

IN2000 V26 - Obligatorisk oppgave 2

Innleveringsfrist: mandag 23. februar, kl. 23.59

Sist endret: torsdag 5. februar, kl. 09.10

Innledning

Denne obligatoriske innleveringen skal løses individuelt, det vil si at du skal programmere alle deler av løsningen selv. Dette betyr ikke at vi fraråder samarbeid – tvert imot. Vi oppfordrer deg til å utveksle faglige erfaringer med andre. Det forventes en større egeninnsats enn i førsteårskursene. Dette innebærer at du finner ut av hvordan du løser oppgaven ved å gjøre ukesoppgaver, lete etter og lese relevant dokumentasjon, møte opp i gruppetimer og eksperimentere med enklere apper.

Hensikten med denne oppgaven er å få deg i gang med verktøyene som senere skal benyttes i prosjektarbeidet. Android Studio skal benyttes som IDE, Kotlin som programmeringsspråk og Jetpack Compose til å lage brukergrensesnitt. Det er ingen krav til hvordan appen skal se ut utover det som er spesifisert i oppgaveteksten.

For å få oppgaven godkjent må alle deloppgaver være gjort og appen skal kunne kjøres i en emulator med API-level 34 **uten å krasje**. Under retting vil en Resizable emulator eller tilsvarende med API level 34 benyttes. Hvis du tester på egen fysisk Android-enhet, vennligst spesifiserer Android-versjonen og enhet i kommentarfeltet i [Devilry](#).

Før du begynner

Det er forventet at du har undersøkt og sett på enkle apper som henter data fra API, som for eksempel [Namage](#) og [WizApp](#) (vist i forelesning), eventuelt [Mars Photos](#) fra Codelabs. Vi forutsetter også at du har sett alle forelesninger og har gjort alle [ukesoppgaver](#) til og med uke 3 før du starter.

Hvis du har problemer med å komme i gang anbefales følgende ressurser:

- [Architecture Components](#) og [Get data from the internet](#)

Levering

Trykk på “Build” → “Clean Project” for å fjerne “byggfiler”, deretter skal mappen som inneholder hele Android Studio-prosjektet komprimeres til en zip-fil, og leveres i [Devilry](#) innen fristen. Du **MÅ** sjekke at du kan laste ned filen, pakke ut, åpne, bygge og kjøre prosjektet og at det fungerer som det skal. Dersom oppgaven leveres med en fil som ikke kan bygges, kan du risikere å få oppgaven underkjent uten nytt forsøk.

Du kan lage en zip av hele prosjektet med File -> Export -> Export to Zip File.

Dersom noe mangler eller ikke fungerer så **må** det redegjøres med en begrunnelse for hva som skjer, hva du tror feilen er, og hva du har forsøkt i en README. Hvis dette ikke er med vil du risikere at oppgaven blir underkjent uten nytt forsøk.

Casebeskrivelse

Alpakkaland har nylig avholdt valg om hvilket parti som skal få styre de neste fire årene. I den anledning skal du lage en app som viser valgresultatet, samt info om de ulike partiene. I Alpakkaland stiller totalt fire partier til valg, og det er tre valgdistrikter som hver har sin egen opptelling av stemmer.

Oppgaven

Oppgaven består av følgende tre deler:

- Del 1: Informasjon om partiene
- Del 2: Visning av det totale valgresultatet
- Del 3: Utvide med mer detaljert visning om partiene + valgresultater per distrikt

I slutten av hver del er det vedlagt en skisse av hvordan de forskjellige delene kan henge sammen.

Formelle krav:

Disse kravene gjelder for alle deloppgaver.

- Benytt innebygde funksjoner på Collections, så langt det lar seg gjøre (som *forEach*, *map*, *filter*).
- Appens arkitektur skal basere seg på [Androids anbefalte app-arkitektur](#)
- Den skal benytte seg av “best practice” [gjennomgått på forelesning](#) og [de sterkt anbefalte praksisene for Android-utvikling](#). I all hovedsak er det kravene innenfor “layered architecture”, “UI Layer” og “ViewModel” som gjelder (Dependency-injection er ikke et krav i denne oppgaven).
- Appens layout bør støtte ulike skjermstørrelser. Det betyr at den skal se lik ut uavhengig av skjermstørrelse og oppløsning.
- Alle filer bør ha minimalt med Warnings. Eventuelle warnings skal redegjøres for i README.md.
- Dersom du benytter deg av andre biblioteker enn de som er nevnt under [relevante biblioteker](#), skal disse spesifiseres i README.md som er vedlagt prosjektmappen samt med hvordan de fungerer i besvarelsen.

Relevante ressurser og biblioteker

- [Material design 3](#)
- [Guide to app architecture](#)
- Bilder fra nettet: [Coil](#)
- [Jetpack ViewModel](#)
- For henting av data anbefaler vi Ktor, men du står fritt til å benytte deg av hvilken http-klient du ønsker:
 - <https://ktor.io/docs/client-dependencies.html>
 - <https://ktor.io/docs/http-client-engines.html#jvm-android>
 - <https://ktor.io/docs/request.html>
 - <https://ktor.io/docs/serialization-client.html>
- [Jetpack compose navigasjon](#)

Det ligger også relevante ressurser spesifikke til hver del under hver deloppgave.

Komme i gang

Sørg for at du har nyeste versjonen av Android Studio (Ladybug) før du starter.

1. Opprett et nytt Android Studio-prosjekt, velg “Empty Activity”
2. “Name” settes til **[UiO-brukernavn]_oblig2**
3. “Package name” settes til **no.uio.ifi.in2000.[ditt uio-brukernavn].oblig2**
4. Velg API 24 som minimum SDK
5. Sørg for at det er “Kotlin DSL (build.gradle.kts) [Recommended]” som er valgt som språk for bygg-konfigurasjonen.

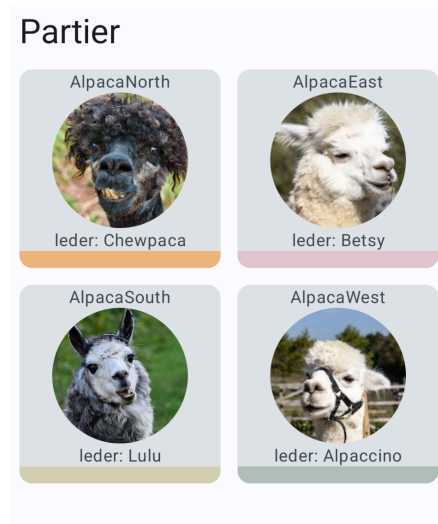
Del 1: Alpakkapartiene

Info om hvert alpaka-parti skal vises i et kort *AlpacaCard*.

Dette kortet skal inneholde:

- Partiets navn
- Lederens navn
- Lederens bilde fra en url
- Partiets farge

Hvordan du velger at kortet ser ut er opp til deg. Men vi legger ved et forslag her.

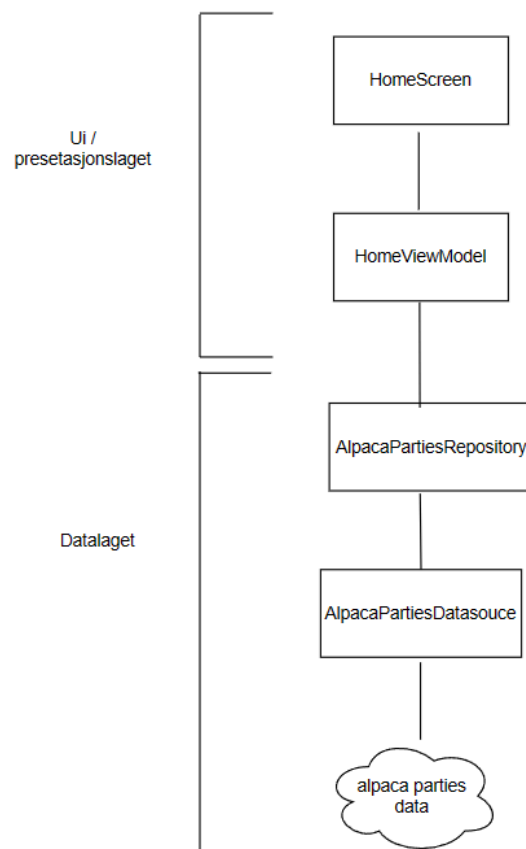


Informasjonen om hvert parti skal hentes fra dette endepunktet:

→ <https://in2000-proxy.ifi.uio.no/alpacaapi/v2/alpacaparties>

For å hente og vise data MÅ du følge den anbefalte arkitekturen til Android. Dette innebærer at du skal opprette:

- En dataklasse *PartyInfo* som inneholder de samme feltene som i endepunktet, altså partiets navn, lederens navn, url til bilde av lederen, partiets farge og en beskrivelse av lederen til partiet som skal brukes i neste del.
- En datakilde *AlpacaPartiesDataSource* som henter og deserialiserer responsen fra endepunktet over og returnerer en liste av *PartyInfo*-objekter.
- Et Repository *AlpacaPartiesRepository* som får data fra *AlpacaPartiesDataSource* og som tilbyr en metode ViewModel-en kan bruke for å få informasjon om partiene. Denne dataen skal være på formatet *List<PartyInfo>*.



- En ViewModel-klasse *HomeViewModel* som holder styr på UI-state. Denne skal bruke *viewModelScope* for å hente data asynkront fra *AlpacaPartiesRepository* og oppdatere UI-staten.
- En skjerm *HomeScreen* som viser kortene i en *LazyColumn* eller *LazyVerticalGrid*. Denne skal **kun observere** UI-staten som ligger i *HomeViewModel*.

Noen filer og pakker berørte i denne delen (det kan hende du vil lage flere):

- ui
 - home
 - HomeScreen.kt
 - HomeScreenViewModel.kt
- model
 - alpacas
 - PartyInfo.kt
- data
 - alpacas
 - AlpacaPartiesDataSource.kt
 - AlpacaPartiesRepository.kt

Nyttige ressurser for denne delen:

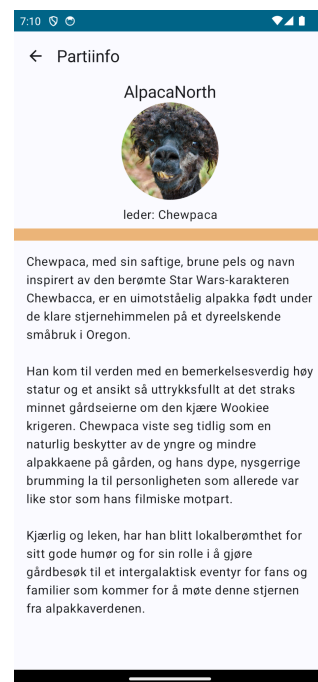
- [Coil](#)
- <https://developer.android.com/jetpack/compose/libraries#viewmodel>
- <https://developer.android.com/topic/libraries/architecture/coroutines#viewmodels>
- [Ktor](#)

Del 2: Skjerm med detaljer

Når du trykker på hvert parti på hjemskjermen blir du sendt til en infoskjerm for hvert parti *PartyScreen*. Denne siden inneholder mer informasjon om partiet som ble trykket på og skal inneholde:

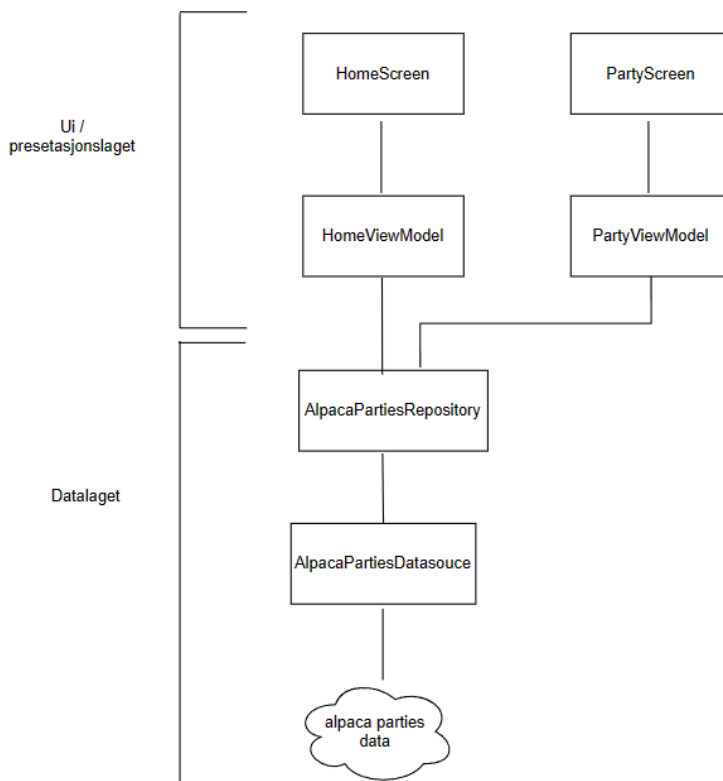
- partinavn
- bilde
- farge
- en beskrivende tekst for partilederen.

Når en navigerer til den andre skjermen skal *partyID* sendes som et argument. Pass på at *viewModel*-en blir initialisert kun én gang når du navigerer til den. Dataen om et spesifikt parti kan du hente fra *AlpacaPartiesRepository* som i forrige del. Start med å teste skjermen uten navigasjon med en hardkodet *partyID*.



I denne delen må du utvide appen med:

- En ny metode i *AlpacaPartiesRepository* som gir deg data for hvert enkelt parti.
- En ny skjerm *PartyScreen*, som ligger i *PartyScreen.kt*, som viser info om et spesifikt parti.
- En ny *ViewModel* *PartyViewModel*. Denne skal også bruke *viewModelScope* for å hente data fra *AlpacaPartiesRepository* for det valgte partiet og oppdatere UI-ets tilstand.



- Navigering til *PartyScreen* der du sender med partyID som argument.
- En mulighet for å navigere tilbake til hjemskjermen ved bruk av en *TopAppBar*.

Noen filer og pakker berørte i denne delen

- MainActivity.kt
- ui
 - party
 - PartyScreen.kt
 - PartyViewModel.kt
 - home
 - HomeScreen.kt
- data
 - alpacas
 - AlpacaPartiesRepository.kt

Nyttige ressurser:

- <https://developer.android.com/jetpack/compose/navigation#nav-with-args>
- <https://developer.android.com/jetpack/compose/components/app-bars>
- <https://developer.android.com/topic/architecture/ui-layer/state-production#initializing-state-production>

Del 3: Valgresultatet

Det har blitt avholdt valg i alle de 3 distriktene i Alpakkaland. Det er stemmer for hvert parti i hvert distrikt sine stemmer. Distrikt 1 og 2 tilbyr stemmer på et format med individuelle stemmer, mens distrikt 3 tilbyr stemmer på et aggregert format. I denne delen skal det lages funksjonalitet for å hente, telle og vise disse stemmene på hjemskjermen.

3.1 Vise frem stemmer

I denne delen skal du lage en komponent *VoteList* for å vise frem stemmer. Denne skal **kun vise frem** en liste av partinavn og tilhørende stemmer. Lag en preview for denne komponenten og noe dummy-data i preview-en.

Parti	Antall stemmer
AlpacaNorth	7569
AlpacaWest	7796
AlpacaEast	9554
AlpacaSouth	7953

Eksempel på komponent som viser stemmer

Filer og pakker berørt her:

- ui
 - home
 - HomeScreen.kt
 - VoteList.kt

3.2 Hente stemmer

Du skal hente stemmer fra flere endepunkter. Vi anbefaler at du setter opp flyten med kun ett endepunkt for hele delen først for å få “flyten” til å fungere, deretter utvide med de resterende. Alle datakildene skal returnere data på samme format selv om endepunktene har forskjellige formater. Dette er formatet *DistrictVotes* som er definert som:

```
data class DistrictVotes(  
    val district: District,  
    val alpacaPartyId: String,  
    val numberOfVotesForParty: Int  
)
```

Der *District* er en enum class for de 3 distriktene.

Endepunkter:

- Distrikt 1: <https://in2000-proxy.ifi.uio.no/alpacaapi/v2/district1>
- Distrikt 2: <https://in2000-proxy.ifi.uio.no/alpacaapi/v2/district2>
- Distrikt 3: <https://in2000-proxy.ifi.uio.no/alpacaapi/v2/district3>

OBS: Sjekk API-et og responsen nøye ved hjelp av f.eks. [Bruno](#), [Postman](#) eller Firefox og sjekk formatet og undersøk forskjellene.

Du skal her opprette:

- En datakilde *IndividualVotesDataSource* som henter og transformerer stemmer fra endepunktene til distrikt 1 og distrikt 2 og returner *DistrictVotes*-objekter fra hvert distrikt.
- En datakilde *AggregatedVotesDataSource* som henter og transformer og returner *DistrictVotes*-objekter fra distrikt 3.

- Et repository *VotesRepository* som henter stemmedata fra *IndividualVotesDataSource* og *AggregatedVotesDataSource* og tilbyr et grensesnitt for å hente stemmer for hvert parti i ett distrikt.
- Utvide *AlpacaPartiesRepository* med funksjonalitet for å kombinere stemmer og partinavn. Stemmer skal hentes via grensesnittet til *VotesRepository*.

Pakker og filer berørte (det kan hende du vil lage flere):

- data
 - votes
 - *VotesRepository.kt*
 - *IndividualVotesDataSource.kt* (distrikt 1 og 2)
 - *AggregatedVotesDataSource.kt* (distrikt 3)
 - alpacas
 - *AlpacaPartyRepository.kt*
- model
 - votes
 - *DistrictVotes.kt*
 - *District.kt* (Enum class)

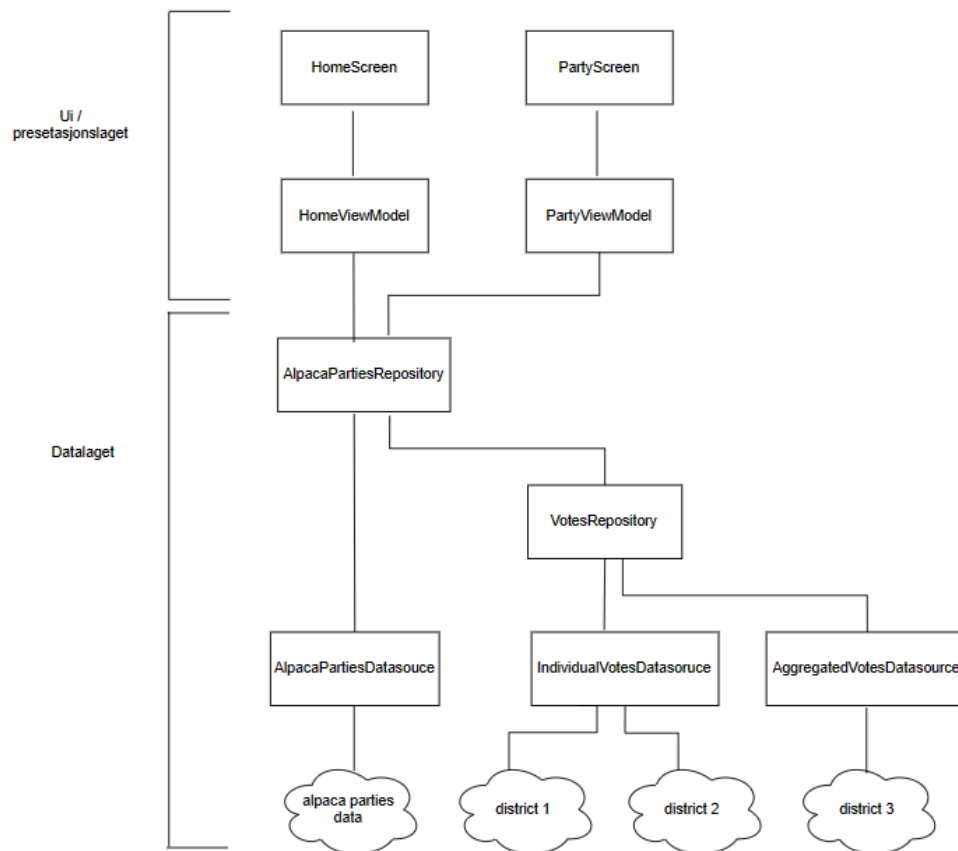
3.3 Utvide HomeScreen

Den komponenten du lagde i 3.1. skal nå brukes for å vise frem de stemmene fra hvert distrikt.

Du skal her:

- Legge til komponenten som viser frem stemmer på hjemskjermen.
- Lage en dropdown-meny der du kan velge mellom de forskjellige distriktene som ved valg av et annet distrikt gir en beskjed til *viewModel*-en om at et annet distrikt er valgt.

- Utvide HomeViewModel sin UI state med informasjon om valgt distrikt og stemmene til det valgte distriktet. Pass på at hjemskjermens tilstand er bevart etter navigasjon til en annen skjerm som betyr at hvis du velger distrikt 2, så vises fortsatt det når du går tilbake fra detaljvisningskjermen.
- **Pakker og filer berørte:**
- ui
 - home
 - HomeScreen.kt
 - HomeViewModel.kt



Foreslått skisse over strukturen i appen til slutt

Ekstraoppgaver

OBS: Disse oppgavene er helt frivillige. Hvis du vil implementere dem, lever først en versjon uten ekstra funksjonalitet som fungerer 100%.

- I prosjektoppgaven senere i kurset skal det i appen vises feilmeldinger hvis internett mangler eller om det ikke fungerer å hente data fra API. Implementer funksjonalitet for dette og vis en beskjed i form av en [snackbar](#)
- Det skal være mulig å se hvor mange stemmer det er kommet totalt for alle distriktene per parti.
- Dataen fra API skal caches, slik at den samme dataen ikke blir hentet flere ganger. Finn ut hvordan dette kan caches lokalt.
- Stemmeoversikten kan visualiseres med en graf eller et diagram i stedet for en vanlig tabell. Her kan du bruke et bibliotek hvis du vil.