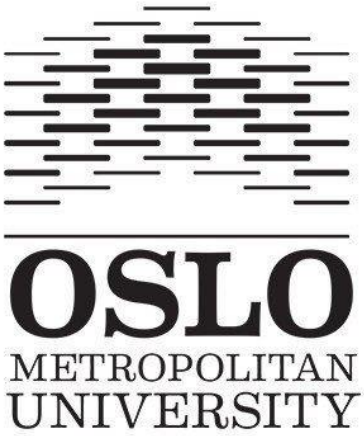


OBLIG 2

ROBERT LARSEN

APP UTVIKLINGd

S349967

30.10.2022

Innhold

OBLIG 2 1
Innledning 3
Innledningsvis om design valg 3
Kontakt register 4
Avtale register 6
Notifikasjons av avtaler 8
Content manager 9

Innledning

Spillet følger en følgende kravspesifikasjon gitt i oblig2, apputvikling, vår 2022:

- Legge til nye kontakte
- Sette kontakter
- Endre kontakt informasjon
- Legge til nye avtaler
- Slette avtaler
- Endre avtaler
- På et tidspunkt hver dag sender en alarm som begynner å sende ut sms til alle kontakter en har avtale med. Samtdig sendes det en intern notifikasjon inne i appen, og en notifikasjon om utenfor, om at det snart er en avtale.
- En kan ta av og på servicen som gjør at en får sendt sms og notifikasjoner
- Tidspunktet når notifikasjoner og sms blir sendt, er satt i sharedprefrence.
- Hvis ingen beskjed er gitt i en avtale, brukes default beskjed gitt i shared prefrence.
- Frivelig å ha med Content provider. Brukes til å dele informasjon med andre applikasjoner på mobiltelefonen. Er antatt det skulle gjøres på både avtale og kontakt, og at det skulle gjøres for slett/update/insert.

Oppgaven går ut på å lage en applikasjon som gir brukere mulighet til å organisere avtaler med kontakter. Det vil si at en kan opprette, slette og oppdatere avtaler og kontakter, samt at denne informasjonen deles videre til kontaktene. Måten dette skulle gjøres på, var å gi alle kontakter på den dagen en har avtale, en sms. Videre skulle det gis notifikasjon til brukeren, om at det er avtaler i dag. Utover dette skulle det lages contentprovider, slik at tredjeparts programvare får tilgang til disse dataene – som eventuelt kan dele informasjonen på andre måter.

Mye av koden og designet har tatt utgangspunkt i foreleser i apputvikling 2022 vår, sine forelesnings notater.

Kodemessig, så virker ting å funke – men oppdatering og sletting i content provider ser ikke til å funke

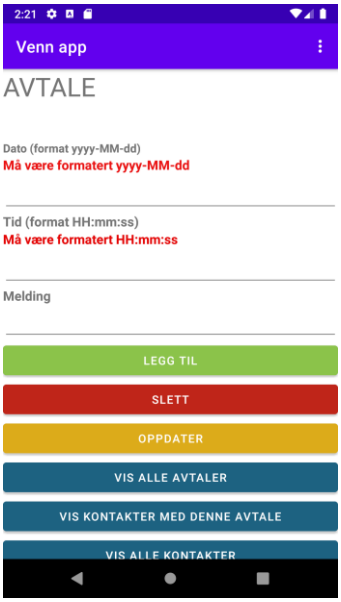
Innledningsvis om design valg

Mye av koden tar utgangspunkt i udelt kode av foreleser i dette faget. Det var også antatt at en skulle ta utgangspunkt i designet gitt av foreleser.

Mye av designet er lik figur1. Designet kan rettferdiggjøres med gesalts prinsipp om proximity[4]. Det som er i nærheten av hverandre, har en slags tilhørlighet. Det som er langt vekke oppfattes som noe som ikke har tilhørlighet. Ved at knapper som kontrollerer skjemaet er i nærheten av dette skjemaet, og at listen som blir påvirket av skjemaet og knappene er i nærheten av nettopp skjemaet og knappene, er å designe i henhold til proximity gesalts prinsipp.

Når en jobber med ulike oppgaver, så kan ulike typer informasjon være relevant. Dermed kan en bytte ut irrelevant informasjon, f.eks lister over informasjon, med noe som er med relevant. Dette er en lur måte å gjøre det på, ettersom en da bevarer gesalts prinsipp om proximity.

Figur 1

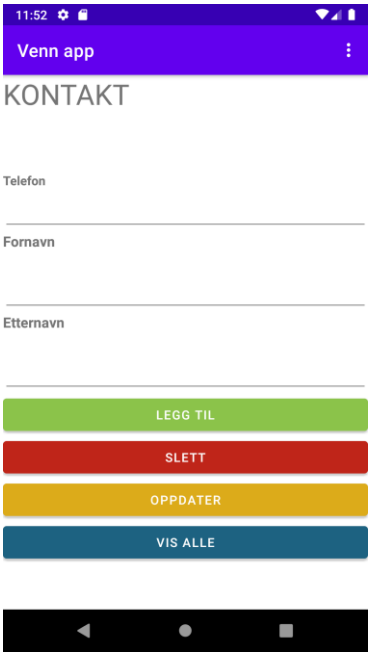


Kontakt register

Å holde rede på informasjon er alltid lurt, spesielt med tanke på informasjon som er vanskelig å huske. Det å lagre informasjon på en plass, fremfor å måtte huske, er til hjelp for den kognitive prosessen når informasjonen skal brukes til å utføre. Alternativt måtte en har skrevet inn informasjonen hver gang systemet trenger informasjonen.

I faget apputvikling vår 2022 (dvs da denne obligen ble skrevet) fikk vi utdelt design. Det er dette designet (se figur1) som er utgangspunktet for kjernen i designet.

Figur 2



Med tanke på inputvalidering, så er det input filtrering med tall pga input typen er av typen phone. For fornavn og etternavn, er det ingen input validering.

Når en legger til en kontakt, vil den nye kontakten legges i liste - som vises når en trykker knappen vis alle.

Designet i figur 2, tar utgangspunkt i en kombinasjon av kjennskap til bootstrap sin cardview [2] og at android også tilbyr cardviews[1]. Cards går ut på at en samler informasjon i en firkant, som gjerne er delt i hode-del og kropp-del. Hode-del fungerer da som en slags oversikt over det som er i kroppen, f.eks ved hjelp av et bilde og diverse informasjon som kan være viktig for å få overblikk over inneholdet. Cardview ser ut til å bruke gesalts lov om proximity [4], hvor en skaper visuell avstand mellom informasjon med å sepaerer dem i cards – som er atskilt fra hverandre.

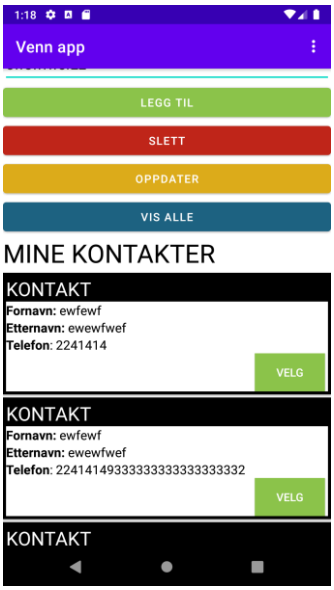
I disse cards, er det bare brukt tittel. I selve kroppen er det brukt en visning over kolonne-verdi par assosiert med entries i tabellen kontakt. Kolonne navn skiller seg fra verdi, ved at det er brukt fet skrift.

Ved å trykke velg, så vil primær nøkellen verdien for denne rad bli plassert i et hidden input felt (ikke synlig for brukeren).

For å hjelpe brukeren å forstå hva som endres, så blir de gamle verdiene ført over i input feltene. En annen fordel med å gjøre dette er at hvis brukeren uheldigvis trykker oppdater etterpå, så vil en endre til nøyaktig det samme – en slipper risiko for å endre til tomme strenger, uten at det skjer en endring i input felt.

På samme måte som en nå kan endre, pga primærnøkkel verdi er overført til hidden input felt, så kan en også slette. Når en har slettet, så vil dette fjernes fra listen over kontakter.

Figur 3



Avtale register

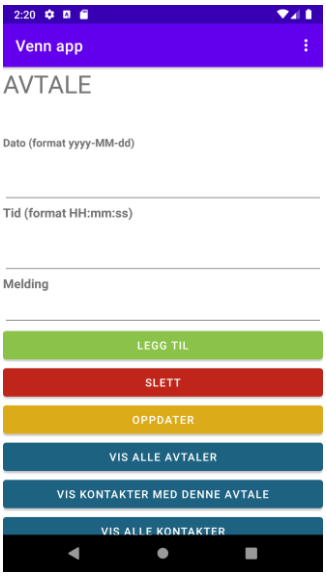
Avtale registerets design tar utgangspunkt i designet gitt av foreleser i faget apputvikling vår 2022. Dvs en side med registreringsform, etterfulgt av masse knapper som kontrollerer input feltene.

I avtale, så kan en registrere dato og tid for en avtale. I tillegg til dette kan det lagres en melding, men en kan godt la feltet stå tom. Dette er meldingen som blir sendt til alle kontakter tilknyttet denne avtalen, via sms. Hvis feltet er tomt, så vil det bli sendt en default beskjed gitt i sharedprefrence.

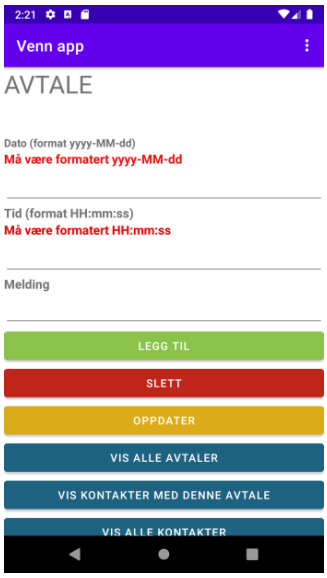
Her er det input-validering for tid og dato, tilsvarende string formatet av java.sql.Date og java.sql.Time. På denne måten kan en sammenlikne strenger, ved at en prøver å forholde seg til kun et format av tid og dato.

Når en legger til en ny avtale, så vil denne kune vises når en trykker vis alle avtaler.

Figur 4



Figur 5

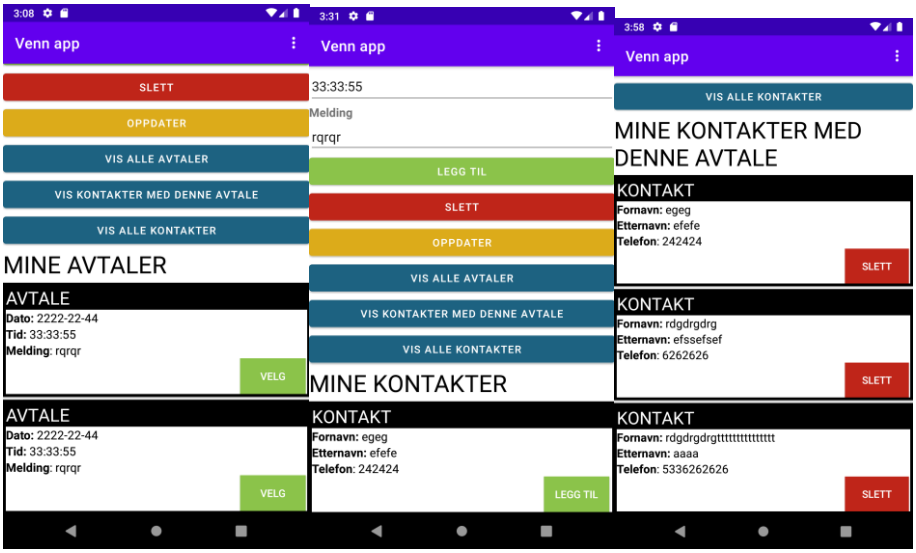


Akkurat som i kontakt registeret, som blir det her også brukt cardview (se figur 5), med de samme argumenter om at en det er en brukt metode for å organisere informasjon. Når en trykker velg, så vil også her primærnøkkel verdien for denne avtalen (raden) bli satt i hidden input felt.

For å legge til en kontakt til denne avtalen, trykker en på vis kontakter. Da vil en vise en liste med kontakter der en før så liste over avtaler – sammen med aktivitet avtaleActivity. Her kan en legge til kontakter. Når en har lagt til en kontakt, vil den vises i listen som dukker opp når en trykker vis kontakt med denne avtale (se figur 7). For å fjerne kontakt fra avtale, trykker en knappen slett.

c

Figur 6 **Figur 7** **Figur 8**



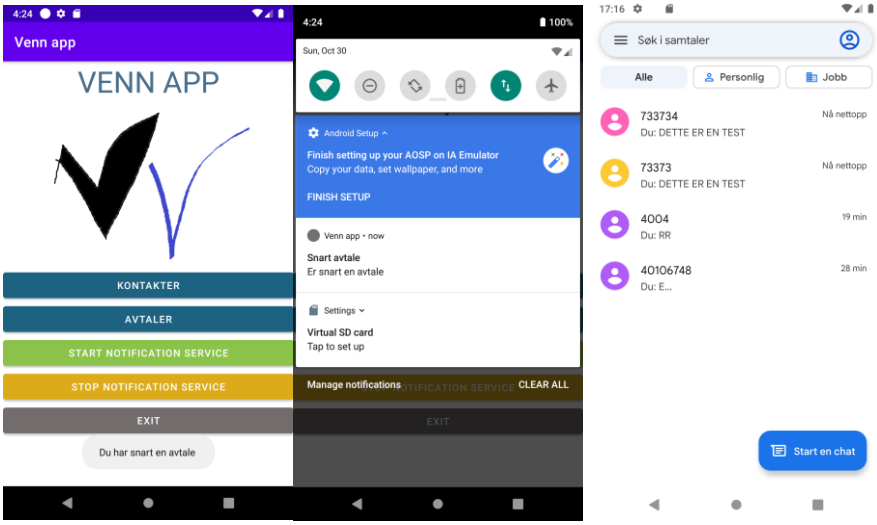
Notifikasjons av avtaler

På et et bestemt tidspunkt, hver dag, basert på et tidspunkt lagret i shared service, så sendes sms til alle kontakter som en har en avtale med denne dagen. I sms, vil de da dukke opp noe som likner figur 10. Videre vil det sendes 2 notifikasjoner, en gjennom en notifikasjons kanal slik at noe som likner figur 9 med «snart avtale» dukker opp. I selve applikasjonen vil noe som likner figur 8 dukke opp,dvs det gråe med tekst i.

Det er notifikasjons servicen som står for å sende ut 3 ulike meldinger, en gjennom Toast.maketex, en gjennom å sende .send() kommando via pendingIntent til SMSActivity, og en .notify() fra notificiation manager. Notifikasjon service gjør alt dette, nøyaktig 1 gang, når den blir aktivert av periodisk service 1 gang i døgnet på predefinert tidspunkt (satt i shared prefrence).

Den periodiske servicen kan aktiveres og deaktiveres, gjennom to knapper i hovedmenyen (se figur 8).

Figur 9 **Figur 10** **Figur 11**



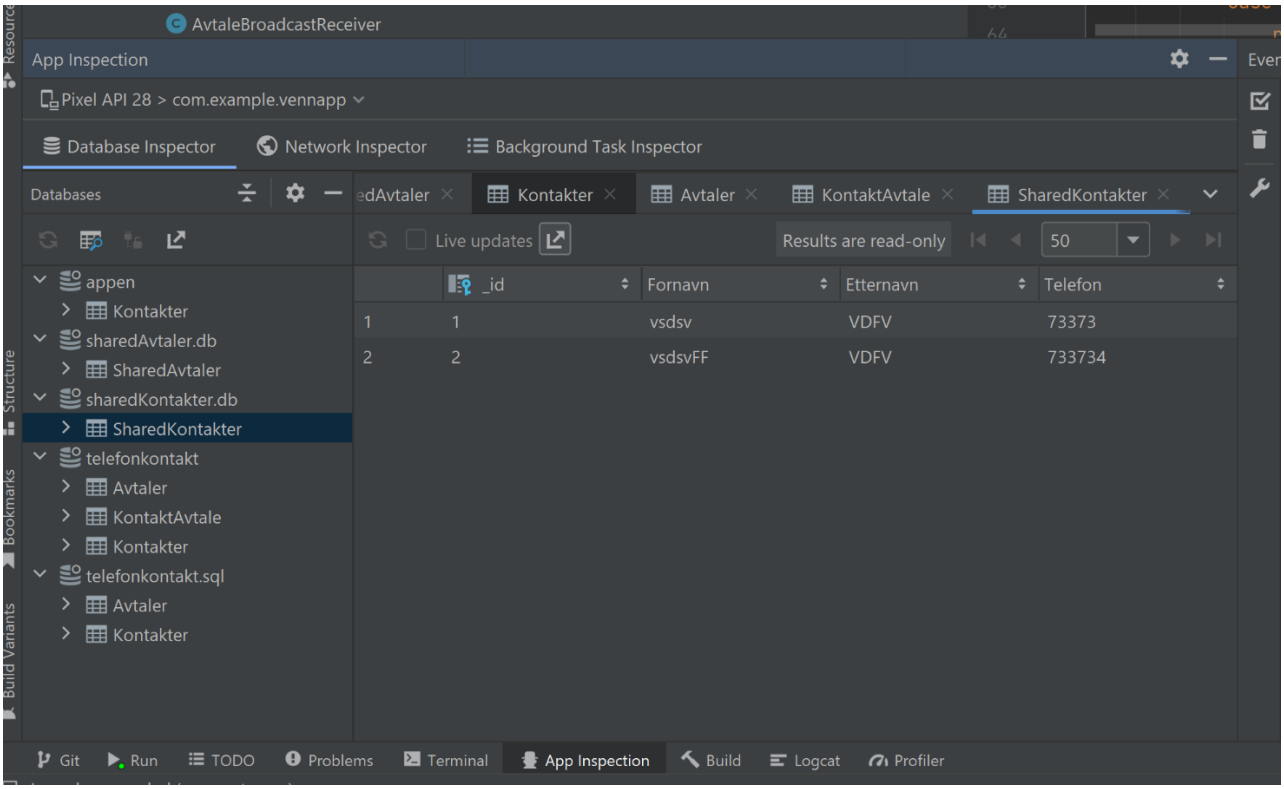
Content manager

Content manager er en mekanismer som muliggjør å dele data med andre applikasjoner, som muliggjør at tredjepart applikasjoner kan ta i bruk denne informasjonen via database spørringer i form av content:// [navn på provider]/[Entiet, som avtale og kontakt].

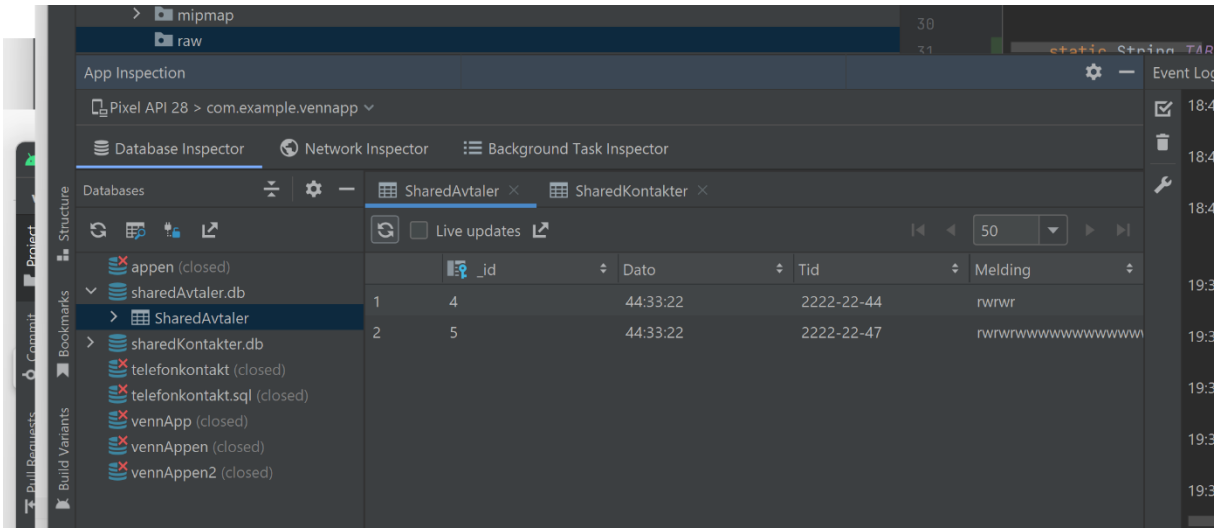
Figur 12 og 13 viser at content deling av avtale og kontakt er i to ulike databaser. Den lokale databasen med tabellene avtale og kontakt er i databasen telefonkontakt.

På grunn av at det kan være mismatch mellom primær nøkkel i content deling, og lokal databse, så legger content manager inn neste rad med maks id fra den lokale databasen. Å legge inn funker, men å oppdatere og slette funker ikke helt – men er ikke pga mismatch mellom hva id i hva som blir lagt til som er årsaken.

Figur 12


AvtaleBroadcastReceiver
App Inspection
Pixel API 28 > com.example.vennapp
Database Inspector
Network Inspector
Background Task Inspector
Databases
Kontakter
Avtaler
KontaktAvtale
SharedKontakter
Live updates
Results are read-only
50
_id
Fornavn
Etternavn
Telefon
1 1 vsdsv VDFV 73373
2 2 vsdsvFF VDFV 733734
appen
Kontakter
sharedAvtaler.db
SharedAvtaler
sharedKontakter.db
SharedKontakter
telefonkontakt
Avtaler
KontaktAvtale
Kontakter
telefonkontakt.sql
Avtaler
Kontakter
Git
Run
TODO
Problems
Terminal
App Inspection
Build
Logcat
Profiler

Figur 13



Referanser

1. https://developer.android.com/develop/ui/views/layout/cardview, lest 30.10.2022
2. https://getbootstrap.com/docs/4.0/components/card/, lest 30.10.2022
3. Forelesningsnotater apputvikling 2022 vår oslomet, lest 30.10.2022
4. Yanxia Liang, **https://iopscience.iop.org/article/10.1088/1757-899X/392/6/062054/pdf,s.2**

Appendix

SQL script for å lage databasen

.schema

CREATE TABLE android_metadata (Locale TEXT);

CREATE TABLE Kontakter(_ID INTEGER PRIMARY KEY, Fornavn TEXT, Etternavn TEXT, Telefon TEXT);

CREATE TABLE Avtaler(_ID INTEGER PRIMARY KEY, Dato TEXT, Tid TEXT,Melding TEXT);

CREATE TABLE KontaktAvtale (

```
        kontaktId       INTEGER,
        avtaleId        INTEGER,
        FOREIGN KEY(kontaktId) REFERENCES Kontakter(_ID),
        FOREIGN KEY(avtaleId) REFERENCES Avtaler(_ID),
        PRIMARY KEY(kontaktId,avtaleId)
);
```

Link til kode: https://github.com/s349967/vennApp