ESCAPE

DAT255 (HT13)

Book, Adam
Eriksson, Johanna
Jansson, Carl
Persson, Simon
Phoohad, Mike
Tholén, Erik

CHALMERS

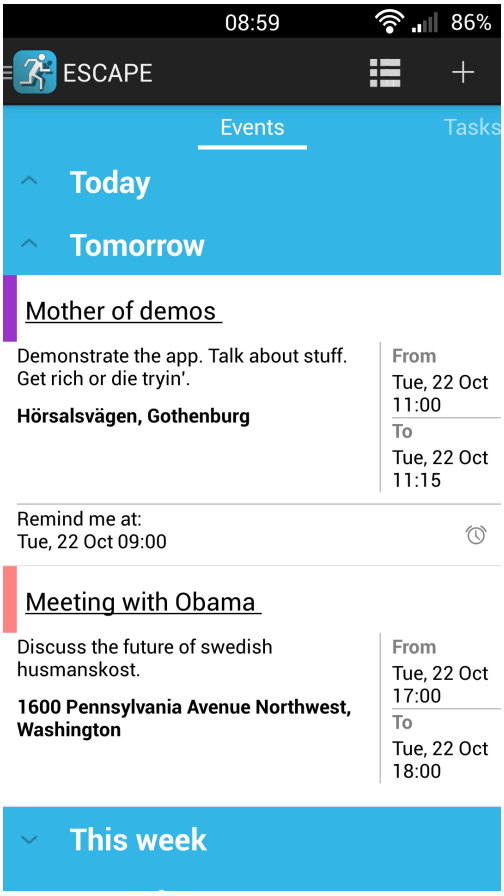
Innehållsförteckning

1 Kort om projektet

Det finns otaligt många Androidapplikationer ute på marknaden idag vars syfte är att underlätta den vardagliga planeringen för individer. Varje app representerar sin utvecklares egen tolkning av hur ett effektivt planeringsverktyg ska fungera och se ut, och några av dem fungerar riktigt bra. Däremot erbjuds sällan några unika, riktigt smarta funktioner som verkligen hjälper användaren med sin planering och det är just där detta projekt, och i synnerhet applikationen ESCAPE, kommer in i bilden.

ESCAPE är ett slags verktyg som kombinerar vanliga “att-göra-listor” och kalenderevent med möjligheten att automatiskt kunna generera veckovisa scheman. Användaren specificerar titel, totalt antal timmar för en arbetsuppgift och hur lång tid varje arbetspass ska pågå, varefter appen automatiskt skapar och placerar ut nya kalenderevent på annars ledig tid.

Utöver autogenereringen finns även funktioner som tids- och platspåminnelser, samt en så kallad Pomodoroklocka som hjälper användaren att hålla ett jämnt fokus under längre arbetspass.



2 Processer som använts

2.1 Scrum

Gruppen bestämde sig tidigt för att använda Scrum, en arbetsmetod som sedan användes under hela projektets gång. Scrummöten hölls nästan varje vardag, med en ny sprint som påbörjades varje måndag och avslutades på fredag. Därefter hölls en sprint review i samband med ett sprintmöte på måndagar för att planera kommande veckans sprint.

Några personer i gruppen tilldelades särskilda roller, som Scrum Master och Product Owner. Syftet var att fördela arbetsbördan något, och låta utvalda personer ha ett större ansvar för de viktiga områdena.

2.2 Parprogrammering

Parprogrammering användes flitigt under de inledande veckorna av projektet. Gruppen delades upp i par, och varje par tilldelades egna arbetsuppgifter att utföra under veckan. Inom varje par turades sedan parmedlemmarna åt om att arbeta med endast en dator till förfogande - den person som inte manövrerade datorn satt helt enkelt bredvid sin arbetskamrat och granskade den kod som skrevs.

De sista veckorna ansågs det vara mer tidseffektivt att jobba på egen hand, delvis på grund av olika kunskapsnivåer inom gruppen men också för att helt enkelt göra det möjligt för varje gruppmedlem att jobba mer på distans och på så sätt få mer gjort.

2.3 Versionshantering

Versionshantering sköttes via Git och ett repo på Github.com. Release skedde på master varje fredag. Annars utfördes arbetet mot dev-branchen, med flera olika featurebrancher.

2.4 Tester

Testmetoderna som användes var JUnit- och manuella tester.

3 Tidsåtgång

Del (tester inkluderade)	Adam	Carl	Erik	Johanna	Mike	Simon	**Totalt**
Scrum/Planering/Möten.	38h	38h	38h	38h	38h	38h	228h
Databas	0h	0h	0h	23h	34h	0h	57h
Navigation (sidpanel/tabbar)	10h	5h	0h	0h	0h	16h	31h
Påminnelser (tid/plats)	0h	0h	0h	0h	0h	61h	61h
Pomodoro	90h	0h	0h	0h	0h	0h	90h
GUI autogenerering	0h	0h	0h	0h	25h	4h	29h
Autogenerering	0h	0h	0h	8h	39h	0h	47h
GUI ny task eller event	0h	10h	57h	0h	0h	22h	89h
GUI nytt block	0h	0h	0h	0h		4h	4h
Redigera befintliga events/tasks	0h	0h	23h	16h	10h	12h	61h
Buggfixar/felsökning/städning	30h	40h	25h	0h	0h	28h	123h
Förslag på platser	0h	0h	12h	0h	0h	0h	12h
GUI Listvyer	0h	60h	23h	3h	0h	0h	86h
Totalt per person:	**168**	**153**	**178**	**88**	**146**	**185**	

4 Designbeslut

Genom hela projektet arbetade gruppen för att appen skulle ge ett professionellt och legitimt intryck på användaren, för detta användes många av Googles riktlinjer när det kom till det grafiska, med allt från designelement till färgval.

4.1 Sidopanel och Svepbara Tabbar

I början av projektet hade applikationen tre toppnivåer (Tasks, Events och Pomodoro), och man anade att det kanske skulle tillkomma ytterligare. Därför gjordes valet att använda en sidopanel som huvudsaklig navigation - dels på grund av att det idag är ett bekant navigationselement för de flesta Androidanvändare, dels för att Google själva rekommenderar användandet av en sidopanel när det finns fler än tre toppnivåer.

En bit in i utvecklandet av appen började sidopanelen verka överflödig. Autogenereringen skulle, i motsats till vad som från början var planerat, inte bli en egen separat del utan istället sammanfogas med vyn för att lägga till ett nytt event. Det totala antalet toppnivåer såg alltså inte längre ut att bli fler än tre, varför det blev svårt att motivera en sidopanel för huvudnavigationen. Panelen togs därför bort till fördel för svepbara tabbar istället.

Ytterligare en bit in i utvecklandet bestämdes att autogenereringen återigen skulle få en egen vy, och att listvyerna för tasks och events borde tillhöra en och samma toppvy då de är snarlika. Detta, i kombination med införandet av en "about-dialog", gjorde att applikationens antal toppnivåer nu var uppe i 4 stycken, varför sidopanelen återinfördes. Den översta toppnivån kallad "Home", innehöll därefter de båda listvyerna för tasks och events inkapslade i svepbara tabbar, medan Blocks(autogenereringen), Pomodoro och About (dialog med information om appen) nu var egna, helt skilda vyer.

4.3 Kontextuellt Menyfält

Applikationen använder en så kallad “Action bar” vilket är ett slags menyfält som alltid är synligt högst uppe i gränssnittet. Detta menyfält huserar knappar och alternativ för olika “actions” användaren kan göra, som till exempel att lägga till ett nytt event eller en ny task.

Menyfältet visar automatiskt rätt knappar för rätt vy, och döljer dem helt när sidopanelen sveps in från sidan. Således vet användaren alltid var alla relevanta knappar för den aktuella vyn finns och slipper onödigt letande.

4.4 Ingen kalendervy

I början fanns tanken att implementera en kalendervy, men det prioriterades bort dels på grund av tidsbegränsningen men också för att det var tillräckligt användbart att ha en Event-vy istället, som visade alla åtaganden i en lista. En annan anledning till kalenderns bortprioritering var att applikationen inte skulle ses som en fullfjädrad kalendar, utan snarare ett komplement till redan befintliga sådana.

4.5 “Long-click”-knappar

Vid ett skede i utvecklingen fanns önskan att implementera checkboxar för att snabbt och enkelt kunna ta bort flera tasks eller events, men eftersom de inte tillförde någon extra funktionalitet prioriterades detta bort. Det ansågs heller inte vara värt arbetet när det kom till user value/tidsåtgång. Istället implementerades edit- och deleteknappar som blir synliga vid en långtryckning på ett objekt i någon av listorna.

4.6 Databas

Eftersom inte databasen skulle synkroniseras med någon server var det smidigast att använda sig av Androids inbyggda SQLite databas. Till den byggdes en

SQLiteHandler som var väldigt omfattande. Detta för att göra det så enkelt som möjligt att implementera och att underhålla under projektets gång. I efterhand var det väldigt smart, eftersom det ytterst sällan fanns behov av att gå in och lägga till eller ändra funktionalitet i databasen.

5 Kända problem (och motiveringar)

Event vars tid har passerat visas i listan under rubriken Today. Detta är en medveten effekt eftersom produkten är en “få-det-gjort”-app och inte en kalender.

Om man trycker på knappen för att välja kategorier innan animationen för att byta mellan event- och taskvyerna är färdigt kommer den föregående vyns kategorilista att visas.

Kategoriknappen är synlig även om ingen eller bara en kategori finns att välja emellan. Om knappen initialiseras i de respektive fragmenten skulle detta gå att fixa, men då uppstår det nya problem med den globala navigeringen.

Om användaren, efter att manuellt ha stängt av appen (eller om appen försvinner från minnet på enheten), trycker på exempelvis “Done” på en gammal notifikation, kan appen krascha.

Om användaren har applikationen öppnad och samtidigt trycker på “Done”-knappen i en notifikation för en task uppdateras inte vyn, och tasken ser ut att vara kvar även fast den är borttagen från databasen.

Ifall man autogenererar ett schema så är det inte självklart att det man autogenererade är kvar på samma plats, i samma ordning, när man startar om applikationen. De autogenererade "eventen" sparas ej i databasen och autogenereras om på nytt vid varje start.

Om man går mellan pomodoroklockan och andra vyer mer än en gång stämmer inte pomodoroklockan. Om man pausar en timer och navigerar bort från pomodoroklockan kommer en timer att köras i bakgrunden. Byter man mellan många vyer är det hyfsat lätt att bugga pomodoroklockan. Dessa problem är kända, men Adam hann tyvärr inte åtgärda de innan tiden för projektet tog slut.

6 Diskussion

6.1 Scrum

Scrum är lägligt att använda i grupper om 5-9 personer som fokuserar på att utveckla mjukvara, och där ändringar kan behöva göras längs vägen. Eftersom Scrum är flexibelt och alla medlemmar i gruppen hålls uppdaterade via möten är det betydligt enklare att göra ändringar och att anpassa sig efter nya förhållanden.

6.1.1 Egen tolkning av scrum

En flow chart, som annars är en del av Scrum, valdes bort tidigt pga. att det ansågs vara svårt att göra en vettig tidsestimering på allt arbete då ingen i gruppen hade jobbat särskilt mycket med Androidutveckling tidigare.

Värt att nämna är även de faktiska rollerna för gruppens Scrum Master och Product Owner. I praktiken blev samtliga medlemmar i gruppen i princip lika ansvariga för alla delar av projektet, och majoriteten av de beslut som fattades togs tillsammans. Det faktum att gruppens Product Owner även var delaktig i utvecklingen av applikationen hade ingen negativ inverkan på projektet. Däremot var gruppens Scrum Master den medlem som drev morgonmötena framåt och såg till att Scrum följdes i någorlunda mån.

6.1.2 För- och nackdelar med Scrum

Scrum visade sig vara ett mycket effektivt verktyg för att hålla gruppen och projektet väl organiserat. Med regelbundna möten varje vecka fick alla en bra överblick av hur projektet fortskred, vilket i sin tur gav en effektivare och mer realistisk planering.

Mötena satte allt som ofta också igång matnyttiga diskussioner om saker som designbeslut, tidsplanering och prioriteringar av diverse funktioner i applikationen, vilket förstås gynnade projektet oerhört.

Att använda sig av en “product backlog” i kombination med veckovis skapade “sprint backlogs” visade sig också vara en mycket bra idé från scrumprocessen. Genom att tidigt i projektet skriva ner vilka funktioner och därmed vilka arbetsuppgifter projektet skulle komma att innehålla var det enkelt att varje vecka komma överens om vad som skulle göras, och av vem.

Tack vare det iterativa arbetssättet med nyuppsatta mål varje vecka var det enkelt att ändra utvecklingsriktning när det behövdes. Om någon i gruppen ansåg att en förändring av något slag var nödvändig diskuterades och utvärderades detta först på ett möte. Därefter gjordes nödvändiga omprioriteringar och projektet tog en ny riktning.

Scrum var, framförallt i början av projektet, ganska så tidskrävande. Men allteftersom projektet strukturerades upp och den grundläggande planeringen färdigställdes tog själva Scrum-biten mindre och mindre av arbetstiden och det blev tydligt att all planering tillslut lönade sig. Arbetet med applikationen kunde fortskrida nästan friktionsfritt resten av projektet, så tidsåtgången för Scrum var i slutändan inget egentligt problem.

6.2 Parprogrammering

Vad gruppen märkte när de började arbeta med parprogrammering var att arbetsmetoden är mycket lämplig att använda vid inlärning och mindre problemlösning så som buggfixning. Om en person har mer erfarenhet så delar denne sina kunskaper med sin partner och tack vare det extra perspektiv som två personer tillsammans besitter kan de lösa problem som en ensam person kanske stirrar sig blind på. Med hjälp av detta var det enkelt att komma igång med arbetet.

Men vad som är en av arbetssättets största styrkor blev för gruppen dess svaghet. Bristen av kunskap inom utveckling för Android ledde ofta till problem som ingen av parmedlemmarna kunde lösa. Detta slutade ofta med att en i paret hittade en lösning som denne arbetade på medan den andra fortsatte med en annan funktion. Även om denna funktion var en del i samma "git-branch" så märktes det snart att mer arbete blev utfört när alla arbetade var för sig.

En aspekt av parprogrammering som behölls var dock att två personer ofta arbetade på relaterade funktioner. Med detta upplägg var det enkelt att både arbeta ensam på distans men om problem uppstod var det även enkelt att hjälpas åt under scrummötet dagen därpå.

Slutsatser som dragits från gruppens försök med parprogrammering är att metoden passar oerhört bra när man ska lära sig något nytt eller bekanta sig med befintlig kod. Men då ingen kunskap finns att föra över och arbete inte kan ske gemensamt vid samma dator blir det betydligt mer tidseffektivt att arbeta var för sig.

6.3 Tester

För vissa områden användes JUnit-tester väldigt omfattande, som t.ex. Databasen och autogenereringen. Till de andra delarna av appen användes det däremot i begränsade mängder. Orsaken till detta var att när det gällde de förstnämnda områdena så var de mer eller mindre ett krav för att veta ifall de fungerade och för att kunna lägga ifrån sig dessa user stories. Prioritet på andra områden låg istället på att faktiskt se resultat (manuella tester) då dessa mer eller mindre hade en stor del "inbakad" grafiskt användargränssnitt redan från början.

6.4 Vad som kunde gjorts annorlunda

Vår version av Scrum kunde ha varit mer formell med mer tyngd på Scrum Master och Product Owner. Detta hade förmodligen lönat sig i större projekt där medlemmarna inte kan ha mycket kontakt med varandra, men i vårt fall hade det inte känts naturligt.

Testdriven utveckling var något problematisk att utföra. Vanligtvis skrivs tester innan koden själv, men på grund utav begränsad kunskap var det svårt att förutse vad som skulle skrivas överhuvudtaget. Istället skrevs kod och manuella tester utfördes för att "se om det fungerar".

Det hade blivit bättre ifall autogenereringen från början varit tänkt att sparas i databasen. Då hade applikationen i slutändan kunnat ha ett autogenererat schema som höll även när man startat om applikationen.