UMEÅ UNIVERSITY **28 oktober 2019**

Prototyputveckling för mobila applikationer
54801ht19

Projektrapport

ID5
Oscar Heimdahl
Linus Ström
Jonathan Smedborn
Albin Frick
Julia Yngve

Handledare
Stefan Berglund

Innehåll

1 Projektets mål

Målet för projektet var att utveckla en mobilapplikation som ska användas när man spelar frisbeegolf för att föra protokoll, se statistik för hur man och ens vänner spelat och hitta närliggande banor att spela på.

Applikationen ska fungera för iOS och android och använda sig av Google Maps kartfunktioner för att hitta banor, visa de olika hålen på banan samt ge användaren möjlighet att markera sina kast på kartan.

1.1 Projektteam

Projektteamet består av:

Oscar Heimdahl
Jonathan Smedborn
Albin Frick
Linus Ström
Julia Yngve

2 Process

I detta avsnitt kommer projektets process att diskuteras från början till slut, från LoFi- till HiFi-prototyp.

2.1 Projektintro

Första steget i projektet var att bestämma vilken sorts applikation vi ville skapa och vilken plattform den skulle utvecklas för. Gruppen redde ut det vid ett möte där frisbeegolf-idéen röstades fram och flutter, som kompilerar till både iOS och Android, valdes till språket vi ville använda och testa på.

2.2 Individuell LoFi

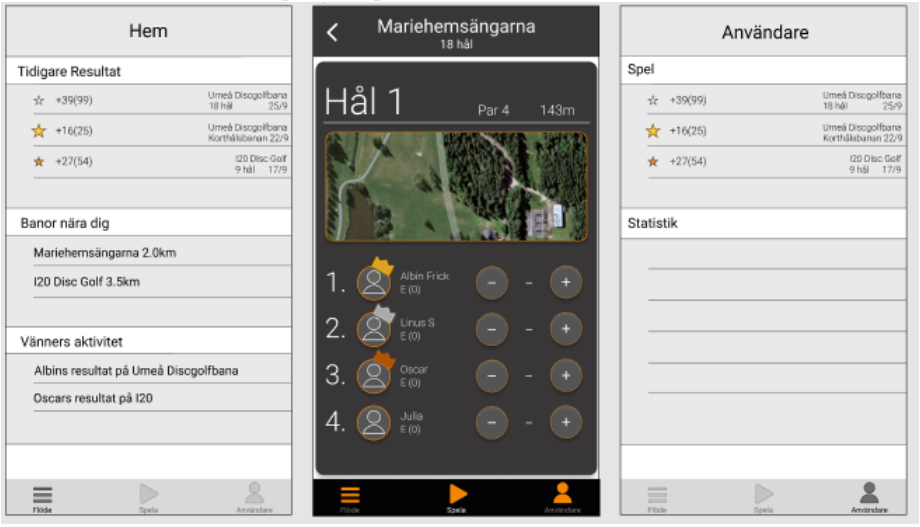
Efter att målet med projektet valts hade alla i gruppen uppgiften att individuellt ta fram en LoFi-prototyp på applikationen. Alla projektmedlemmar användartestade sedan sin prototyp och redovisade sin prototyp tillsammans med testresultaten för resten av projektgruppen.

2.3 MidFi-prototyp

Nästa steg i projektet var att skapa en gemensam MidFi-prototyp. Denna baserades på de bästa delarna i våra individuella LoFi-prototyper samt förbättringar och nya idéer på de delar som inte hade en bra lösning i någon av LoFi-prototyperna.

MidFi-prottypen skapades i Figma som är ett kollaborativt designverktyg där alla i gruppen samtidigt kunde jobba i samma projekt. Prototypen bestod av tre huvuddelar som kunde nås från en navigationsbar i botten av

skärmen. Användartabben där en användare kan se sin statistik, flödestabben där användaren kan se relevant information om sina vänner och speltabben där användaren kan starta spel och protokollföra sina kast.



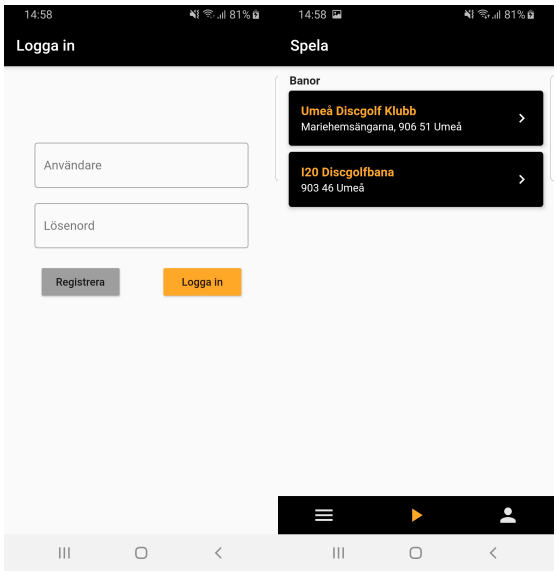
Bilden visar exempel på hur skärmarna för dessa tabbar såg ut i Midfi-prototypen.

2.4 HiFi-prototyp

HiFi-prototypen skapades med hjälp av Flutter och Firebase. Enbart Oscar hade en tidigare erfarenhet med Flutter. Innan själva arbetet började hade Oscar därför en workshop där han gick igenom hur flutter är uppbyggt och hur det fungerar så att resterande i gruppen skulle så snabbt som möjligt komma in i den nya miljön och språket.
Se Bilaga 1 för vilka krav vi satte för projektet.
Kraven var uppdelade i sex olika områden, alla utom ett område fullbordades. Det enda som inte blev fullt utfört var kraven på kartan, och detta var på grund av tekniska begränsningar.

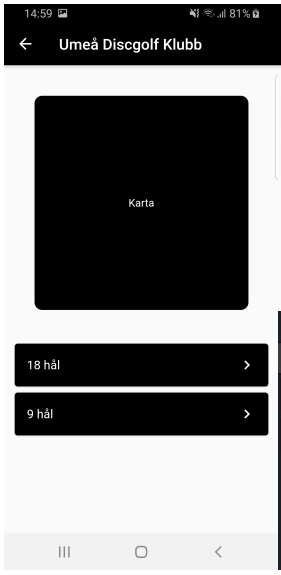
För att alla i gruppen skulle kunna samarbeta med samma kod använde vi oss av GIT. All historik för projektet går att hitta på här[1].

Bilderna nedan visar stora delar av den färdigställda prototypen.

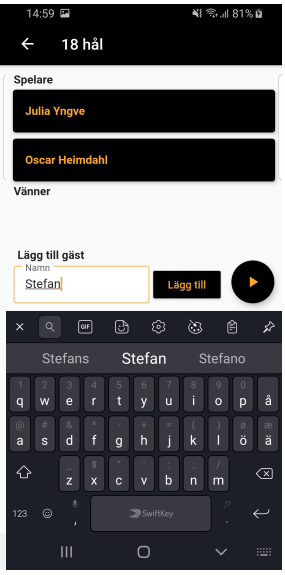


Figur 1: Logga in

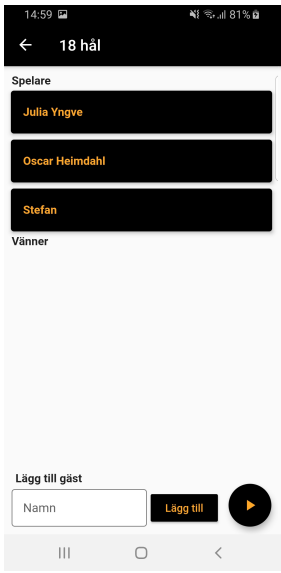
Figur 2: Spela



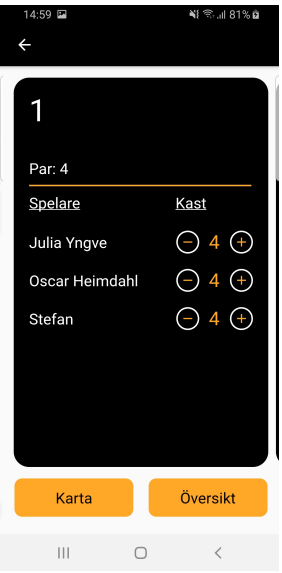
Figur 3: Välj bana



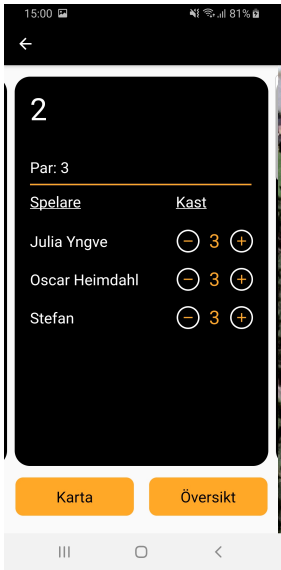
Figur 4: Addera spelare



Figur 5: Börja spela

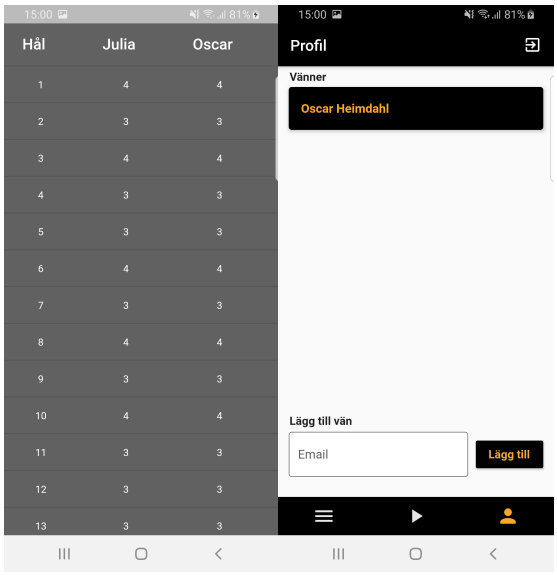


Figur 6: Scorekort 1



Figur 7: Scorekort2

Figur 8: Karta



Figur 9: Spelöversikt Figur 10: Profil

3 Verktyg

För att kunna skapa vår HiFi-protoyp behövdes en del verktyg. HiFi-prototypen är tillhuvudsak byggd på tre verktyg, Flutter, Firebase och Google Maps. I detta avsnitt kommer dessa tre att förklaras.

3.1 Flutter

Flutter är ett Software Development Kit (SDK) från Google som tillåter en att bygga applikationer till iOS och Android från en gemensam kodbas. Flutter använder sig av komponenter som kallas *Widgets*, som är lätta att skapa och återanvändas. Allt i Flutter är en widget, så saker som ett textfält, en icon och, en scrollbar vy är alla widgets. Många widgets är redan skapade av Google, vilket gör det snabbt och smidigt att komma igång.

3.2 Firebase

Google tillhandahåller en realtidsdatabas och backendservice för mobil apputveckling som heter Firebase.

Vår applikation använder sig av Firebase Authentication som är ett inloggningssystem som hanterar våra användare samt Firestore som är en Realtidsdatabas där vi lagrar data om våra användare och deras spelomgångar.

Firebase Authentication tillanda håller olika inloggningsfunktioner där vi endast använder oss av deras mail/lösenord-inloggning. Den gör att användare

kan skapa en användare med sin mail och binder användaren till ett user ID som används för att binda data i databasen till användaren.

Firestore är en dokumentbaserad realtidsdatabas som ligger i molnet. Vår databas är uppbyggd av tre ”Collectionsäv dokument.

- **Users** Denna kollektion innehåller dokument för olika användare. Varje dokument representerar en användare och identifieras med dens user ID. Har för och efternamn för användaren, dens vänner och inbjudningar till spel.
- **Courses** Denna kollektion håller alla olika banors information. Varje dokument representerar en bana och har dens namn, position och banans olika spår. Ett spår innehåller alla hålen på det spåret. Varje hål har information för vilket nummer den är, vad dess par är och geopositioner för dens tee och korg.
- **Games** Denna kollektion håller informationen för alla spelomgångar som startats. Innehåller datum, Course ID och trackname för att identifiera vilken bana och vilket spår spelomgången körs på samt alla spelare som är med i spelomgången. Varje spelare har en Map för att lagra dess kast för varje hål samt möjlighet att lagra geopositioner för kasten de gör.

3.3 Google maps

Det finns ett par olika paket till Flutter för att implementera kartor och vi valde google maps flutter”. Eftersom Flutter är ganska nytt så är vissa funktioner inte implementerade än, vilket resulterade i att vi fick slopa vissa funktioner eller lösa det på annat sätt. Alla funktioner som är med i kravspecifikationen är dock implementerade.

Kartan använder sig av geografiska koordinater för att rita ut bestående komponenter, såsom markörer och linjer, vilket skapas från datatyper som finns i paketet. I kartans objekt så finns det attribut för båda två, *markers* och *polylines*, som båda är av typen *Set*. Något som gör det omständigt att ändra markörer eller linjer, är att de måste tas bort och läggas till igen med en ny position eller färg.

3.3.1 Markörer

En markör är en ikon som syns på kartan och är klickbar. Dessa skapas som objekt, där position, ikon m.m. sätts. Ett problem som vi stötte på direkt var att vi ville ha text på markören som alltid syntes, såsom kastnummer samt kastlängd. En markör har en egenskap som är ett *InfoWindow*, men denna visas bara om användaren klickar på markören. Vi hittade aldrig ett sätt att få text direkt på kartan, så lösningen blev att skapa 11 olika ikoner för kastnummer, där kast över 10 bara visar ”...”. Funktionalitet för text direkt på kartan är en *requested feature*, så det bör komma inom en snar framtid.

3.3.2 Linjer

För att rita linjer på kartan så skapas ett *Polyline* objekt, som har egenskaper för bredd, färg m.m. För att rita ut en kontinuerlig linje så skapas ett enda objekt, med ett *Set* av positioner som linjen ritas ut emellan. Vi använde oss av två olika polyline-objekt, där ett av dom hade en *Pattern* egenskap för att skapa en streckad linje.

3.3.3 Spelarens position

I google maps paketet finns funktionalitet för att visa enhetens GPS-position, vilket automatiskt ritas ut på kartan. Dock så innehåller den standard-markören en stor ljusblå cirkel som indikerar osäkerheten i positionens precision, vilket var väldigt påträngande för vårat syfte. För att lösa det så använde vi oss av ett externt paket som heter *Geolocator*, som har funktionalitet för att lyssna efter förändringar i enhetens position. Vi skapade en markör utifrån en egen ikon och ritade ut den på nuvarande position.

3.3.4 Kamerans position

Eftersom alla hål har olika längd och riktning, så gick det inte att använda samma zoom på kameran. Vid uppdatering av kamerans position finns en funktion som heter *LatLngBounds*, vilket innehåller två motsatta hörnpositioner, nordöst och sydväst, där kameran lägger sig precis i mitten.

4 Applikationen

Applikationen består huvudsakligen av 3 flöden, en för varje vy under respektive tab i applikationens tab-bar.

4.1 Flöde

Flödet var tänkt att visa relevant information om spelarna senaste spel och ens vänners aktivitet. Idag så visar vyn information om alla tidigare spel som den inloggade spelaren har spelat. Denna informationen hämtas från samlingen games"i firebase och sorterar sedan ut de spel som berör den inloggade spelaren.

Om användaren väljer att klicka på ett av spelen som visas i vyn så presenterar applikationen ytterligare information om det valda spelet.

Denna vy har plats för mer funktioner, för att bli mer tänkt som det initiala målet. Till exempel: visa vänners aktivitet, mer specifik information om spelarnas spel, rekordlångt kast annan information som kan vara roligt för användaren.

4.2 Spela

Vyn för att spela är den mest invecklade av vyerna. Spela-vyn låter användaren välja bana utifrån de tillgängliga i databasen, har kan användaren också acceptera en inbjudan från en annan användare.

När användaren väljer en bana kommer användaren till en sida där den får mer information om banan. Denna sida är tänkt att visa en kartvy över banan och annan intressant info. Idag är denna vy ganska tom och har en placeholder för kartan, något som kan utvecklas vidare i framtiden. När spelaren har valt vilket spår den vill spela på banan så visas vyn för att bjuda in spelare. Denna vy fungerar idag som tänkt men är känslig mot förändringar i databasen, då den är starkt beroende av den.

Till sist så kommer användaren till själva spelvyn. Här kan användaren i realtid ändra inbjudna spelares kast och öppna kartan för att markera hur långt som kastats för varje hål. Inbjudna spelare har också möjlighet att nå denna skärm för att interagera med samma vy. Detta är möjligt genom att i Flutter använda widgeten StreamBuildersom visar informationen från databasen då den uppdateras. Och då användaren ökar eller minskar antalet kast så uppdateras databasen.

Några saker kan vara bra att förbättras och utvecklas vidare med spelvyn är:

- **Realtidsuppdateringen** och uppdateringen av databasen är idag väldigt beroende av en snabb uppkoppling. Om användaren adderar ett kast på en spelare och det tar tid att uppdatera databasen så kommer inte användare se att kastet uppdateras direkt. En lösning som hade kunnat implementerats är att direkt i applikationen uppdatera kasten och sedan uppdatera databasen i bakgrunden.
- **Inbjudan** Mycket information ackumuleras när spelaren manövrerar mellan vyerna, denna information används för att konstruera spel-vyn. För att smidigt kunna ge samma förutsättningar till spelvyn för en inbjuden person, så skickas en stor del av den ackumulerade informationen till databasen. Mycket av denna informationen är överflödig och hade kunnats städas upp.

4.3 Användare

I denna vy kan användaren lägga till och ta bort vänner. Likt flödesvyn så har denna plats för fler funktioner. Som att visa mer intressant information om användaren.

Referenser

[1] https://github.com/AlbinFrick/discgolf

[2] https://flutter.dev/

[3] https://firebase.google.com/

5 Bilagor

UMEÅ UNIVERSITY **12 september 2019**

Prototyputveckling för mobila applikationer
54801ht19

Kravspecifikation

ID5
Oscar Heimdahl
Linus Ström
Jonathan Smedborn
Albin Frick
Julia Yngve

Handledare
Stefan Berglund

1 Krav

1.1 Allmänt

- Alla spelare ska inte behöva ha appen eller vara medlemmar.
- Ska fungera på Android(kit-kat eller senare) och iPhone(iOS 9 eller senare).
- Spelare ska kunna påbörja sin omgång på valfritt hål och dokumentationen ska hantera det.

1.2 Funktioner

- Användare ska kunna skriva in antalet kast för varje hål.
- Användare ska kunna se information kring varje hål på banan i en vy.
- Användare ska kunna importera ett scorekort från kartan och fylla i det.
- Användare ska kunna göra ett eget scorekort.
- Användare ska kunna lägga till andra spelare i ett scorekort.
- Om andra spelare på scorekortet är registreade användare ska ha tillgång att se och redigera scorekortet.

1.3 Användare

- Användaren ska kunna dokumentera sina kast på banan.
- Användare ska kunna skapa en profil där information lagras.
- Användare ska kunna ta bort sin användare och byta lösenord.
- Användare ska kunna se historik från tidigare spel.

1.4 GPS

- Banor ska kunna hittas och väljas utifrån en karta.
- Kastlandningar ska kunna dokumenteras på karta med GPS.
- Antal kast per hål ska kunna dokumenteras med hjälp av GPS.
- Startposition och korg för varje hål ska kunna visas på kartan.

1.5 Statistik

- Vid val av bana ska information kring varje hål på banan laddas in i scorekortet.
- Dokumentering av kast ska fungera för stora grupper av spelare (10+).
- Dokumenteringen ska kunna utföras av alla personer i gruppen för alla.
- Statistik för kast på hål och banor.

2 Ordlista

- Scorekort = Tabell med information om varje hål respektive hela banan”