BusCount by hiHats
Post Mortem-rapport

30/10 2015

Henrik Nilson

Axel Aringskog

Filip Hallqvist

Introduktion

Rapporten avser att förklara och beskriva hur processen gick till när konceptet och prototypen för appen BusCount byggdes, samt ta upp de fram- och motgångar gruppen stött på under utvecklingen.

Projektöversikt

Projektet har gått ut på att bygga och planera appen BusCount. Appens syfte är att låta användare få poäng för sina bussresor i utbyte mot erbjudanden på ekologiska produkter och tjänster. De aktörer som knyter sig till tjänsten köper en marknadsplats och marknadsför sig genom att sträcka sin miljöprofil.

För att användaren ska kunna hålla koll på sina poäng och se statistik över tidigare genomförda bussturer kräver appen att varje användare skapar ett konto innan appen kan användas. Detta lämnar inte bara utrymme för användaren att se intressant information om sina egna resor, utan öppnar också för möjligheten att i framtiden koppla ihop appen med andra tjänster och sociala nätverk.

All data som behöver lagras för att appen och dess koncept ska kunna fungera i praktiken har gruppen valt att lagra med hjälp av tjänsten Parse (parse.com). Denna data innefattar bland annat geografisk information för busshållplatser, erbjudanden, användares genomförda bussturer och inloggningsuppgifter.

För att garantera att användaren får poäng för sina resor krävs att denne tillåter appen att ha tillgång till telefonens position under delar av bussresan, åtminstone vid varje hållplats. I samband med detta valde gruppen att använda Google's “Fused Location API” för att på bästa sätt kunna bestämma enhetens latitud och longitud, avgöra ifall användaren faktiskt åkte buss och vidare på vilken buss användaren satt.

Under utvecklandet av appen har gruppen nyttjat arbetsflödet Scrum med Trello (trello.com) som plattform. Detta satte också sin prägel på hur gruppen arbetade med versionshantering. Versionshanteringen sköttes med hjälp av GitHub (github.com), där arbetsflödet i Git bestod av feature-branches kombinerat med sprint-branches.

Kommunikationen i gruppen har skett med hjälp av Slack (slack.com), en meddelande och planerings-plattform.

Tidsplanering

Planeringen har under hela projektet utgått ifrån en 20 timmars arbetsvecka. Självklart är dock att det varierat lite mellan de olika veckorna hur mycket tid som lagts ner. Någon vecka har det blivit lite mindre en annan har det blivit mer, beroende mycket på dels händelser i den parallelt gående kursen men också viktiga “deadlines” inom projektet. Tyvärr har det resulterat i en mer eller mindre exponentiell utveckling istället för det som var önskat, nämnligen en konstant.

Alla i gruppen har försökt att följa tidsplaneringen så gott det går, och har lyckats relativt väl med detta. Det har förekommit att några i gruppen jobbat utanför tidsplanering och därmed bidragit mer till projektet. Detta har dock inte lett, som man kanske skulle kunna tro, till någon sorts “varför gör jag mycket mer än alla andra” -frustration då vi på ett tidigt stadium gick igenom och satte upp riktlinjer för hur mycket som borde göras varje vecka och att man på egen räkning sedan kunde få jobba hur mycket man så önskade på fritiden.

Reflektion över metoder och verktyg

SCRUM

När vi gick in i det här projektet valde vi att använda arbetsmetodiken SCRUM. Det har tidvis varit bra och vi har plockat ut några metoder som vi tyckte fungerade bra och som nämns nedan. Tidvis har det även fungerat sämre då vi till viss del saknat disciplinen som krävs för att hålla arbetstempot uppe under sprintarna och slarvat med planeringen.

Sprints

Vi använde oss utav en vecka långa sprints under projektet, det vill säga fyra stycken under tävlingen och en under sista veckan. Under sprintarna fokuserade vi på att dela upp större uppgifter, use cases, i små lätthanterliga delar. Vi försökte att alltid bli klara med uppgifterna och att ha ett körbart program efter varje vecka. Vi använde oss av

verktyg såsom Trello (trello.com), för att hålla ordning på vem som gjorde vad, och git branches för att undvika att förstöra föregående sprints arbete. Både Trello och Git (git-scm.com) har fungerat väldigt bra och vi skulle helt klart använda det igen i kommande projekt.

Fördelarna med sprintar var att man varje vecka fick gjort någonting – att man hela tiden pushade projektet framåt. Har man en stor uppgift som sträcker sig över lång tid kan det vara svårt att se slutet på arbetet, men med hjälp av sprints och uppdelningen av små uppgifter fick vi en metodisk och effektiv arbetsrytm som bidrog till att projektet drevs framåt.

Det svåra med sprintarna var att veta hur man på bästa sätt skulle dela upp olika use cases och sedan fördela dessa jämnt över gruppmedlemmarna. Ibland var det också svårt att hålla sig till tidsplanen, ofta just för att man underskattade hur lång tid en uppgift skulle ta. Även att fullfölja en uppgift med tester har också gruppen haft svårt för. Vilket man i efterhand kan se genom att viss avsaknad av tester.

Själva grundtanken med sprintarna är att man varje vecka ska ha ett fungerande program/applikation och då vi efter varje sprint lyckades uppfylla detta grundkoncept så tycker vi att metoden var väldigt tidseffektiv.

Om man blir tilldelad ett projekt där kunden/produktägaren inte riktigt är hundra procent säker på vad hen önskar och man på kort tid behöver ta fram en eller flera prototyper skulle vi använda oss av sprintar igen.
I projekt där produktägaren vet exakt vad hen vill ha anser vi att det kan vara överflödigt med sprintar och man kan använda planeringstiden till att utveckla programmet/applikationen istället.

Standup Meetings

Under projektet så använde vi oss under ett par tillfällen av standup meetings.
Även om vi inte hade särskilt många möten så var fördelarna med metoden tydliga. Efter varje möte så blev arbetet mer effektivt och vi fick mer gjort än de dagar då vi inte hade möte. Föredelarna grundar sig i att det blev lättare att sätta delmål, få en överblick över

dagens problem och att alla fick en chans att i ett tidigt stadium identifiera och få hjälp med problemen genom ett par timmar parprogrammering.

Nackdelarna med metoden ansåg vi vara få, men det betyder inte att de inte fanns. Det mest utmärkande problemet vi hade med metoden, vilket också var anledningen bakom att det inte blev så många möten, var att det kunde vara svårt att få hela gruppen samlad och fokuserad samtidigt. Ibland var det någon som var sen när ett möte skulle inträffa, ibland hade någon ett sådant flow att ett avbrått i arbetet kändes omöjligt. Även om det gick att ha ett möte på 3 av 5 personer så var det inte alls lika effektivt som när alla var där och bidrog. Här är något gruppens medlemmar får jobba mer inviduellt på, att alltid komma i tid så att Standup meetings inte försummas.

Med tanke på att andelen avklarat arbete blev större de dagarna som vi hade möte och att möte ofta är korta så blir även denna metod tidseffektiv.

Vi skulle använda samma metod igen om jag hade ett liknande projekt framför mig, det vill säga utveckling av någon sorts produkt i ett mindre team.

Egentligen ser jag inte nåt projekt där man inte skulle vilja använda sig av standup meetings. Eventuellt om man vet med sig att man kommer jobba mycket på olika platser. Eller att det är överflödigt att använda sig av om man har en väldigt bra överblick över vad som behöver göras och hur man ska genomföra det.

Pair Programming

Fördelarna med parprogrammering är många, det är till exempel alltid bra att få en utomstående åsikt till hur man löser problem. Det uppstår även ofta situationer där man kan dra lärdom av varandras kunskaper. En annan fördel är att man underlättar för allas förståelse av projektet genom att tillsammans diskutera fram bästa lösningen på ett problem.

Även om det som sagt finns många fördelar med parprogrammering så finns det också nackdelar, man får som sagt en större överblick och bättre insikt i hela programmet, men det är på bekostnad av ett mindre tidseffekitvt arbete, därför att man helt enkelt har två personer som jobbar på samma uppgit.

Som nämnt ovan kan det på kort sikt bli mindre tidseffektivt att använda sig av parprogrammering, om man är i tidsnöd och snabbt behöver få fram en lösning på ett stort problem eller behöver få fram en prototyp till en kund kanske man ska undvika metoden så gott det går, men om man har god tid på sig finns det långsträcklig vinst att plocka hem på det. Det för att förståelse för programmet och bättre insikt i hur de olika delarna i programmet fungerar leder till mindre refaktoreringsarbete och minskar risken för upprepat arbete, det vill säga att flera personer skiljt från varandra jobbar med samma typ av problem.

I framtida projekt där det är viktigt att från början ha gemensam uppfattning, där någon i gruppen är ledande på ett kunskapsområde eller där det krävs lite extra för att komma igång skulle vi använda oss av parprogrammering igen.

Däremot skulle vi inte anväda den här metoden om man som sagt snabbt skulle behöva visa upp en produkt för en kund eller möjlig investerare, där de som jobbar i projektet kan ansvara för en egen liten del och inte behöver bry sig så mycket om hur det samverkar
med resten av gruppens arbete.

Vi använde oss mycket av parprogrammering i början, när vi skulle sätta igång projektet och bestämma vad alla skulle jobba med. Det var bra då alla fick vara med och bestämma hur upplägget skulle se ut och alla fick en bra bild över hur arbetet skulle fortskrida. Senare har vi arbetat mer eller mindre enskilt på olika områden vilket också har fungerat bra, vilket vi tror är en följd av att vi tillsammans la upp projektstrukturen i början.

Parse

Att arbeta med Parse har varit förvånansvärt enkelt vilket gjorde att gruppen slapp lägga onödig tid på att själva bygga en liknande molntjänst, och kunde lägga mer tid på att utveckla själva appen. I tidigare kurser har vi fått lära oss att programmera mot modulära “black boxes”, att man enkelt ska kunna byta ut en del av applikationen emot någon annan vilket gjorde att vi ville göra samma sak med Parse.

En av de största utmaningarna med just detta har varit att hantera all data asynkront, men genom att bygga hjälpklasser med funktionell programmering i bakhuvudet löstes

detta problem med hjälp av callbacks. Tyvärr sköts fortfarande en del av sorteringen av datan i fragmenten, något som annars kunde undvikts med lite smartare databasfrågor. Detta är ett område vi gärna lagt lite extra krut på om det funnits mer tid.

Innovationsplattformen

Vi använde oss mycket under utvecklandet utav ElectriCity's eget API, "Innovationsplattformen". Även om vi förstår att det kan vara svårt att ta fram ett API som detta och att man inte vet riktigt hur det kommer att fungera innan det används så var det tråkigt och frustrerande att man flertalet gånger blev avbruten i sitt arbete på grund av att servern låg nere. På det stora hela har API:et dock fungerat i linje med förväntan och det har varit både intressant och spännande att jobba med ett så innovativt och kreativitetsgenererande verktyg.

Vi använder oss inte, förutom en del små api:er kopplat till android och google, utav några andra stora api:er. Detta är dock något som vi skulle vilja ändra på i framtiden under vidare utveckling av applikationen. Vi har funderat på att lägga till möjligheten att dela sina prestationer på olika sociala medier och att utöka applikationen till att bli en fullfjädrad reseplanerare. Dessa är båda användningsområden som kräver tillgång till olika API:er.

"Runt Omkring"

För att kunna göra ett bra jobb är det viktigt för varje projektgrupp att hålla uppe en god stämning och se till så att var och en i gruppen mår bra och är på bra humör. För att lyckas med att uppnå denna utopi av välmående och att hålla moralen uppe såg vi till att, genom noga planerat utförande, alltid hålla koffein och sockerintag på acceptabla nivåer. Detta i form av den ständigt återkommande "klockan tre fikan". Skämt o sido så var det viktigt för att uppehålla ett effektivt arbetssätt att införa pauser där man inte behövde tänka på jobbet. Dels för att det motverkar risken för att köra fast, dels för att det ökar de sociala aspekterna inom gruppen och dels för att miljöombytet ger en mer avslappnad stämning inom gruppen. Ett bra sätt att få gruppen att jobba mer effektivt helt enkelt.

Det förekom också pauser innehållandes ett eller annat parti datorspel, ett verktyg som fungerade bra som moralhöjare, och sammarbetsstärkare för gruppen. När man märkte att man kört fast, behövde tänka på något annat en liten stund eller bara av någon annan anledning behövde en paus i arbetet.

Problemområden

Utvecklingsprocessen har inte bara bestått av framgångar, utan gruppen har även stött på en hel del utmaningar. En av de större utmaningarna har varit att se till att följa upp, dokumentera och utvärdera koden i samma tempo som den skrivs. Detta syns tydligast på avsaknaden av uppföljningsmöten efter varje avklarad sprint. Att arbeta på det sättet gör att utvecklaren kan få ut mer kod i produktion i ett snabbare tempo, men resulterar också i fler kodstycken som måste städas upp i efterhand.

I samband med det blev också tidsfördelningen över hela projektet väldigt ojämn. Att kod följs upp löpande leder till att utvecklandet i sig blir mer konkret, något som annars sker först när programmet det mesta av funktionaliteten kommer på plats. Med andra ord är det först i slutet av projektet som buggar och felaktigheter i logiken upptäckts. Under projektets gång fanns endast en enhet att testa ap pen på. Emulatorer idag fungerar väldigt bra, men naturligtvis går det aldrig att simulera alla tänkbara scenarion. Att gruppen bara hade en enhet att testa på gjorde att utvecklingen gick långsammare och i kombination med att det knappt skrevs några tester var det ofta bara att hålla tummarna och lita på att koden även fungerade på en riktig enhet.

Lessons learned

Kommunikation

Stämningen i gruppen har varit riktigt bra under kursens gång, vilket har bidragit till att vi har kunnat jobba som ett lag. Alla i gruppen har vågat uttrycka sin åsikt och hela gruppen lyssnar. Detta har underlättat arbetet och gjort att gruppen har kunnat koncentrat sig på projektet.

Dock har vi haft olika åsikter om definitionen av klar och vad som behövs prioriteras. Vilket har resulterat i att gruppen inte har haft ett tydligt mål, utan olika mål. Då kursen inneburit mycket mer arbete än att skapa en stabil kodbas har vi haft vissa splittringar inom gruppen.

Arbetssätt

Vi har under projektets gång i princip alltid arbetat gemensamt i studierum eller på café. Det har varit ett bra sätt att få insikt i vad var och en i gruppen jobbar med och snabbt kunna få hjälp med ett problem. Det har också ökat förståelsen för hur applikationen byggts upp. Då vi kände varandra sedan tidigare projekt kunde vi snabbt komma igång med arbetet och det har inte varit några problem att föra diskussion över hur man skulle tackla ett problem.

“The big picture”

Gruppen missade alla utvecklingsverktäder vilket inte var avsiktligt vi hade lottat om vem som skulle gå på vilken utvecklingsverkstad. Men sen glömdes de bort under projektetsgång. Vilket var väldigt synd då vi missade helhetstänket i konceptet och prototypen. Gruppen har sakta börjat klättra ut ur sin svettiga programmeringskällare dock inte tillräckligt, vilket märktes på presentations dagen. När vi gick in för att pitcha idén till juryn kände vi oss säkra på konceptet. Snabbt blev vi motbevisade av juryn när de börjde ställa frågor som vi inte var beredda på.

Vi insåg försent att utveklingsverkstäderna hade kunnat hjälpa oss genom att plantera tankar utanför andriod stuido och få oss att även tänka på hur idén kan finansieras, är någon intresserad av att knyta sig till vårt koncept?
Att tänka på helheten i idén och inte bara jämföra vad som är snyggast mellan en if sats och switch sats. Precis som sista utvecklingsverskatden hette “Från ide till Affär” så tar vi med oss lärdomen att en app kan vara grym men om man inte tänker på hela processen med allt från kod till marknadföringn så spelar det ingen roll hur snyggt UML diagrammet är.

En annan viktig lärdom som vi tar med oss från dethär projektet; Det är lätt att sätta en tidsplan, men svårt att följa den. Detta är något som vi fått erfara under tidigare projekt

men extra mycket under detta projekt då vi till skillnad från tidigare inte endast programmerat ett program (eller i det här fallet en applikation) utan även deltagit i en innovationstävling med allt vad det innebär.

Vi gjorde en uppskattning på att vi hade mellan 10-15 timmar per vecka för att programmera. Enligt SCRUM principen tilldelade vi timmar poäng och delade upp olika use cases i mindre poängsatta och prioriterade uppgifter. Som nämnt ovan visste vi på ett ungefär hur mycket tid vi var tvungna att lägga ner på programmering varje vecka. Det vi inte hade räknat med var hur lång tid det skulle ta att filma koncept/prototyp-filmer, skriva genomförandeplaner, förbereda pitch och så vidare. Arbete med att skapa film och slipa på interaktionen med appen var väldigt lärorikt och roligt men det tog enormt mycket tid. Detta resulterade sedan i att vi fick lägga betydligt mer tid på det sistnämnda och därför hade svårt att följa tidsplanen.

Till nästa projekt

Det har varit spännade att lära sig tänka mer agilt och testa på att arbeta med scrum. Dock skall nog scrum appliceras på ett projekt där alla deltagare har en vardag som inte innehåller en kurs i matematik och andra distraktioner. Det agila tänket är något hela gruppen tar med sig.

Tidsplanering och tidsestimering av uppgifter är verkligen skit svårt. En uppgift är aldrig skriven i sten och ändras kontinuerligt. När man jobbar med en iterativ metod så ändras den ännu oftare.

En annan aspekt som vi kommer ta med oss ifrån projektet är “think outside the code”, ett uttryck som växt fram under projektets senare del, det innebär att man inte endast vara fokuserad på att få ut en väl fungerande app med bra kod under skalet utan också tänka över hur man ska uppnå de ambitioner man har med appen. Man märkte tydligt hur oförberedda vi var när vi skulle pitcha våran idé, eller kanske rättare sagt efter vi hade pitchat vår idé och skulle svara på frågor om hur vi ville ta projektet vidare.