CrowdPlay

ITSMAP-01 Eksamensrapport

Gruppe 02

201404118	Anders W. Birkelund
201271001	Rune D. Rask
20080899	Khaled G. Edwan
201405166	Jonas R. Hartogsohn

Indhold

Indledning

CrowdPlay er en app, der lader alle deltagende gæster ved en fest være med til at bestemme den musik, der bliver spillet. En smartphone / tablet kobles til det pågældende anlæg, og herfra vil der blive spillet de sange, som gæsterne vælger at tilføje til play-listen. I løbet af festen kan gæsterne up-vote sange, de synes godt om. Jo flere up-votes en sang får, desto højere kommer den op ad play-listen og vil derved blive spillet tidligere end de øvrige sange. For hvert up-vote en sang får, modtager den gæst, der tilføjede sangen ét point. Dette skaber et konkurrenceelement der giver brugeren et ekstra incitament til at bruge appen. Samtidig vil det gennem festens levetid være muligt at følge med i, hvilken gæst der er bedst til at tilføje sange, der rammer bredest hos publikum.

Kravspecifikation

Til projektet var der på forhånd givet nogle krav til den applikation (app), der skulle udvikles. Den pågældende app skulle inkludere følgende formelle krav:

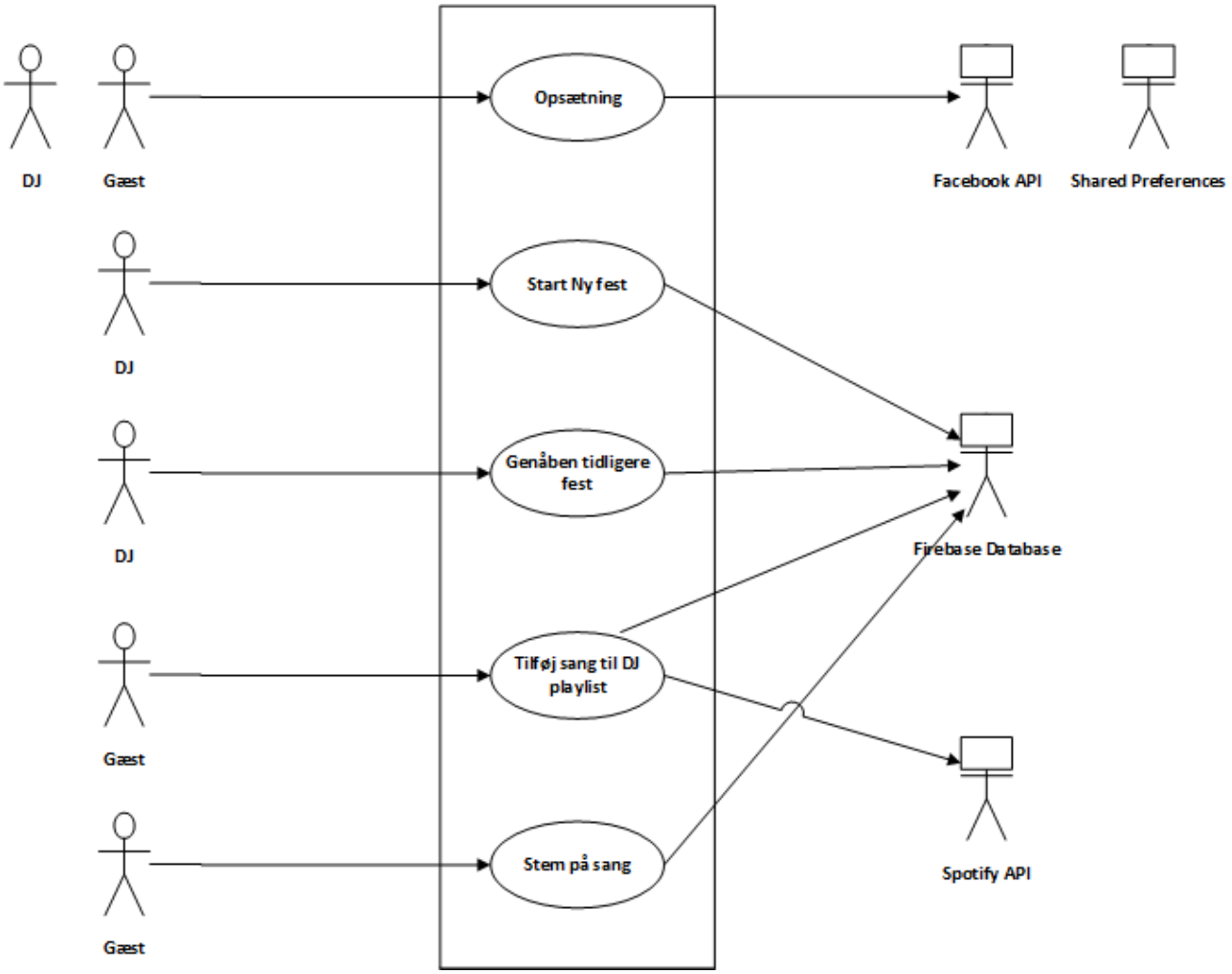
- Mindst to 'Activities'
- Brug af 'Intents' til at sende data mellem komponenter
- Persistering af data gennem 'SharedPreferences' og/eller en SQLite database
- Nogen form for kommunikation ved brug af internet, Bluetooth og/eller WIFI
- Mindst én 'Service'
- Brug af Asynkron processering
- Brug af relevant 'Resource externalization'
- Brug af 'Layouts' der adapterer til mindst to forskellige skærmstørrelser (telefoner og tablets)
- Understøttelse af (mindst) dansk og engelsk som sprog

Desuden opstillede projektgruppen selv nogle krav til app'en, der skulle definere den grundlæggende opbygning af denne. Disse krav er som følger:

- Firebase skal benyttes som database for applikationen
- Firebase skal desuden benyttes som kommunikationsredskab mellem enheder
- Et eksternt API skal benyttes til at hente data omkring sange
- Spotify SDK skal benyttes til afspilning af sange i applikationen

Use Cases

For at klarlægge de overordnede funktioner af app'en samt illustrere den typiske anvendelse, er der udarbejdet 5 use cases, der beskriver fremgangsmåden for at benytte app'en som DJ og gæst. Herunder ses et use case-diagram, som viser aktørerne i systemet.



Figur 1 Use Case Diagram, viser aktører for de 5 use cases

Aktører

- **DJ**'en er den person der opretter festen i app'en og sætter sin smartphone / tablet til et anlæg.
- **Gæst**en repræsenterer de øvrige deltagere ved festen, som tilføjer nye sange til playlisten.
- **Facebook API** er app'ens interface til Facebook, hvorigennem brugerne logger ind.
- **SharedPreferences** er telefonens lokale placering til at gemme login-data.
- **Firebase Database** er den database, der benyttes til lagring af data samt synkronisering mellem de forskellige enheder.
- **Spotify API** er app'ens interface til Spotify, der benyttes til at søge efter kunstnere og sange.

Use Case 1

Use Case 1 beskriver, hvordan brugeren laver en opsætning af app'en. Dette indebærer, at give app'en forskellige tilladelser samt logge ind med Facebook. Use Casen er fælles for både DJ og Gæst-aktører, og disse vil derfor betegnes "Bruger" i denne Use Case.

USE CASE 1 – OPSÆTNING

Formål - Opsætning af app
Aktører: DJ eller Gæst

Forudsætninger

- Bruger har appen CrowdPlay installeret på sin Android smartphone med API 15 eller højere.
- Brugeren har en personlig Facebook-profil.
- Brugeren har ikke Facebook-appen installeret på sin smartphone.
- Brugerens smartphone har internetforbindelse.
- Brugerens smartphone kan finde sin location via enten GPS eller nærliggende WIFI.

Hovedscenarie

1. Brugeren starter appen ved at trykke på appens ikon.
2. Brugeren tillader at appen bruger smartphonens "location" ved at trykke på knappen "Allow".

2.1 Brugeren tillader ikke, at appen bruger smartphonens "location" ved at trykke på knappen "Deny".
Se Undtagelser pkt. 1.

3. Brugeren indtaster sin e-mail til Facebook. Brugeren indtaster sin password til Facebook. Brugeren trykker på "log in" knappen.
   Brugeren trykker på OK knappen

3.1 Brugeren trykker på cancel i en af mulighederne i pkt. 3
Se Undtagelser pkt. 1.

Undtagelser

1. Appen lukkes

Use Case 2

Use Case 2 beskriver, hvordan man som DJ opsætter en ny fest, så gæster efterfølgende kan tilgå denne, for derefter fylde den tilhørende playliste.

USE CASE 2 – START NY FEST

Formål - Starte en fest
Aktører: DJ

Forudsætninger

- Use Case 1 Hovedscenarie er gennmført af DJ
- Brugeren har et premium account på Spotify.

Hovedscenarie

1) DJ trykker på *"DJ-ikonet"* på forsiden
2) DJ indskriver ønsket navn til festen i *"Name"*-tekstfeltet
3) DJ indtaster ønsket kode til festen i *"Password"*-tekstfeltet
4) DJ trykker på *"START NEW PARTY"*-knappen
    a) Undtagelse 1
5) DJ fortæller sine gæster navn og adgangskode til *CrowdPlay*-festen
6) DJ venter på at gæsterne tilmelder sig festen og tilføjer sange

Undtagelser

1. DJ har ikke skrevet et navn i *"Name"*-tekstfeltet, og får ikke lov til at oprette ny fest

Use Case 3

Use Case 3 beskriver, hvordan man som DJ genåbner en gammel fest, med dens tilhørende playliste. Gæster kan derefter tilgå festen og tilføje flere sange.

USE CASE 3 – GENÅBN TIDLIGERE FEST

Formål - Genåbne tidligere *CrowdPlay*-fest, for at genbruge festens playliste
Aktører: DJ

Forudsætninger

- Use Case 1 Hovedscenarie er fuldført
- DJ har et premium account på Spotify.
- DJ har tidligere brugt app'en til at holde fest

Hovedscenarie

1. DJ trykker på *"DJ-ikonet"* på forsiden
2. DJ trykker på den fest, han ønsker at genåbne på listen over tidligere fester
3. DJ fortæller sine gæster navn og adgangskode til den genåbnede *CrowdPlay*-fest

Use Case 4

Use Case 4 beskriver, hvordan man som gæst tilmelder sig en fest, for derefter at tilføje nye sange til festens playliste.

USE CASE 4 – TILFØJ SANG TIL DJ PLAYLIST

Formål - Tilføje en sang til en DJ playliste
Aktører: Gæst

Forudsætninger

- DJ har gennemført Use Case 2 eller 3
- DJ har ikke lukket DJ Activity
- Gæst Har password til Party fra Use Case 2 eller 3

Hovedscenarie

1. Gæst trykker på "Gæst" ikonet
2. Gæst finder fest på kortet og trykker på marker
3. Gæst trykker på infovindue med festens navn
4. Gæst indtaster password
5. Gæst trykker på "OK" knappen

5.1 Gæst trykker på "Cancel" knappen. Se undtagelser pkt. 1

6. Gæst swiper til højre, eller trykker på "SEARCH" menuen
7. Gæst indtaster en søgetekst i "Search" tekstfeltet
8. Gæst trykker på "søg" knappen
9. Gæst trykker på "tilføj" knappen på den ønskede sang

9.1 Det ønskede nummer er ikke på listen, eller der er ingen numre på listen. Se Undtagelser pkt. 2

Undtagelse:

1. Password dialog lukker. Infovindue for fest lukker - start fra pkt. 3.
2. Gæst starter igen fra pkt. 7. Indtaster mere kortfattet søgetekst.

Use Case 5

Use Case 5 beskriver, hvordan en gæst stemmer på en allerede tilføjet sang, på den tilmeldte fests playliste.

USE CASE 5 – STEM PÅ SANG

Formål - Stemme på en sang så den bliver afspillet tidligere af DJ-smartphone
Aktører: Gæst

Forudsætninger

- En DJ har startet en fest jf. Use Case 2 eller Use Case 3.
- Gæst har gennemført Use Case 4
- Gæst eller an anden Gæst har gennemført Use Case 4, mindst 2 gange

Hovedscenarie

1. Gæst trykker på "Playlist" menuen
2. Gæst trykker på "Upvote" ikonet på den sang Gæst ønsker spillet tidligere.

11.1 Gæst har allerede trykket på "Upvote" ikonet for den valgte sang tidligere, og den valgte sang er ikke blevet spillet i mellemtiden. Se Undtagelser pkt. 1

Undtagelser

1. Sangen får ikke flere stemmer.

Design

Før selve programmeringen gik I gang, blev nogle grundlæggende tanker omkring design og opbygning gennemgået i gruppen. Først var det antaget, at der skulle bruges 3 'Activities' (jævnfør gruppens synopsis), men i løbet af udviklingen blev det tydeligt, at det var mere logisk at fordele noget ansvar ud i flere Activities og klasser. Desuden var det først tiltænkt at benytte en service til afspilning af sange, men dette viste sig at være unødvendigt, da Spotify's SDK selv opretter en service, der kører med en musikafspiller. Nedenfor ses det endelige komponent-diagram for systemet.

Component Diagram

Figur 2 Componentdiagram over vigtige elementer i for use cases 2-5

Component diagrammet viser de mest væsentlige komponenter i systemet. Diagrammet viser komponenter der er vigtige for Use Cases 2 til og med 5.

Hvad der *ikke* vises i diagrammet

Systemet er i praksis mere komplekst end vist på diagrammet, og består af flere komponenter f.eks recyclerviewadaptere, JSONparser, SharedPreferencesConnector og DTOer.

Desuden vil brugeren ved første brug af appen blive bedt om at lade appen bruge ens ”public facebook profile”. Hvis dette ikke accepteres, vil appen afsluttes. I tilfældet hvor brugeren accepterer vil disse data om brugeren blive gemt i Shared Preferences og blive brugt i alle klasser der implementerer FireBase Database komponentet.

Hvad der vises i diagrammet

Komponenterne der er vist er hhv. GUI elementer – activities, fragments og dialogbokse, samt databaseelementer og et komponent til at kommunikere med Spotifys API.

- De grønne pile viser flowet imellem appens activities.
- Blå pile viser afhængigheder mellem komponenter.
- Gule pile viser at en klasse bruger Firebase Database til at sende modtage data.

Fra Mainactivity får man som bruger muligheden for at for at vælge at være DJ eller gæst.

DJ'en opretter festen og når der tilføjes sange til festen vil de automatisk afspilles i rækkefølge efter flest votes.

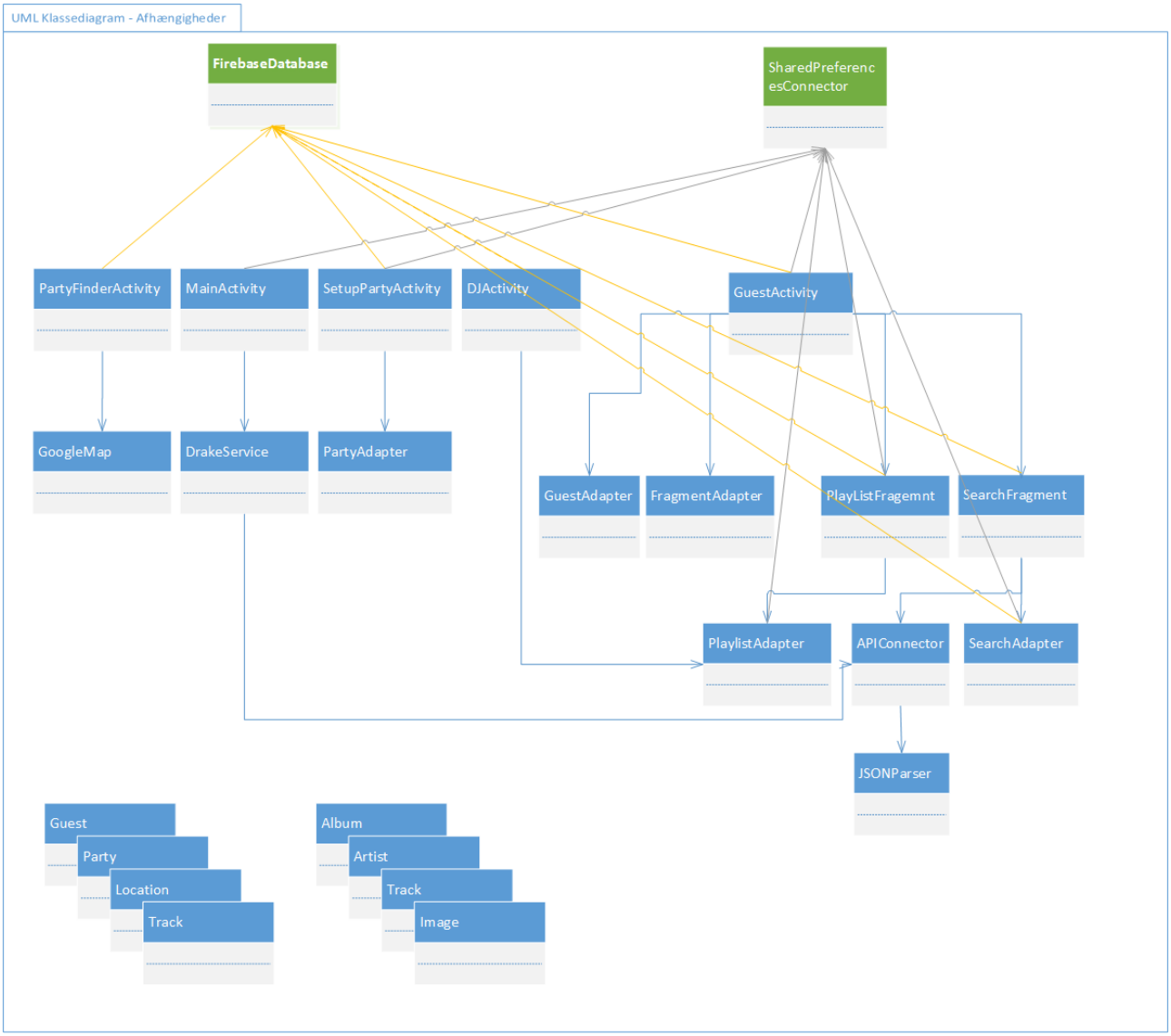
Gæsten finder festen på maps, og logger ind med festens password. Efter login har gæste mulighed for at tilføje sange fra spotifys bibliotek til DJ-telefonens playliste. Samtidig kan gæsten stemme på andre gæsters tilføjede sange.

Applikationen er bygget op omkring Google’s Material Design principper, med tre primære farver og tre sekundære farver, og niveauforskel i visuelle elementer.

UML Klassediagram

UML klassediagrammet viser afhængigheder i systemet. DTO klassen spotifyModel er ikke komplet, men viser de vigtigste properties i forhold til systemet. SpotifyModel indeholder i alt 13 klasser.

Det ses at Activities er øverst i klasse-hierakiet da der ikke går afhængigheder imod Activities. Mange af klassern anvender FirebaseDatabase til at persistere og indhente data. Samtidig anvendes SharedPreferencesConnector klassen af mange af klasserne til at få adgang til brugerens "public Facebook profile" informationer. Adapter klasserne fungerer hovedsageligt som ViewModels i MVVM mønstret, hvori data bliver modelleret til fremvisning på GUI. SearchAdapter klassen bryder dog med MVVM principperne, da der skrives direkte til databasen fra denne klasse.



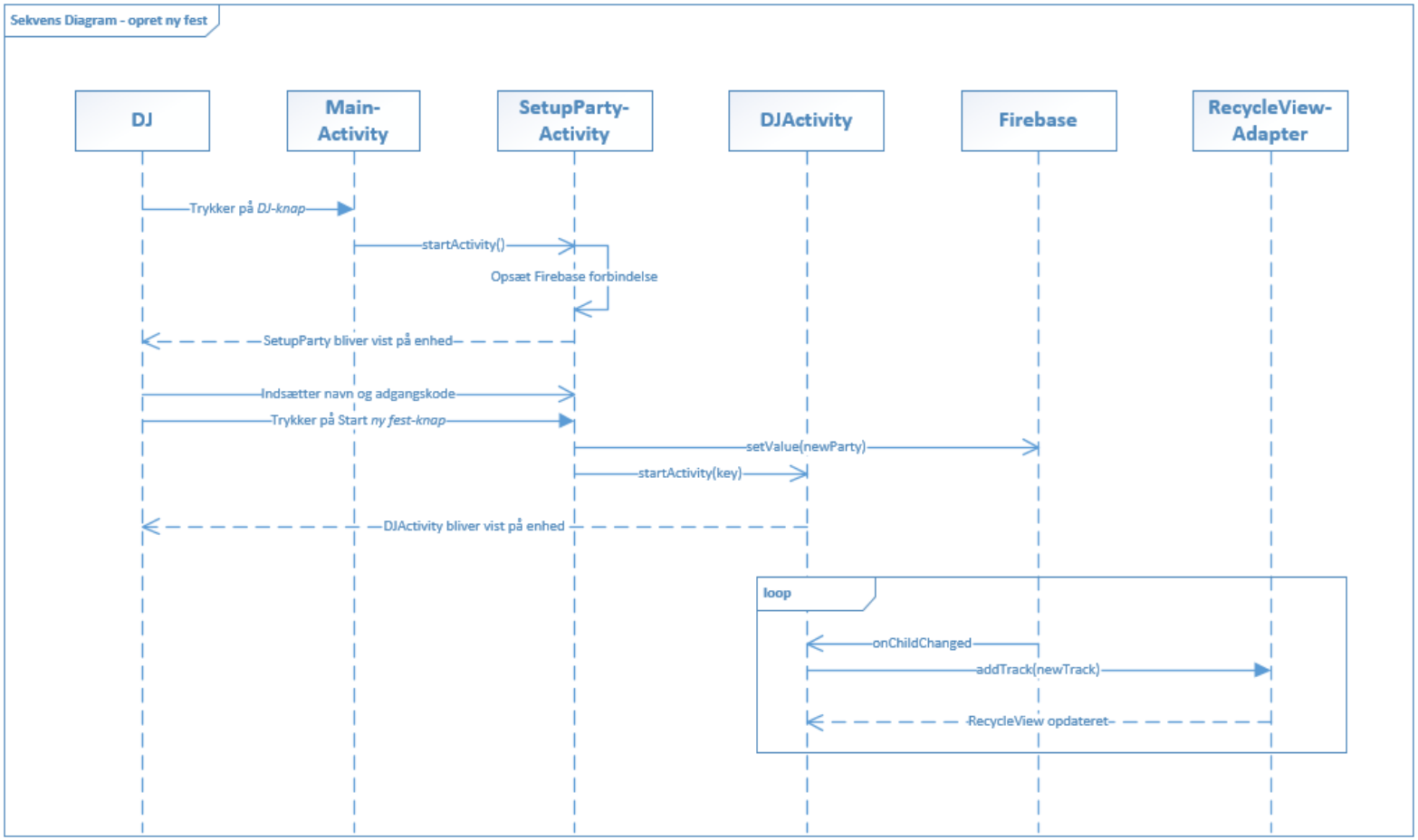
Figur 3 UML Klassediagram med beskrivelse af afhængigheder i systemet

Sekvensdiagrammer

På de følgende sider findes to sekvensdiagrammer, der beskriver to grundlæggende funktioner i app'en for henholdsvis DJ og Guest. Det første diagram beskriver forløbet i koden under use case 2 (Opret ny fest) og det næste beskriver forløbet i use case 4 (Tilføj sang til DJ-Playlist). Sekvensdiagrammerne fortæller overordnet om metodekald og forløbet i koden i disse scenarier.

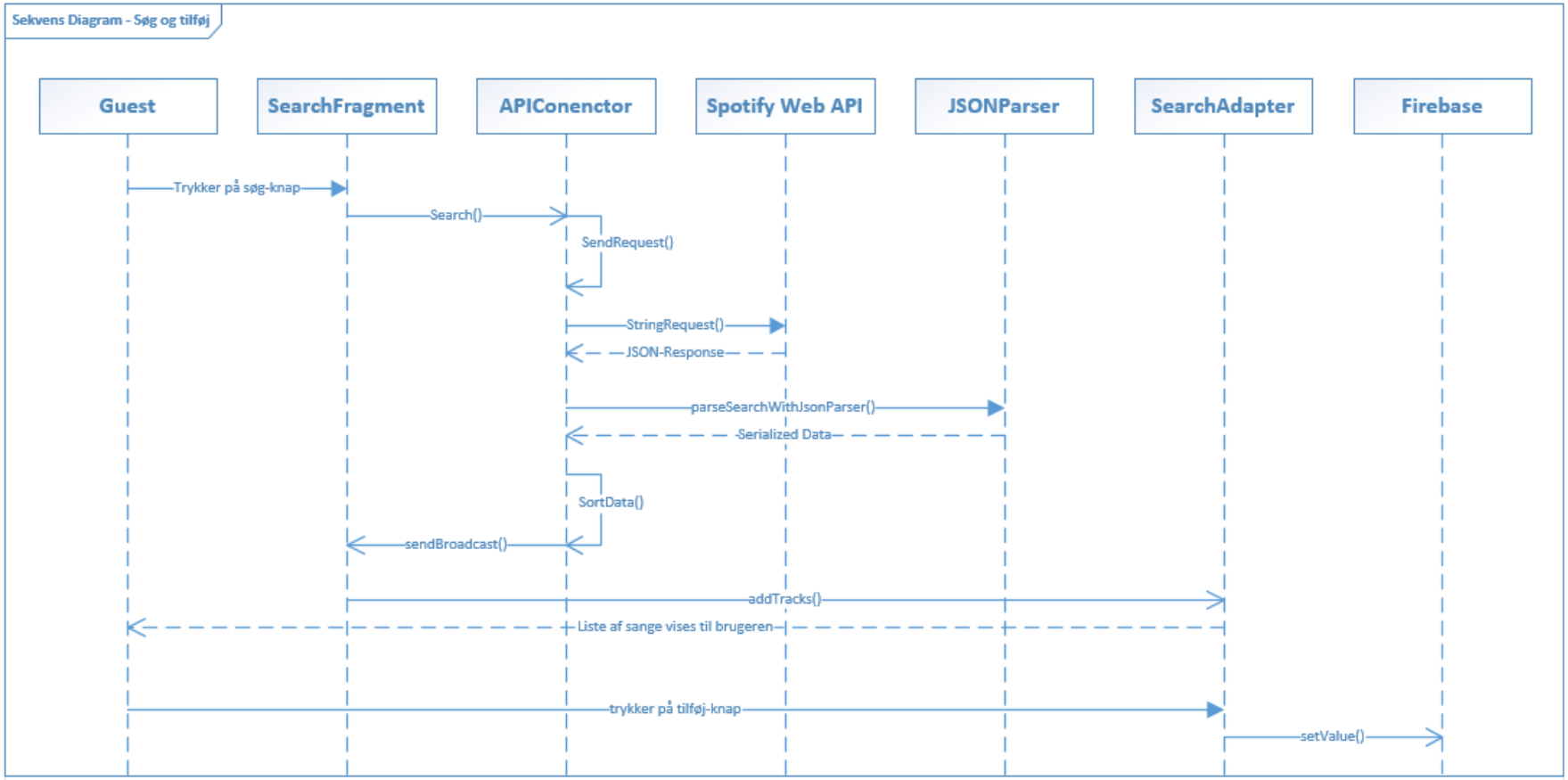
Sekvensdiagram (Start ny fest)

Nedenstående diagram, tager udgangspunkt i Use Case 2 – Start ny fest, hvor DJ'en ønsker at opsætte og publicere en fest. Gæsterne kan derefter tilslutte sig den oprettede fest. Sekvensdiagrammet slutter i en uendelig løkke, hvor den lokale playliste bliver opdateret hver gang der kommer ændringer på Firebase databasen. Disse ændringer vil forekomme hver gang en gæst enten tilføjer en ny sang, eller stemmer på en af de allerede tilføjede sange.



Sekvensdiagram (Søg og tilføj)

Nedenstående sekvensdiagram beskriver metodekaldene i situationen, hvor en vil søge efter en sang og tilføje denne til playlisten. Selve brugergrænsefladen ligger i et fragment, da der i GuestActivity er mulighed for både at se playlisten og søge efter nye sange. APIConnector-klassen sender requested til Spotify's Web API via Volley og får derved søgeresultater tilbage. SearchAdapteren er en Recycler-adapter, der benyttes af Search-fragmentet til at lægge sange ind i et Recycler-view og derved få dem vist til brugeren. Når den valgte sang i sidste ende tilføjes til Firebase, vil de andre brugere blive opdateret omkring dette, da alle enhederne er synkroniseret via denne database.



Konklusion

Slutteligt kan det konkluderes, at målene for projektet er nået. Både de formelle krav og gruppens egne krav til projektet er opfyldt, og alle ønskede features for app'en er implementeret til et funktionelt niveau. App'en har den ønskede 'feel', der er kommet til udtryk ved brug af blandt andet Fragments og Custom Recycler-views, hvilket tillægger brugeroplevelsen et mere professionelt touch.

Gruppen ønskede fra start at få mere erfaring med, hvordan flere enheder kommunikerer med hinanden f.eks. via internettet. Dette er der opnået god erfaring med, da essensen af app'en er, at alle tilkoblede enheder er synkroniseret med hinanden.

Hovedsageligt har processen været yderst lærerig, da gruppen har arbejdet fokuseret med at udvikle en applikation fra idéstadie til et umiddelbart færdigt produkt.

Arbejdsplan

Nedenfor ses et overblik over arbejdsplanen for projektet, hvor ansvaret for de forskellige områder er uddelegeret til de individuelle gruppemedlemmer:

- **Idé-udvikling** Alle
- **Firebase** Anders, Rune og Jonas
- **Layout** Anders, Rune og Jonas
- **Implementering** Anders, Rune og Jonas
- **Facebook API** Anders
- **Google Maps** Anders
- **Spotify SDK** Rune
- **Spotify Web API** Jonas
- **Tablet-layout** Khaled