scrum Mastern även bör hålla kontakten, samt ha ansvar för att kontakta medlemmar under sprinten för att stämma av hur arbetet går.

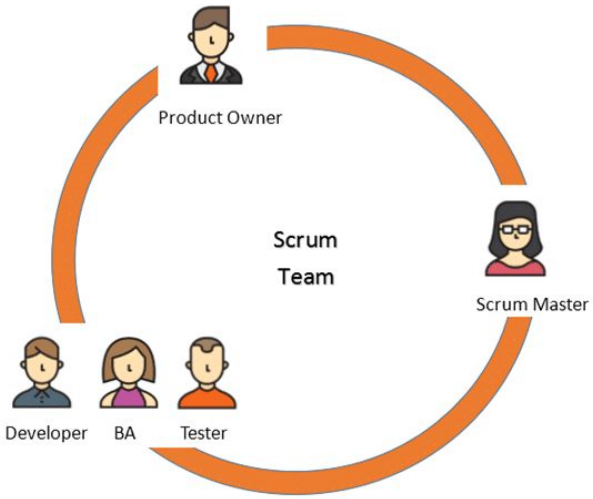
Detta koncept förändrades inte så mycket förrän sprint nummer 5, där vi även kom fram till att produktägaren skulle ta en något annorlunda roll än tidigare. Den rollen produktägaren skulle ta bestämdes under denna sprint till att även innefatta att godkänna eller neka det sista steget i en PR på GitHub.

Mål

Målet till ett framtida projekt, oavsett om det är ett mjukvaruprojekt eller inte, är att ansätta roller tidigare i projektet. Detta för att underlätta administration, arbetsflöde och även vem som ansvarar för specifika uppgifter såsom att boka mötesrum, hålla anteckningar, merge:a in saker i dev (och se till att det inte finns konflikter mellan separata PR) samt att se över tester och hålla koll så att det gruppen skapar alltid är värdefullt. Dessa uppgifter är typexempel på sådant som gruppen under projektets gång tilldelade alla ansvar för att hålla koll på, vilket resulterade i att det under flera sprints var en, några eller alla av dessa uppgifter som slarvades med. Något som formulerades under projektets gång var följande:

> " Har alla ansvar – har ingen ansvar " - Emil Svensson

Efter att vi har jobbat tillsammans ett tag och testat vad som fungerar har gruppen insett att vissa roller hade varit särskilt bra att använda sig utav, förutom de traditionella rollerna som nämndes ovan. Dessa kan sammanfattas i bilden nedan och består alltså av Scrum Master samt produktägare som tidigare, medan roller i Scrum team har delats upp i rollerna Developer, Business Analyst och Tester. Vi tror att dessa roller hade hjälpt gruppen att förstå och bibehålla struktur på arbetsflödet. Om någon får rollen som Tester som är avsedd för att sköta testningen i projektet är det större chans att den blir gjord och att koden som går igenom alltid har testats ordentligt. Om en Business analyst hade funnits, alltså någon alltid ansvarar för att analysera värde i varje steg, säkerställs det att de User Stories som gruppen tar fram kommer att leverera värde. Denna person kan även se till att framtida User Stories gör det samt ta fram en riktning på eller faktiskt ta fram User Stories. Den tredje rollen är Developer, alltså den som håller koll på och ansvarar för de tekniska delarna. Denna roll kan flera i gruppen anta, beroende på projektets omfattning.



Figur 16: En utökad version från de traditionella rollerna i Scrum med tillskott av Developer, BA och Tester istället för bara "team"

Ett annat mål för framtida projekt är att inte rotera roller under projektets gång. Eftersom det gjordes blev det svårt att hålla koll på de exakta ansvarsområden som medlemmarna i gruppen hade och mindre saker, som vem som har ansvar för att boka grupprum, blev lätt bortglömt. I ett projekt där fast ansatta roller finns kan en sådan uppgift enkelt läggas till på Scrum Masterns uppgifter, eftersom den inte är så tidskrävande och bara måste göras. Genom att ha fasta roller istället för roterande, bör det även underlätta arbetsflödet och göra att medlemmar kan bli bättre på sina arbetsuppgifter efter att de kommit in i sina roller. En annan fördel med att ha tydligt definierade uppgifter är att gruppmedlemmarna kan få en starkare inre motivation genom att känna att de är ansvariga för en tydlig del av projektet. Det medför även att en förståelse för hur den egna insatsen spelar en roll i projektet fås och vad de andra medlemmarna kommer att göra.

Ta sig dit

För att ta sig till målvisionen bör det tidigt i projektet specificeras vilka de olika rollerna är och vad dessa kommer innefatta i termer av ansvarsområden och uppgifter. Dessa bör sedan skrivas ner i ett socialt kontrakt och gruppmedlemmar bör ansättas till dessa rollerna. Genom att skriva ner en beskrivning av rollerna möjliggör det för gruppmedlemmar att gå tillbaka till kontraktet för att förstå vad dess ansvarsområden är. Vidare bör detta kontrakt vara ett levande kontrakt i det avseendet att rollernas ansvarsområden kan utvidgas under projektets gång för att byggas på med sådana uppgifter som faller inom en rolls ansvarsområde.

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

Idag

Följande agila metoder användes under projektet:

- **Backlogs (Product and Sprint)**

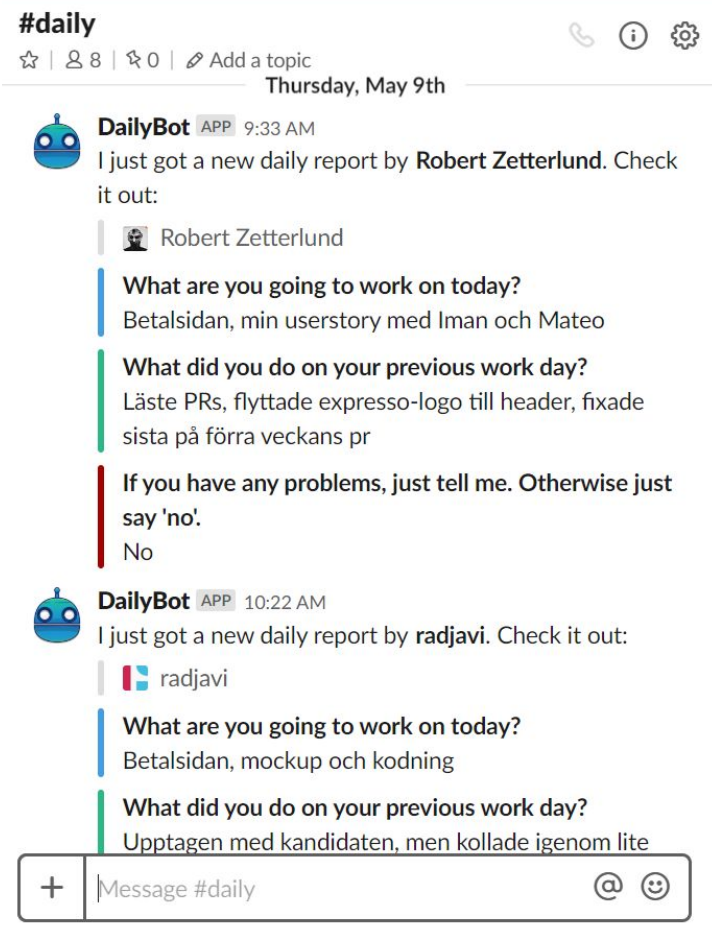
  Vi följde vår product backlog och sprint backlog under sprintarna. Product backlog listade i prioritetsordning allt som behövde levereras inom ramen för projektet och låg som grund för

sprint planning. Sprint backlog listade allt som behövde levereras under aktuell sprint. Båda backlogs gav överblick över vad som behövde göras.

- **Scrum events**

  Närmare bestämt sprint planning, sprint review och retrospective. Dessa hjälpte till att strukturera och fördela sprintens arbete (planning), diskutera det som levererats under sprinten (review) och analysera sprinten, kommunicera lärdomar och diskutera vad som borde ha gjorts annorlunda (retrospective).

  För att hålla en bättre kommunikation använde gruppen sig även av en modifierad version av Daily stand-ups. Eftersom gruppen inte alltid satt och jobbade på samma ställe inleddes varje möte med ett snabbt stand-up möte där gruppmedlemmarna fick dela med sig av eventuella problem eller bara allmänt dela kunskap sinsemellan. Förutom detta implementerades även en chatbot i Slack som skickade en påminnelse varje dag och bad gruppmedlemmar att lämna en statusrapport, se figur 17 nedan.



Figur 17: Utdrag från Slack för att visa exempel på hur inrapporteringen kunde se ut från Daily stand-up möten med chatbot

- **Agila roller**

  I gruppen ingick och roterade varje sprint rollerna Scrum Master och produktägare. Läs mer om hur dessa roller påverkade vårt arbete i frågan ovanför.

- **User Stories**

  User Stories hjälpte till att sätta ord på det som behövde göras under sprintarna. User Stories bröts ned i specifika tasks. Detta avgränsade arbetet och hjälpte till vid arbetsfördelning.

- **Velocity och effort**

  Med velocity kunde gruppen avgöra hur förutsättningarna såg ut inför en sprint. Hur många timmar kunde samtliga gruppmedlemmar lägga på projektet under sprinten? Velocity var således summan av alla timmar. Sedan kunde effort sättas på User Stories, också detta specificerat i timmar. Med velocity och effort kunde vi alltså avgöra exakt hur många User Stories som kunde levereras under en sprint.

  Gruppen hade ibland problem att uppskatta effort eftersom gruppmedlemmarna låg på olika kunskapsnivåer. Om en person uppskattade att en User Story hade effort 1 kanske en annan uppskattade att den hade effort 5.

- **Iterativ och inkrementell utveckling**

  Det som levererades under den avslutade sprinten utvärderades av gruppen och sedan planerades nästa sprint. Appen omarbetades och förbättrades alltså kontinuerligt utifrån respons från gruppen.

  Förslagsvis borde produktägaren varit mer aktiv under utvärderingen snarare än att hela gruppen gav respons.

Mål

I fråga om agila metoder borde gruppen bli bättre på att uppskatta effort till User Stories under sprint planning. Just sprint planning borde, liksom alla andra Scrum events, effektiviseras.

Ta sig dit

Genom att reflektera kring effort hos avklarade User Stories kan effort hos nya User Stories lättare uppskattas. Detta bör göras under retrospective, alltså i slutet av en sprint. Gruppen borde då fråga sig om effort stämde överens med den tid som behövdes för att göra färdigt en User Story. Om inte, vad berodde detta på? Dessa reflektioner borde utgöra underlag för nästkommande estimeringar av effort snarare än att gruppmedlemmarna enbart utgår ifrån sin egen kunskapsnivå och hur mycket tid hen behöver för att avklara en User Story.

Scrum events kan effektiviseras om de har en mötesordförande (Scrum master), en agenda som specificerar vad som måste behandlas och slutligen ett tydligt mål. Gruppmedlemmarna borde

hindras från att byta ämne eller avvika från agendan. Slutligen borde produktägaren och Scrum master förbereda sig ordentligt inför mötet och bestämma sprintens mål.

The sprint review and how it relates to your scope and customer value

Idag

Under varje sprint review har gruppen presenterat vilka inkrement som gjorts under sprinten. Detta har gett produktägaren en möjlighet att kommentera funktionaliteten på vad som gjort för att se om det är redo för att gå i produktion eller om det behöver göras förändringar. Sedan har även alla i gruppen fått komma med insikter om design och kodstruktur. I slutet av varje sprint review har produktbackloggen uppdaterats.

Mål

I framtida projekt bör en sprint review skötas mer systematiskt. Istället för att det blir en spontan process under möten är målet att kunna åsidosätta dedikerad tid till detta där gruppen diskuterar mer utförligt kring respektive User Story. Detta är bra för att verifiera att en User Story skötts rätt enligt projektets standard.

Ta sig dit

För att i framtida projekt säkerställa att detta görs på ett systematiskt sätt bör en standard kring hur sprint-review görs upprättas. Detta kan vara i form av ett simpelt schema där samma punkter alltid gäller och de betas av i samma ordning i varje sprint review. Detta bör även modereras av Scrum mastern på ett tydligt sätt, speciellt ifall det dyker upp saker som måste behandlas, men som är utanför schemat.

Best practices for learning and using new tools and technologies

Idag

I detta projekt har många tekniker använts som tidigare varit främmande för majoriteten av gruppmedlemmarna. För att rationalisera arbetsbördan av att hämta in all ny kunskap som har krävts för att genomföra projektet sattes en Wiki upp på Github. Att sätta upp en Wiki bidrog till att vi kunde arbeta effektivare och lättare implementera verktyg som andra i gruppen använt tidigare, då en eller några få gruppmedlemmar kunde sammanställa matnyttig information och dokumentera viktiga steg och eventuella fallgropar i detalj.

I gruppen har vi tagit nytta av varandra för att på bästa sätt ta till oss nya kunskaper. Vi har arbetat med att dela med oss länkar för att alla i gruppen ska utgå från samma praktiker vilket gör det enklare att förstå vad som görs i kodbasen. Med många utvecklare som arbetar med olika User Stories var det viktigt att göra koden homogen. Att alla i gruppen arbetar efter samma dokumentation och exempel gör det enklare att gå in och utveckla befintliga kodstycken.

Mål

Att alla i gruppen ska arbeta efter samma kunskap och enkelt kunna nå den kunskap som finns inom gruppen. Alla gruppmedlemmar ska få goda möjligheter att erhålla praktiska kunskaper om nya verktyg och tekniker. Genom att lära av varandra går det att undvika att få medlemmar som sitter på nischad kompetens om delar av kodbasen, som ingen annan i gruppen besitter kompetens inom.

Ta sig dit

Genom att vidareutveckla vår Wiki kan vi enkelt dela med oss av kunskap om olika tekniker inom gruppen. Detta skulle kunna inkorporeras som en del av varje sprint att tids avsätts för att utvecklarna ska kunna dokumentera sitt arbete i Wikin för att säkerställa att den nya information som inhämtats görs tillgänglig och hålls uppdaterad för hela gruppen.

Vi ser även att stand up meetings och parprogrammering kan nyttjas bättre för att lära sig nya verktyg. Detta kan ge ett bättre utbyte inom gruppen och möjlighet att stötta varandra i de tekniska problem som uppkommer under utvecklingsfasen. Parprogrammering särskilt då det dels ger praktiskt erfarenhet, men också för att det ger tid för utvecklare att faktiskt sitta ned ihop och dela med sig av sina kunskaper. För stand-up meetings har vi använt Slack, men detta har kunnat använts på ett mer konsekvent sätt. Under arbetet har svaren från Slack inte följts upp på ett tillräckligt bra sätt som garanterar att informationen når ut till alla gruppmedlemmar varje sprint.

Relation to literature and guest lectures

Idag

Genom arbetet har vi gjort vår främsta för att arbeta agilt med Scrum, som metoden presenterades på föreläsningarna under uppstarten av kursen. Vi har även deltagit i de gruppövningar som kursen erbjöd, för att snabbare sätta sig in i arbetssättet. Vi har gjort en bra progression inom gruppen med att arbeta agilt. Genom arbetet har vi försökt ta stöttning av litteratur för att sätta upp KPI:er, identifiera roller i gruppen och hitta agila arbetsmetoder.

Mål

Det är viktigt att ta till vara på den interna kunskapen som finns inom gruppen på ett bra sätt, framförallt om det finns praktisk erfarenhet inom gruppen. Detta är något som i första hand bör göras, framför att söka sig kunskap från extern litteratur. Vid de tillfällen där det inte finns intern kompetens bör gruppen alltid försöka se ifall det finns relevant litteratur som kan användas för att hjälpa projektet gå framåt.

Något som är viktigt är att vara kritisk till all kunskap, både intern och extern och inse att det kan bero på fall till fall och vara kontextberoende. Innan kunskap som gruppmedlemmar läst/hört används är det därför viktigt att analysera och jämföra för att se hur kunskapen kan göra nytta i den givna kontexten gruppen och projektet befinner sig i för stunden.

En del i varje sprint är att se över och dokumentera vårt arbetssätt. Ett större ansvar för detta bör läggas på Scrum Master. Scrum Master kan göra en sammanställning av den utvärderingen som görs och analysera det med hjälp av litteraturen. Scrum Master kan då presentera förbättringar inför nästkommande sprint som grundar sig i befintlig litteratur.

Arbetssätt

I figur 18 nedan återfinns ett utdrag ur Git Inspector.

Statistical information for the repository 'togepi_code' was gathered on 2019/05/31.
The following historical commit information, by author, was found in the repository:

Author	Commits	Insertions	Deletions	% of changes
Emil Svensson	30	1351	300	7.42
Iman Radjavi	65	4255	1878	27.57
Markus Pettersson	84	3192	1650	21.76
Mateo Raspudic	3	195	104	1.34
Robert	10	1006	650	7.44
RobertZetterlund	67	1674	1041	12.20
Sofija Zdjelar	2	278	27	1.37
lucasekmarkfallqvist	30	2586	1040	16.30
minken01	1	191	15	0.93
pklara	12	170	171	1.53
raspudic	3	219	148	1.65
sofija zdjelar	1	108	0	0.49

Below are the number of rows from each author that have survived and are still intact in the current revision:

Author	Rows	Stability	Age	% in comments
Emil Svensson	458	33.9	0.3	6.11
Iman Radjavi	1950	45.8	0.5	1.79
Markus Pettersson	1713	53.7	0.5	12.08
Mateo Raspudic	185	94.9	0.3	0.00
RobertZetterlund	1073	64.1	0.7	4.47
Sofija Zdjelar	191	68.7	0.5	5.24
lucasekmarkfallqvist	490	18.9	0.6	4.49
minken01	84	44.0	0.3	14.29
pklara	45	26.5	0.3	4.44
raspudic	209	95.4	0.1	0.00

Figur 18: Git Inspector i slutet av projektet som visar varje gruppmedlems bidrag

Git Inspector riskerar ge en missvisande bild av gruppmedlemmarnas insats. Istället måste detta ses i relation till olika parametrar. I nedanstående lista finns orsaker till varför Git Inspector ser ut som det gör:

- Programmering skedde ofta i par eller i grupper om tre vilket gjorde att alla kod-skribenter som deltog inte syns.

- Eftersom gruppmedlemmarna inte hade någon tidigare erfarenhet av React Native resulterade detta i att de behövde lära sig detta istället för att börja utveckla direkt.
- Kod som ligger till grund för hela projektet, exempelvis färdiga bibliotek, bidrog till väldigt många commits och rader kod.
- Formateringsverktyget Prettier ställde ibland till problem vilket resulterade i att en mängd rader kod räknades in när det egentligen inte var något nytt.
- En package-lock.json fil som ingår i Node.js var ibland problematisk med verktyget Travis vilket också resulterade i att en mängd rader kod räknades in felaktigt.

För att undvika att kommande projekt har liknande problem kommer vi att använda oss av gits "co-commits". Det vi lär oss av Git Inspector är att antal rader kod är ett komplicerat sätt att mäta deltagande och att vi gjort rätt genom att inte ha det som ett KPI.

Avslutande reflektioner

I början av projektet bestämde vi oss att vi ville tänka brett med avseende på mobilitet och hitta en lösning som kunde hjälpa människor i sin vardag med att bli mer hållbara. Vår lösning blev en app som kan tackla problematiken kring engångsartiklar. Appen är ett hjälpmedel för att dricka kaffe "on the go" som samtidigt skapar en lättanvänd, hållbar lösning för ersättning av engångsartiklar.

Under projektets gång har vi lärt oss mycket om att arbeta agilt, mer specifikt med Scrum-metodiken. Det som märks tydligast är hur kommunikationen förbättrats inom gruppen och hur mycket bättre vi blivit på att i varje inkrement leverera värde till produktägaren. Vi har också tagit mycket lärdomar om hur framtida arbeten kan läggas upp. Att ha dedikerade roller under projektet gör det enklare att skapa struktur under arbetet och skapar förutsättning för att personer kan utvecklas inom rollerna för att göra arbetsprocessen bättre.

Till framtida projekt hoppas vi kunna gå in med en större mognad till Scrum som arbetssätt och metodik. Vi har tagit stora lärdomar om vad som är viktigt att göra för att lyckas med arbetsstruktur och sätt att förbättra processen. Under projektet har vi även lärt oss mycket om att utveckla mobilapplikationer. För många i gruppen var JavaScript nytt, men vi valde ändå att utveckla i React Native. Detta var en chansning som gav goda resultat. Vi har utvecklat en mobilapplikation som är väl genomtänkt och presentabel. Kunskaperna inom React Native och utvecklingsprocessen är något vi kan ta med oss till framtida projekt.