Konsept: Ofte er hotellresepsjonen en møteplass før man drar ut, da særlig når gjestene venter på at de alle har blitt en samlet gruppe. Mange av gjestene er også turister, og har den kjente problemstillingen om å ikke vite helt hvor de skal dra/hva de skal gjøre. Applikasjonen min skal prøve fylle dødtiden de har mens de venter, samtidig som at det skal være et verktøy for å dra steder der hotellet anbefaler. Dette hjelper også å avlaste resepsjonistene, da de vil få færre spørsmål angående aktiviteter fra gjestene. Applikasjonen min skal benyttes på ett nettbrett som er montert i resepsjonen. Jeg har tatt utgangspunkt i at hotellet ligger der Thon Hotel Opera befinner seg for øyeblikket, men har valgt å ta fra avstand fra spesifikke bedrifter og har valgt det generiske navnet «Hotel» som hotellnavn.

Løsningen: Løsningen er tilpasset en Google Nexus 9 med google play services installert, bruker engelsk tastatur og har stabil tilkobling til internett. Løsningen har minimum SDK 21(Lollipop), men siden løsningen ikke skal på mange forskjellige enheter kunne den fint ha vært langt høyere. Løsningen har blitt utviklet med Genymotion som emulator da jeg ikke har tilgang til en Android-telefon og Android studio sin emulator håndterer kart veldig tregt, og anbefaler at du gjør det samme hvis du opplever det samme.

Generelt om produktet: Generelt føler jeg koden er fleksibel, skalerbar, lett forståelig og at grensesnittet er intuitivt. Hjelpemetoder, smarte løkker, indre klasser og entiteter er blant det jeg føler hever kodekvaliteten. Det er noe småpirk som må fikses på og det mangler tester, og er det jeg ville ha satt fokus på ved eventuell videre utvikling. Utenom det føler jeg at jeg gått forbi minstekravene og prøvd med på mye spennende og som har vært med å heve oppgaven. Jeg har valgt å fjerne statusbaren (som viser Wifi, batteri etc.) da nettbrettet appen kjører på vil være montert med strømtilførsel. Jeg har vært veldig på gjerdet når det gjelder å fjerne navigasjonsbaren (knappene nederst på fleste androidtelefoner), siden nettbrettet kun skal ha ett formål. Samtidig vet jeg det er veldig mange måter å komme seg ut av applikasjonen som jeg ikke klarer å ta høyde for, og har derfor valgt å la baren bli da mange brukere er godt vant med den. Det kan jo også være et pluss hvis de ønsker å google informasjonen som har blitt gitt til dem.

RootActivity: Det finnes kun en activity i applikasjonen. Layouten har et basefragment som byttes ut med andre fragments etter som de blir kalt på. Dette hjelper veldig med å spare på ressurser, da det er mindre tungt å bytte ut fragments enn å åpne nye activities med intents. Det er også lettere å kommunisere mellom en activity og et fragment enn det er mellom activities. Utenom fragments så har RootActivity en statisk header som er til stede på alle sidene og er den som står for navigasjon mellom fragmentene.

FrontPageFragment: Forside og standardside for applikasjonen. Målet ved siden er at man lett kan se hva appliksjonen tilbyr uten å måtte gå helt bort til nettbrettet og trykke seg rundt. Initielt hadde jeg en plan om at det skulle være en intern timer på applikasjonen som ville åpne forsiden etter en periode. Den kastet jeg fra meg da det ville vært masse kode for en veldig spesifikk funksjon, som ikke nødvendigvis hever brukeropplevelsen.

MapsFragment: Målet med siden er å vise brukere steder hvor de kan dra, med filtre etter humøret deres. Det blir opp til hotellet å velge steder de anbefaler, og er en god plattform for avtaler med andre bedrifter for hva som skal vises. Det er lagt til rette for å kunne legge

til mange fler markere, men jeg holdt meg til fire per kategori da jeg føler at jeg har fått vist funksjonaliteten på en god nok måte. En eventuell utvidelse jeg ikke har implementert hadde vært å bruke google sin veiviser for å vise ruten til de forskjellige punktene, men jeg utelot det da jeg tror få vil memorisere/skrive ned ruten de har blitt tilbudt, men heller bruke egne midler. Jeg har ikke benyttet meg av MapsFragment da vi allerede befinner oss i et fragment og fragment i fragment er en dårlig konvensjon.

WeatherFragment: Målet med siden er å gi et kjapt overblikk på hvordan vært vil være utover dagen. Designet er ikke det vakreste (da særlig med veldig lav-res bilder), men er funksjonelt. Fragmentet lagrer alle sine views i to arrays og sender det til WeatherManager for at det skal populeres der. En svakhet med siden er at den laster inn data fra APIet hver gang den åpnes og burde egentlig lagres internt og oppdateres gitte mellomrom. Siden APIet har en grense på maks 60 kall i minuttet var det ikke en prioritet for meg.

WeatherManager: Hjelpeklasse som skal hente data fra et eksternt APIet og populere viewsene den fikk fra WeatherFragment med værdata. Klassen arver fra AsyncTask og implementerer derfor metodene dens for at klassen skal hente data asynkront. Mens dataen lastes blir et lastevindu presentert. En problemstilling jeg nådde var at dataen jeg hentet ble hentet så kjapt at vinduet bare blinket, noe som fikk det til å virke at det var noe galt med appen. Alternativet var å fjerne hele vinduet, men da fryser appen hvis hentingen går sakte. Løsningen jeg landet på var å sette Thread.sleep i et halvt sekund, nok til at vinduet vises hvis det skulle gå fort, samtidig som at den er tilstede når det går tregt. Dette er selvsagt ikke en god løsning, og jeg har det kun med for å vise at jeg kan benytte meg av lastevinduer og asyncmetoder.

APIet jeg benytter er http://openweathermap.org/forecast5, hvor nøkkelen står i mitt navn. Dette er en gratis tjeneste som gir værdata i form av JSON eller XML hvor jeg valgte JSON. Siden jeg benytter gir værvarsler med 3 timers mellomrom hvor jeg selv kan velge hvor langt frem i tid jeg vil, jeg valgte nåtid, om 3 timer og om 6 timer da jeg tenker det er det mest praktiske for gjestene. Siden jeg henter data fra: http://api.openweathermap.org/data/2.5/forecast?id=3143244&units=metric&appid=06dfc6fc98ae6066c0e8b59e582b65cd.

Weather: Standard POJO som WeatherManager benytter for lagring av data og enkel populering av views.

RecommendationFragment: Målet ved siden er at brukere kan legge inn egne anbefalinger til andre gjester ved hotellet. Brukere kan da stemme på anbefalingene de syntes var bra/dårlig. Brukere har også muligheten til å sortere etter det nyeste eller det med best score. Siden benytter seg av et RecyclerView og hvis det blir veldig mange anbefalinger vil den hjelpe veldig med å holde kostbarheten nede, særlig siden hver rad har to bilder i seg. En svakhet dette medbrakte er at det er veldig vanskelig å direkte påvirke hva som skjer i hver rad og det måtte en del konfigurering i PostsAdapter for at ting skulle fungere som tiltenkt. Selv med en del konfigurasjon fikk jeg ikke til alle funksjonene jeg ønsket, da jeg hadde ordinært en plan om at det er en begrenset mengde man kan stemme i et visst tidsrom. Løsningen som den er nå kan man «spamme» knappene så lenge man vil. Jeg har

tatt høyde for at det kan være masse tulleposter fra gjestene. Da kan de ansatte velge hvilke poster de kan fjerne ved å skrive inn et passord og long-clicke på postene de vil fjerne.

PostsAdapter: Dette er en hjelpeklasse for å populere RecyclerViewet med data den har fått fra RecommendationFragmentet. Den gjør det ved å arve fra RecyclerView.Adapter<PostsAdapter.ViewHolder> og bruke metodene den overstyrer derfra. Klassen har en indre klasse og interface for å holde styr på viewene inne i hver rad og for å kunne sette lyttere til disse viewene fra fragment-nivået.

Post: Standard POJO som benyttes for å lagre score og tekst for postene.

DBHandler: Hjelpeklasse som fungerer som bindeledd mellom logikk og database. SQLite har blitt benyttet som database da den er lett å bruke og passer for det tiltenkte bruksområdet. Queryene er ikke avanserte, og går mest på standard CRUD og henting av data i en bestemt rekkefølge.

Kilder:

Mye av koden er inspirert av gamle øvinger, forelesninger og for lengst glemte tutorials på nettet. Nedenfor ser du kode jeg har hentet inspirasjon fra på temaer jeg slet med/var ikke kjent med.

Maps:
https://developers.google.com/maps/documentation/android-api/marker

API + JSON:
https://androidkennel.org/android-networking-tutorial-with-asynctask/

Adapter + Recyclerview
https://guides.codepath.com/android/Using-the-RecyclerView#creating-the-recyclerview-adapter
http://stackoverflow.com/questions/30284067/handle-button-click-inside-a-row-in-recyclerview