

# **ITE 1621 - Applikasjoner for web og mobil**

**Karaktergivende oppgave - rapport om programmet  
TracknHide**

**501629 - Knut Lucas Andersen**



**2013**

# Innholdsfortegnelse

---

Innholdsfortegnelse .....	2
Innledning.....	3
Hva er Google Cloud Messaging (GCM)?.....	4
Så hva kan en da gjøre med GCM?.....	4
Hva består GCM av? .....	5
Android - applikasjonen .....	7
MainActivity og Google Maps.....	7
MapOverlay .....	9
InnstillingsActivity, SystemloggActivity og VisRuterActivity .....	11
SQLite, filer og lagring av posisjonsdata/ruter .....	12
GCMIntentService og tilkobling til via ServerTilkobling .....	15
Server .....	18
Forelderklassen og arvede funksjoner .....	18
Mottak og sending av melding .....	19
Nettsidene og hva administrator får se.....	21
MySQL og databasens innhold .....	22
Oppsummering/konklusjon.....	23
Kilder og litteraturliste .....	24
Android .....	24
Exceptions/Errors .....	24
Google Cloud Messaging (GCM).....	24
Google Maps.....	25
Server .....	25
Litteraturliste.....	25
Vedlegg 1: AndroidManifest.xml.....	26
Vedlegg 2: Android - Klassen Enhet.....	28
Vedlegg 3: Server - ForelderServlet.....	30
Vedlegg 4: Server - SendMeldinger .....	31
Vedlegg 5: MySQL og SQLite database.....	36
MySQL databasen på server.....	36
SQLite databasen på Android (klient) .....	36

# Innledning

---

Navn: Knut Lucas Andersen

Studentnr: 501629

E-post: [darkphoenix@live.no](mailto:darkphoenix@live.no)

Dette er rapporten for utviklingsprosjekt, en karaktergivende oppgave i faget ITE1621 - Applikasjoner for web og mobil. Hovedtema i faget har vært programmeringsutvikling for Android ved å bruke programmeringsspråket Java. Android er et operativsystem som brukes på mobiler og tablets, men som og har begynt å bli tatt i bruk på tv-er. Operativsystemet som utviklingen er siktet mot er mellom API lvl8 (OS 2.0) til API lvl15 (OS 4.0.3), og navnet jeg ga min applikasjon er TracknHide.

Oppgaven baserte seg på å lage en "sporingapplikasjon" hvor brukernes posisjon ble sendt til en server, som da via GCM<sup>1</sup> videresendte brukerens posisjon til andre enheter som var tilkoblet serveren. Måten posisjonen skulle fremvises til brukerne var ved å implementere Google Maps API lvl1<sup>2</sup>, hvor brukernes rute ble tegnet på kartet. Videre forslag var muligheter som å kunne skifte farger på ruter vist på kartet, melde seg av og på sporing (*sending av posisjonsdata*), angi zoom-nivå på kartet og lagring av posisjonsdata for senere å kunne vise rutene på kartet. Men grunnleggende funksjonalitet var at brukere skulle kunne se hverandre på kartet.

En annen del av oppgaven var utvikling av en server. Serveren skulle håndtere mottakelse og videresending av posisjonsdata til tilkoblede/påloggede brukere. Tilkoblede skulle da lagres temporært i en database (MySQL) inntil klient avregistreres/avslutter tilkobling til server. Et krav til server var at det skulle brukes Servlets og at posisjonsdata ikke skulle lagres på server, men at administratorer skulle kunne se brukers siste posisjon.

Litt om hvordan rapporten er bygd opp. Rapporten starter med denne innledningen, etterfulgt av et kapittel hvor jeg forteller litt om Google Cloud Messaging (GCM) og hva det er. Rapporten fortsetter deretter med utredning om selve programutviklingen. Utredningen starter med Android delen, hvor jeg forteller om hva jeg har gjort, erfaringer jeg har tatt med meg og problemer som oppsto. Jeg fortsetter videre med utviklingen av Server og koblingen Server-Android. Rapporten avsluttes med en oppsummering, etterfulgt av referanser til kilder og vedlegg.

Vedleggene som følger med er:

- AndroidManifest.xml
- Enhet (*kildekode for enhetsklassen i Android-applikasjonen*)
- ForelderServlet (*kildekode servlet*)
- SendMeldinger (*kildekode servlet*)
- MySQL og SQLite - tabelloversikt

## NB!

Merk at deler av kildekode vist i vedlegg er fjernet for plassreduksjon.

---

<sup>1</sup> Google Cloud Messaging

<sup>2</sup> Denne er nå deprecated : <https://developers.google.com/maps/documentation/android/v1/mapkey>

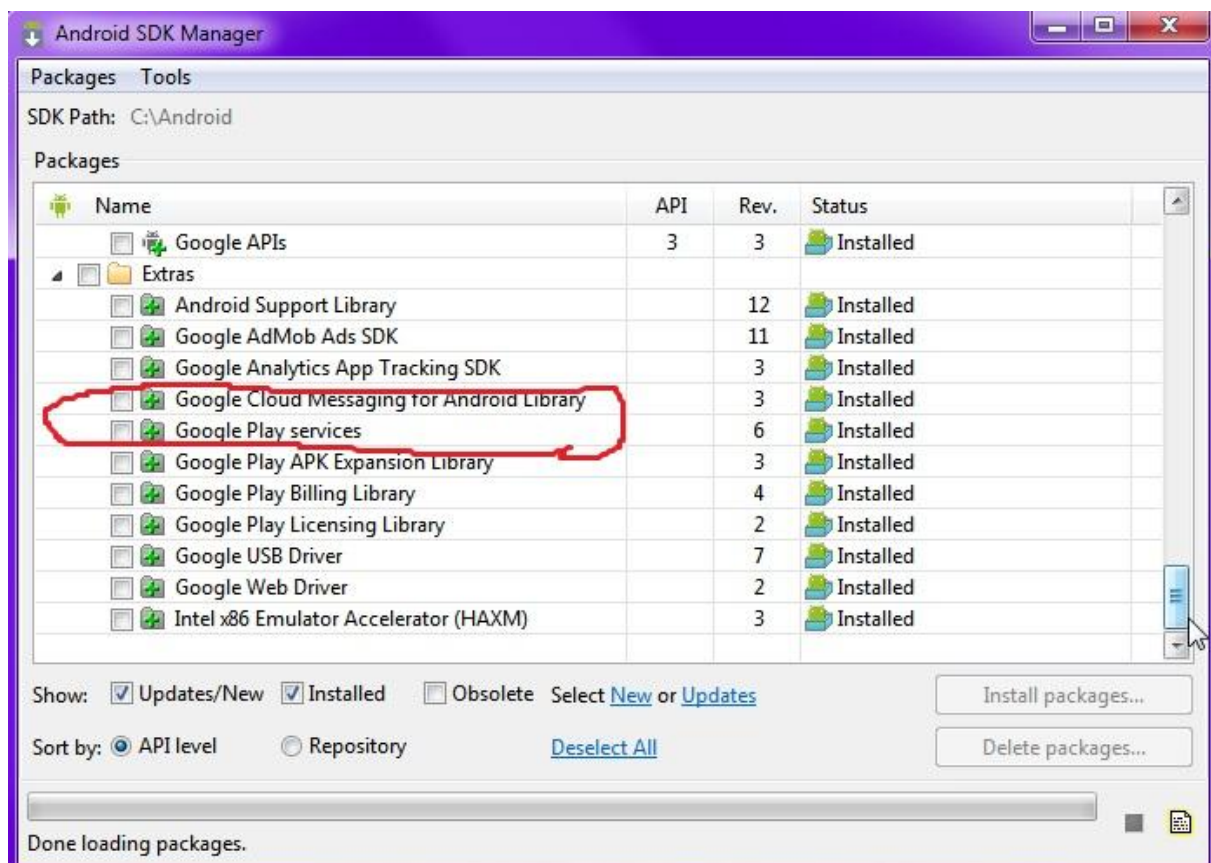
# Hva er Google Cloud Messaging (GCM)?

Google Cloud Messaging (GCM) er en service som brukes for Android. GCM gjør at du kan sende data fra egen server til tilknyttede brukers Android enheter. Dette kan være alt fra en kort beskjed til en stor melding på opptil 4kb med data. GCM tar for seg alle aspektene av opplisting av meldinger og levering til Android enhetene.<sup>3</sup>

## Så hva kan en da gjøre med GCM?

En kan la egen (eller andres) server sende beskjeder til brukere med egenutviklet Android applikasjon. Applikasjonen trenger faktisk ikke å være aktiv for å motta nye meldinger. Dette pga GCMIntentService<sup>4</sup> har mulighet for å "vekke opp" systemet så lenge BroadcastReceiver og tillatelser er satt opp i AndroidManifest

GCM har ikke et grensesnitt, foruten håndtering av meldinger som inneholder data. GCM sender bare dataene rett til Android applikasjonen, som da får full kontroll over disse. Hva en velger å gjøre med data som mottas er da opp til utvikler å bestemme. Husk å aktivere Google Cloud Messaging under API Console<sup>5</sup> ---> Services og i Eclipse installere bibliotekene markert i rødt:



<sup>3</sup> Direkte oversatt fra: <http://developer.android.com/google/gcm/index.html>

<sup>4</sup> Servicen som behandler meldinger og registrering av/på server

<sup>5</sup> <https://code.google.com/apis/console/>

## Hva består GCM av?

### Mobil enhet:

Kravet for å kjøre GCM er Android OS 2.2 eller høyere, og Google Play Store må være installert. For kjøring i emulator kreves Android OS 2.2 med Google API. Siden GCM bruker en eksisterende tilkobling til Google Services, kreves det en Google konto for Android mobiler som har OS > 3.0 Google konto kreves ikke for mobiler som kjører Android OS 4.0.4 eller høyere.

### 3rd-party applikasjonsserver:

En applikasjonsserver satt opp med formål å bl.a. implementere GCM. Data sendes fra server til mobil enhet via GCM. Måten enhetene identifiseres på er via deres registrationID.

### GCM-server:

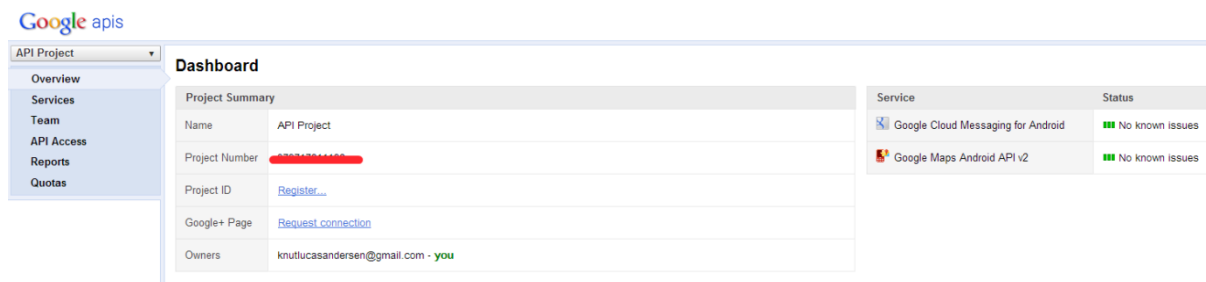
Google server som tar for seg den faktiske sendingen fra applikasjonsserver til mobil enhet.

### Google brukerkonto:

Kreves hvis enhet har Android OS lavere enn versjon 4.0.4.

### SenderId:

Prosjektnummer man får fra API konsollen<sup>5</sup>. Brukes for å identifisere en Android applikasjon som har tillatelse til å motta beskjeder fra GCM.



### RegistreringsID:

En identifikasjon som gis ut av GCM. RegistreringsID tilsier at denne applikasjonen/enheten har tillatelse til å motta meldinger fra GCM. Når en mobil enhet har fått en registreringsID, så sendes denne også til applikasjonsserveren, slik at denne blir registrert sammen med andre tilkoblede enheter. RegistreringsID er unik for hver enhet, men ikke for alltid. Dette siden registreringsID gjelder kun så lenge enhet er tilkoblet GCM (NB! Dette er min oppfattelse av hva som står skrevet)<sup>6</sup>.

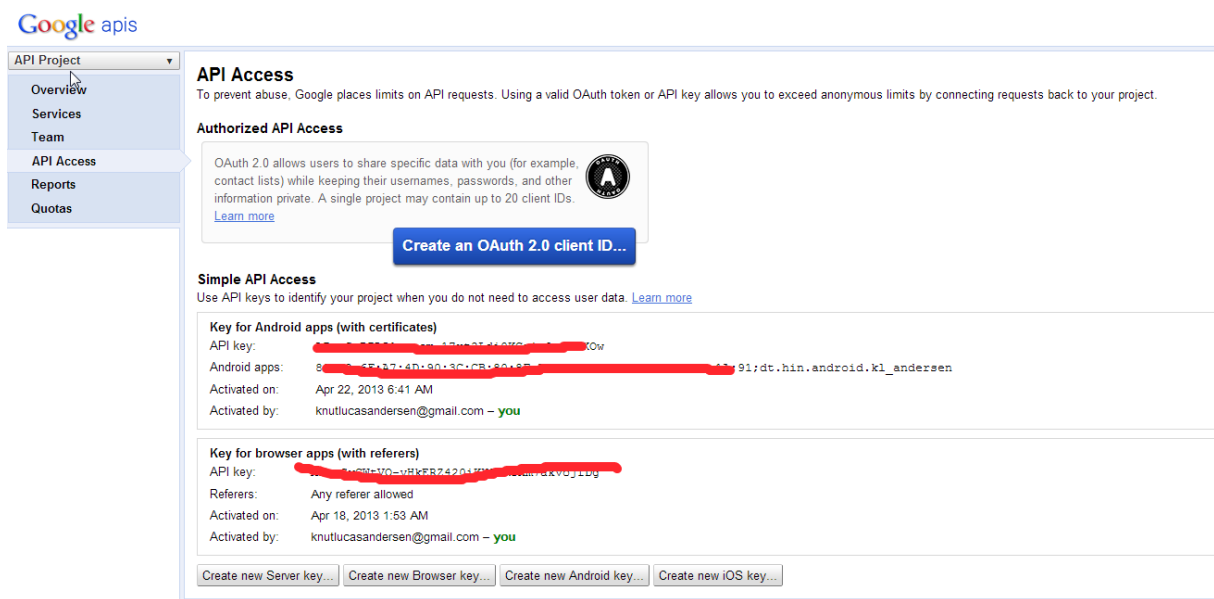
<sup>6</sup> <http://developer.android.com/google/gcm/gcm.html#arch> ---> RegistrationID

## ApplikasjonsID:

Android applikasjonen som har tillatelse til å motta meldinger fra GCM. Identifikasjonen for dette er pakkenavnet som brukes for applikasjonen (*pakkenavnet er det som settes for å generere browser key under API Access*<sup>5</sup>).

## Senders autentiseringssymbol:

API nøkkel som lagres på applikasjonsserveren, for å gi server autentisering tilgang til Google services. Denne nøkkelen får du via API Access<sup>5</sup>. I vedlegg 4 kan du se servleten som tar for seg utsending av melding til tilkoblede enheter.



The screenshot shows the Google APIs console interface. On the left is a sidebar with a menu: API Project, Overview, Services, Team, API Access (selected), Reports, and Quotas. The main content area is titled 'API Access' and includes a sub-header 'Authorized API Access'. Below this, there is a section for 'Simple API Access' which lists two keys: 'Key for Android apps (with certificates)' and 'Key for browser apps (with referers)'. Each key entry shows the API key, the Android apps or referers, the activation date, and the user who activated it. At the bottom of the console, there are buttons to 'Create new Server key...', 'Create new Browser key...', 'Create new Android key...', and 'Create new iOS key...'.

Som en avsluttende kommentar kan det være verdt å nevne at Google Cloud Messaging (GCM) har tatt over for Cloud To Messaging Device (C2DM)<sup>7</sup>.

<sup>7</sup> <http://developer.android.com/google/gcm/c2dm.html>

# Android - applikasjonen

---

Det originale navnet jeg ga applikasjonen var TrackMyPosition. Men, når jeg startet på å skrive rapporten, så kom jeg på at det er en ting som heter Copyright. Ved googling fant jeg ut at dette navnet allerede var i bruk, det samme gjaldt for andre navn jeg tenkte på. Jeg ga derfor applikasjonen navnet TracknHide. Jeg valgte dette navnet fordi at under utviklingen, når jeg fikk til å tegne flere ruter på kartet, så tenkte jeg for meg selv at dette måtte være fremtidens svar på gjemsel!

Et oppstartsproblem som spiste mye tid var det faktum at Google Maps API lvl1 var blitt deprecated. Jeg fikk nøkkel til Google Maps API lvl1, men uansett hva jeg gjorde, så fikk jeg kun gråe ruter på kartet. Problemet med Google Maps API lvl2 er at denne kan kun kjøre på ekte enheter, og ikke i emulator. Jeg forsøkte å finne måter å kjøre Google Maps lvl2 i emulator, men til ingen nytte. Det er her og verdt å nevne at det finnes artikler<sup>8</sup> og forslag til måter å få dette til å funke, men som det og er nevnt, så er dette såkalt "Software Piracy"<sup>9</sup>. Grunnen til dette er at de to .apk'ne som nevnes i artikkelen er programvare som er hentet ut fra en fysisk enhet. Det finnes måter å hente ut disse fra egen enhet og installere på emulator, men det går jeg ikke noe videre inn på i denne rapporten.

Jeg kan som en avsluttende kommentar nevne at løsningen for å få Google Maps API lvl1 til å fungere var å legge en fil kalt debug.keystore i min .android mappe. Filen debug.keystore kom fra samme person som jeg fikk nøkkelen fra og dette er da løsningen dersom det skulle være behov for å bruke Maps API lvl1 med andres nøkler.

## *MainMapActivity og Google Maps*

MainMapActivity er navnet på hovedaktiviteten som lastes inn når applikasjonen starter, og det er her bruker får se kartet. Klassen arver fra MapActivity og implementerer LocationListener. Her kan bruker:

- Starte/Stoppe sending av posisjonsdata
- Vise logg over meldinger mottatt fra GCM, feilmelding og adresser (*SystemloggActivity*)
- Vise historikk over lagrede ruter (*VisRuterActivity*)
- Endre innstillinger for hva som skal lagres og hva som skal vises på kart (*InnstillingsActivity*)

Det første som skjer når bruker starter opp applikasjonen er at det sjekkes at bruker er tilkoblet internett. Deretter lastes Google Maps inn og (*forsøk på*) tilkobling til GCM og server opprettes. På neste side vises kodesnutt med forsøk på oppkobling til GCM.

---

<sup>8</sup> <http://nemanjakovacevic.net/blog/2012/12/how-to-make-android-google-maps-v2-work-in-android-emulator/>

<sup>9</sup> <http://stackoverflow.com/questions/14445093/how-to-show-google-map-v2-on-android-emulator>

```

//hent denne enhetens registreringsID
final String registrationID = GCMRegistrar.getRegistrationId(context);
//er enhet registrert fra før?
if (registrationID.equals("")) {
    //enhet ikke registrert, hent registrationID for denne enheten
    GCMRegistrar.register(context, context.getString(R.string.sender_id));
} else {
    //er enhet registrert på server?
    if (GCMRegistrar.isRegisteredOnServer(context)) {
        String feilmelding = context.getString(R.string.already_registered);
        Filbehandling.oppdaterSystemLogg(context, feilmelding);
        Toast.makeText(context, feilmelding, Toast.LENGTH_LONG).show();
        //registrert på server, hent ut brukers registrationID fra sharedPreferences
        String regID = deltPreferanse.getString(Filbehandling.REGISTRATIONID_NOKKEL, "");
        //sjekker for sikkerhets skyld at det er en verdi registrert i SP
        if (!regID.equals("")) {
            //se utdypende kommentar i ServerTilkobling.gjenopprettServerTilkobling()
            //kort forklart: forsikrer at bruker er tilkoblet vår server
            mAsyncTraad = new AsyncTraad((MainMapActivity) context, regID,
            AsyncTraad.GJENOPRETT_SERVER_TILKOBLING);
            mAsyncTraad.execute();
        } //if (!regID.equals(""))
    } else {
        //siden det er nødvendig å kansellere tråd via onDestroy() brukes AsyncTask
        mAsyncTraad = new AsyncTraad((MainMapActivity) context, registrationID,
        AsyncTraad.AVREGISTRERING);
        mAsyncTraad.execute();
    } //if (GCMRegistrar.isRegisteredOnServer(context))
} //if (registrationID.equals(""))

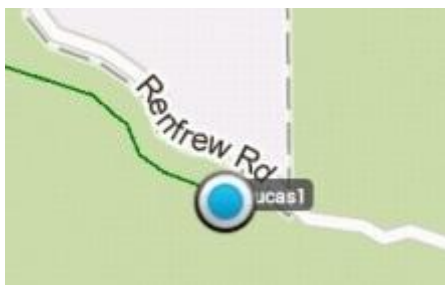
```

Foruten å bruke klassen MapOverlay for å tegne ruter på kartet, så har jeg og måttet ta i bruk et objekt av klassen MyLocationOverlay. Det som skjedde var nemlig at når jeg fjernet funksjonskallet på MyLocationOverlay.enableMyLocation()<sup>10</sup>, så tok det over 1 - 3 minutter før forflytning på kartet skjedde (*selv om det var kort distanse*). Det tok såpass lang tid før onLocationChanged() ble kalt. Og dersom jeg fjernet List<Overlay> så ble ikke rutene tegnet på kartet (*kodesnutt vist under*).

```

//opprett mapoverlay og objekt som skal vises (mylocationoverlay)
mapOverlay = new MapOverlay(context);
overlayList = mapView.getOverlays();
overlayList.add(mapOverlay);
MyLocationOverlay myLocationOverlay = new MyLocationOverlay(this, mapView);
overlayList.add(myLocationOverlay);
myLocationOverlay.enableMyLocation();

```



<sup>10</sup> enableMyLocation() gjør at bruker får se en blinkende "knott" som viser brukers siste lokasjon (se bilde over)



## MapOverlay

MapOverlay er en klasse som arver fra Overlay. Denne klassen gjør det mulig å tegne på kartet, og i tillegg bruker jeg denne klassen for å opprette en popup-dialog som gir brukeren følgende valg:

- Vise adresse (*på lokasjon klikket*)
- Skifte mellom StreetView og SateliteView
- Finne sin lokasjon
- Vise brukere som er online (*vises etter applikasjon er startet og har mottatt koordinater*)

En ting som jeg slet med under utviklingen var tegning av rutene på kartet. Hvordan skulle jeg i første omgang ta vare på alle koordinatene? Hvordan skulle jeg tegne rutene, for ikke glemme adskillige rutene, slik at ting ikke ble tegnet på kryss og tvers?

Løsningen jeg endte opp med var å legge alle brukere som var online i en ArrayList. ArrayList'n var basert på en klasse jeg kalte for Enhet<sup>11</sup>. I Enhets-klassen hadde jeg da en egen ArrayList<GeoPoint> som tok vare på enhetens koordinater, og som gjorde at jeg hadde en mye ryddigere og lettere tilgang til hver enkelt enhets koordinater. Litt mer om dette i slutten av kapitlet.

Tanken min når det kom til tegning, var at det var ønskelig at bruker var den eneste med sin farge på kartet, for å gjøre den unik. Det som jeg slet med her var egentlig selve kontrollen av dette, pga tilgjengelige farger ligger i en Array. Siden arrays har en fast størrelse og jeg ville at brukers farge skulle være unik, så måtte jeg passe på at indeksen ble inkrementert, men og at den ble nullstilt når indeks ble større enn arrays størrelse. Dette var et problem som kvernet rundt i hodet mitt:

*"Inkrementering av indeks må skje før tegning av hver rute. Samtidig må det sjekkes at indeks ikke er større enn array, og i tillegg sjekkes at indeks ikke er satt til brukers farge dersom det er tegning av andres rute. Men hva om indeks 0 er brukers farge? Samtidig kan brukers farge være alt fra 0 - n. Hvordan skrive en funksjonell kode som gjør nettopp dette uten å bli gjentakende?"*

```
private static int hentFargeSomSkalBrukes(int fargeIndeks, boolean egenEnhet) {
    int antFarger = _fargeArray.length - 1;
    //hent ut indeks for hvilken farge brukers rute skal vises i
    int brukersValg = _deltPreferanse.getInt(Filbehandling.RUTEFARGE_NOKKEL, 0);
    //er dette brukers egen enhet?
    if (egenEnhet) {
        fargeIndeks = brukersValg;
    } else {
        //øk farge og kontroller om fargen må nullstilles
        fargeIndeks++;
        fargeIndeks = nullStillFargeIndeks(antFarger, fargeIndeks);
        //hvis farge ble nullstilt, eller hvis fargeIndeks ble økt,
        //sjekk at nåværende farge ikke er den samme som brukerens farge
        if (brukersValg == fargeIndeks) {
            fargeIndeks++;
            fargeIndeks = nullStillFargeIndeks(antFarger, fargeIndeks);
        } //if (brukersValg == _fargeIndeks)
    } //if (egenEnhet)
    return Color.parseColor(_fargeArray[fargeIndeks]);
} //hentFargeSomSkalBrukes
```

---

<sup>11</sup> Klassen Enhet kan du se i vedlegg 2, s. 28

```
private static int nullStillFargeIndeks(int antFarger, int indeksFarge) {
    if (indeksFarge > antFarger) {
        indeksFarge = 0;
    } //if (indeksFarge > antFarger)
    return indeksFarge;
} //nullStillFargeIndeks
```

En annen ting jeg merket meg som utviklingen progresserte var at dersom emulator ble inaktiv (*belysning forsvinner*), så sluttet ruten å bli tegnet (*ved tegning av andre brukers rute*). Min løsning på dette var å opprette en funksjon i MainMapActivity som invaliderer kartet hver gang bruker mottar nye koordinater. Dette gjorde til at ruten fortsatt ble tegnet, selv om belysning forsvant fra skjermen. Her er koden for dette (*bruker postInvalidate() siden kall kommer utenfra*):

```
public static void invaliderMapView() {
    mapView.postInvalidate();
} //invaliderMapView
```

Det å tegne enhetens navn på kartet var og en jobb. Problemet var egentlig knyttet til lengden på navnet og rektangelet som tegnes i bakgrunnen for å fremheve navnet<sup>12</sup>. Jeg fant egentlig ikke en god løsning på det, annet enn at jeg tok et lengre navn, og testet med tallverdier for å komme frem til en ca. verdi som gjorde at rektanglet stemte sånn noenlunde. Det eneste problemet er at ved tegning av historikk (*lagrede ruter*) så stemmer ikke lengden overens, da det i noen tilfeller blir altfor langt rektangel. Et annet problem var at navnene ble tegnet oppå hverandre og ikke oppå tilhørende enhets rute. Grunnen til dette var at jeg brukte en lokal, og ikke global variabel av typen Point.

Så hvordan tegnes rutene egentlig på kartet? Dette avhenger litt av brukers innstillinger. Når bruker starter opp enheten, så vil enheten registreres hos GCM og opprette tilkobling til server for å motta posisjonsdata. Posisjonsdataene blir så lagret i databasen, og de blir liggende der til bruker enten sletter de eller applikasjon avsluttes<sup>13</sup>.

Et litt mer beskrivende eksempel. Bruker mottar posisjonsdata, som lagres i databasen. Bruker starter så opp applikasjonen og mottar nye posisjonsdata. Dataene vil da fortsatt lagres i databasen, men de blir også oversendt til en klasse kalt FellesFunksjoner, som har en funksjon som fyller opp en ArrayList med enheter som er online. Det blir da sjekket om koordinatene som er mottatt tilhører en enhet som er registrert. Dersom enheten ikke er registrert (*allerede vist på kartet*), opprettes enheten ved å lage et objekt av klassen Enhet.

Som nevnt tidligere, har klassen Enhet en egen ArrayList<GeoPoint>. For å sørge for at ruten som tegnes på kartet skal være fullstendig (*vise eksisterende og tidligere mottatte data*), så blir det da sjekket mot databasen om denne enheten med gitt registrationID har koordinater lagret. Hvis enhet har koordinater i databasen, så hentes disse ut, legges i en ArrayList<GeoPoint> som deretter returneres. Deretter blir de siste mottatte koordinatene (*de bruker mottok etter applikasjon startet*) lagt inn bakerst i ArrayList. Så kalles funksjonen MainMapActivity.invaliderMap() som oppdaterer tegning på kartet.

<sup>12</sup> Grunnlaget for tegning av enhetens navn er basert på: Professional Android 4 Appl. Dev., Reto Meier, s. 549 - 552

<sup>13</sup> Sletting av posisjonsdata ved avslutning forutsetter at bruker ikke vil beholde posisjonsdataene

## *InnstillingsActivity, SystemloggActivity og VisRuterActivity*

Felles for disse er at de er alle sub-aktiviteter, men med hver sin separate hensikt. InnstillingsActivity er, som navnet tilsier en aktivitet som inneholder innstillinger tilknyttet applikasjonen og visning på kartet. Bruker får her muligheter til å:

- Navngi enhet (*navn som skal vises på kartet*)
- Sette hvor ofte posisjonsdata skal sendes (*i minutt og meter*)
- Hvilken farge bruker ønsker sin rute skal ha på kartet og default zoom nivå
- Lagring av egne og andres posisjonsdata (*rutehistorikk*)
- Logging av meldinger (*feilmeldinger, til/frakoblingsmeldinger og melding fra GCM*)
- Visning av til/frakoblingsmelding(er) tilknyttet GCM og Server ved oppstart og avslutning

Disse innstillingene blir lagret i SharedPreferences og lastes inn når InnstillingsActivity vises.

SystemloggActivity fremviser loggede meldinger, hvor her kan nevnes:

- Feilmeldinger
- Registrering/Avregistrering GCM og server
- Meldinger som mottas fra GCM
- Adresse via popup-dialogs menyvalg

VisRuterActivity fremviser lagrede ruter, hvis det finnes noen. Det kan nevnes her at komponentmessig er SystemloggActivity og VisRuterActivity ganske like, siden de begge kun inneholder en ListView og en Button. Men, jeg valgte allikevel å holde de separate av den grunn at oppgavene som utføres er for forskjellig.

SystemloggActivity fremviser kun en liste over loggede hendelser og muligheten til å slette disse. Mens VisRuterActivity lister opp lagrede ruter fra databasen, og foruten muligheten til å slette disse, så kan bruker klikke på en rute og få denne tegnet opp på kartet. En utvidelsesmulighet hadde vært å inkludere Checkboxes slik at bruker kunne velge hvilke ruter som skulle slettes/tegnes. Dette fikk jeg ikke tid til, så jeg valgte å holde den "enkel", ved at bruker kan velge å slette alle fremviste ruter, eller klikke på en rute for å få denne tegnet på kartet. Men, bruker kan velge flere ruter fra historikken og få disse tegnet. Bruker kan deretter fjerne ruter tegnet på kartet ved å trykke "Fjern historikk fra kartet".

En annen ting verdt å merke seg når det kommer til VisRuteActivity er at den lister kun opp ruter for brukere som ikke er online. Siden bruker kan motta posisjonsdata uten å ha startet applikasjon, så vil historikk ligge i databasen. Men, dersom historikken er ny (*bruker er online*) vil ikke denne være tilgjengelig i historikken før bruker går offline. Dette er for å unngå at bruker sletter ruter som blir tegnet på kartet, men også for å unngå at historikk tegnes opp på eksisterende ruter.

Problemet som oppsto under utviklingen av denne funksjonaliteten var at etter jeg hadde fått til tegning av historikk på kartet, så ble ikke brukers rute tegnet på kartet. Man kunne se bruker forflytte seg via markøren man får fra MyLocationOverlay, men ingen farge eller streker kom opp. Mer merkelig var det da jeg avsluttet og startet opp applikasjon og brukers rute ble tegnet opp og ikke nok med det, den hadde t.o.m. lagret historikken i databasen. Så hvorfor ble ikke brukers rute tegnet når jeg tegnet historikk?

Dette ble en virkelig jobb å finne ut av, og det endte med at jeg klarte å lokalisere feilen til min AsyncTask klasse. Det viste seg nemlig at funksjonen `doInBackground(Void... params)` aldri ble kalt. Jeg forsøkte å google med forskjellige formuleringer, men fant få svar som var relevant med mitt problem. Til slutt var jeg heldig. På StackOverflow var det en bruker som hadde et lignende problem, hos denne brukeren var problemet at `doInBackground` kastet Exception<sup>14</sup>.

Jeg puttet da en try/catch i min `doInBackground`, og der var faktisk problemet. Problemet var det faktum at jeg hadde satt at historikk ruter skulle ikke ha en `registrationID` (*jeg satte verdien til null*), for å unngå blanding med eksisterende/online ruter. Feilen lå i det at i funksjonen hvor jeg hentet ut indeks for enheter online, så brukte jeg String funksjonen `equals`. `equals` sjekket da mot `registrationID` (*som var null ved historikk*) og dette kastet en `NullPointerException`. Løsningen ble å legge til en ekstra sjekk om at valgt `registrationID` ikke var null, og deretter funket det. En viktig lærepenge der, AsyncTask kaster ikke feilen til grensesnittet, den bare avslutter oppgaven uten å si i fra.

```
protected static int finnEnhetsIndeks(String registrationID) {
    int indeks = -1,
        teller = 0;
    boolean funnet = false;
    String gjeldendeID = "";
    //loop gjennom arraylist til ID er funnet eller alle elementer er sjekket
    while (!funnet && teller < _onlineEnheter.size()) {
        gjeldendeID = _onlineEnheter.get(teller).getRegistrationID();
        if (gjeldendeID != null && gjeldendeID.equals(registrationID)) {
            funnet = true;
            indeks = teller;
        } //if (gjeldendeID.equals(registrationID))
        teller++;
    } //while
    return indeks;
} //finnEnhetsIndeks
```

### SQLite, filer og lagring av posisjonsdata/ruter

I denne applikasjonen bruker jeg ikke filer så mye. Lagringsmåten som skulle brukes var noe jeg brukte ekstremt mye tid på (*noe som også var tilknyttet utviklingen av klassen Enhet*). Det som jeg måtte ta hensyn til var nettopp hva som skulle lagres og hvordan det som ble lagret skulle bli hentet ut. Problemet her var jo mengden data og muligheten til å kunne skille dataene fra hverandre. Jeg vurderte først å bruke filer, siden det ville være raskere enn database. Men, så var det nettopp dette med mengden data som skulle tas vare på.

Et GeoPoint består av latitude og longitude og jeg måtte vite hvilken bruker ruten tilhørte (*og om dette var applikasjons brukerens rute*). Dette pga at bruker skal kunne velge å ta vare på alle, ingen eller egne/andres ruter. En mulighet jeg tenkte på var å bruke `registrationID` som filnavn, men hvordan skulle jeg da i ettertid finne igjen filen? En kunne jo da lagd en egen fil som tok vare på `registrationID`, men der igjen så var det mengden filer en ender opp med. Og som nevnt i første kapittel, `registrationID` er ikke 100% unik for en enhet. Filnavn og identifisering til side, hvordan skulle jeg lese inn og skrive tilbake innhold på en lett måte?

<sup>14</sup> <http://stackoverflow.com/questions/8597088/async-task-never-executing-doinbackground-but-does-in-debug-update-fixed>

En tanke som slo meg var å sette at første linje var enhetens navn, neste linje inneholdt latitude og longitude, og resten av filen var resterende koordinater. Men så er det jo nettopp den jobben å lese og skrive korrekte data til fil. Og skrive funksjonell kode og holde styr på alt. Etter mye om og men, og mye "selvransaking" så kom jeg frem til at database ville være enklere. Lett å holde styr på, rask tilgang til data (*strukturemessig*) og jeg ville slippe å styre med hvordan ting skulle leses, skrives og navngis.

Men så var det denne registrationID'n da. Jeg trodde denne var unik, at en enhet ble tildelt en ID, og så hadde enheten ID'n til den ikke var mer. Dette stemte ikke. Jeg baserte den originale databasestrukturen med registrationID som primærnøkkel i tblEnhet og fremmednøkkel i tblKoordinater. Slik kunne brukere ha flere ruter, hvor hver rute var tilknyttet sin enhets registrationID.

Det var faktisk helt tilfeldig at jeg fant ut at registrationID ikke var unik. Jeg satt og testet lagring av ruter for å kontrollere at rett ruteID<sup>15</sup> ble generert, da det i LogCat plutselig sto ruteID = 1, enda forrige utskrift så sto det ruteID = 2. Min forståelse basert på tilhørende tekst på Android Developer<sup>16</sup> (og *egen testing*) er at registrationID eksisterer kun så lenge enhet er tilkoblet GCM's server.

Dette var en brutal oppvåkning for min del. Jeg hadde brukt hele dagen og den forrige på testing, korrigering og ferdigstilling av funksjonalitet til en database som nå var fullstendig ubrukelig<sup>17</sup>. Spørsmålet jeg nå måtte stille meg var: "*Hvordan skal data lagres og adskilles? Kan databasen fortsatt brukes?*" Siden jeg nå hadde mistet den unike nøkkelen som skilte de ad, var jeg ikke sikker på hvilken struktur og filløsning jeg skulle gå for.

Løsningen jeg kom frem til var å beholde databasestrukturen jeg hadde, med noen få modifikasjoner. Jeg gikk over til å bruke en auto-inkremerende integer som primærnøkkel for enheter og fjernet registrationID fra begge tabellene. Problemet ble bare det at når jeg nå la til enheter som ikke var brukers, så fikk alle tilkoblede brukere samme primærnøkkel...

Grunnen til dette var egentlig ikke så rar. Spørringen min baserte seg på å hente ut primærnøkkel basert på at den skulle være lik fremmednøkkel i tblKoordinater og siste registrerte rutetidspunkt kortere enn en dag gammel.

Konklusjonen ble at den endelige databasestrukturen ikke ble endret noe særlig. Jeg endte opp med å legge registrationID tilbake i tblKoordinater for å adskille tilkoblede enheter, og bruke auto-inkrement som primærnøkkel i tblEnhet. Men, det førte til at jeg måtte oppdatere Enhetsklassen ved å legge til en identifikator variabel, som da er enhetens primærnøkkel i databasen. Grunnen til dette var for å sørge for at rett posisjonsdata ble lagret på rett bruker.

Jeg bruker også SharedPreferences en god del. Foruten å ta vare på brukerens innstillinger, så lagrer jeg også enhetens navn, registrationID, siste posisjon (*latitude og longitude*) og primærnøkkelen. Dette kan egentlig virke litt bortkastet, med tanke på at dette er informasjon som ligger i databasen. Men, det er viktig å huske på at databaser tar tid. Mye tid, hvis man tar i betraktning den mengde med posisjonsdata som til slutt kan ligge avhengig av intervall som posisjonsdata sendes/mottas.

---

<sup>15</sup> RuteID er en "identifikator" som jeg bruker for å sette alle koordinatene som mottas sammen til en rute

<sup>16</sup> <http://developer.android.com/google/gcm/gcm.html#arch>

<sup>17</sup> Original og ny databasestruktur kan du se i Vedlegg 5, s. 36

For å få rask tilgang, så legger jeg overnevnte verdier i SharedPreferences for å raskere få tilgang til data som brukes ofte. Samtidig så er det viktig at disse dataene ligger også i databasen, nettopp fordi at de blir brukt. Rutehistorikk f.eks. Her brukes bl.a. enhetens navn for å hjelpe til å identifisere rutene som listes opp. Samtidig slipper en å måtte gå inn i databasen for å hente ut brukerens siste posisjon, siden disse ligger lett og raskt tilgjengelig via SharedPreferences.

Som en avsluttende kommentar til dette delkapitlet skal jeg nevne et interessant problem som oppsto ved bruk av SQLite som database og det å ha flere tabeller i denne. Jeg glemte ut det faktum at i funksjoner som overrides, så ligger det kun kall på en tabell (*se kodesnutt under*) i grunnlaget som vises i lærebok<sup>18</sup>.

```
//hent radens ID
String radID = uri.getPathSegments().get(1);
selection = TBLENHET_PRIMARY_KEY + "=" + radID
//hvis selektering ikke er tom, legg den med via 'AND'
+ (!TextUtils.isEmpty(selection) ?
" AND (" + selection + ") : "");
```

Problemet jeg her skal nevne var tilknyttet sletting fra databasen. Jeg hadde en enkel where-klasul:

```
//denne lå først for å slette fra tblKoordinater
String where = fkEnhet + " != " identifikator;

//etter sletting fra tblKoordinater kom denne where-klasulen
where = pkEnhet + " != " identifikator;
```

Jeg forsøkte med **" != "**, med IS NOT, NOT LIKE og <> for å slette spesifiserte ruter (*både fra tblEnhet og tblKoordinater*). Uansett hva jeg skrev, så slettet den alle koordinatene som brukers enhet hadde i databasen.

Jeg satt altfor lenge og dunket hodet i veggen, før jeg tilslutt lagde en egen funksjon for sletting, hvor sletting gikk rett inn i databasen.

Da jeg senere leste over koden (*i ContentProvider klassen*), fant jeg ut at som en kan se i ovenstående kodesnutt, at det ville (*uansett hva jeg hadde skrevet i where*) kun bli slettet ved å bruke tblEnhet sin primærnøkkel. Jeg har ikke prøvd ut den Override'de delete funksjonen etter jeg oppdaterte den til å hente og sjekke fra rett database-tabell. Det orket jeg rett slett ikke, og siden "min" delete funksjon fungerte, så jeg heller ikke noe poeng i å bruke mer tid på det.

---

<sup>18</sup> Professional Android 4 Application Development, Reto Meier, s. 261-262

## GCMIntentService og tilkobling til via ServerTilkobling

GCMIntentService er det ikke så mye for meg så skrive noe om. Den er basert på eksemplet som medfølger android-sdk i mappen .\extras\samples\gcm-demo-client. Det eneste unntaket er at jeg har skrevet om mottak av meldinger og notifications som vises til bruker (*vist under*).

@Override

```
protected void onMessage(Context context, Intent intent) {
    String melding = "",
        latitude = "",
        longitude = "",
        enhetsNavn = "",
        registrationID = "";

    //uthenting av meldinger - hva slags melding er mottatt?
    if (intent.getStringExtra(context.getString(R.string.readable_message)) != null) {
        melding = getString(R.string.gcm_message);
        melding += intent.getStringExtra(context.getString(R.string.readable_message));
        FellesFunksjoner.visMelding(context, melding, true);
        opprettNotification(context, melding);
    } else {
        if (intent.getStringExtra(PARAMETER_ENHET_GIKK_OFFLINE) != null) {

            registrationID = intent.getStringExtra(PARAMETER_ENHET_GIKK_OFFLINE);
            FellesFunksjoner.fjernOfflineEnhet(registrationID);
        } else {
            //sjekk at verdi er oversendt og hent ut verdiene
            if (intent.getStringExtra(Filbehandling.REGISTRATIONID_NOKKEL) != null) {
                registrationID =
            intent.getStringExtra(Filbehandling.REGISTRATIONID_NOKKEL);
            } //if (intent.getStringExtra(PARAMETER_REGISTRATION_ID) != null)
            if (intent.getStringExtra(PARAMETER_ENHETSNAVN) != null) {
                enhetsNavn = intent.getStringExtra(PARAMETER_ENHETSNAVN);
            } //if (intent.getStringExtra(PARAMETER_ENHETSNAVN) != null)
            if (intent.getStringExtra(Filbehandling.LATITUDE_NOKKEL) != null) {
                latitude = intent.getStringExtra(Filbehandling.LATITUDE_NOKKEL);
            } //if (intent.getStringExtra(Filbehandling.LATITUDE_NOKKEL) != null)
            if (intent.getStringExtra(Filbehandling.LONGITUDE_NOKKEL) != null) {
                longitude =
            intent.getStringExtra(Filbehandling.LONGITUDE_NOKKEL);
            } //if (intent.getStringExtra(Filbehandling.LONGITUDE_NOKKEL) != null)

            lagreKoordinater(context, registrationID, enhetsNavn, latitude, longitude);
        } //if (intent.getStringExtra(PARAMETER_ENHET_GIKK_OFFLINE) != null)
    } //if (intent.getStringExtra(LESBAR_MELDING) != null)
} //onMessage
```



```

private void opprettNotification(Context context, String melding) {
    int ikon = R.drawable.icon_notification;
    //notificationmanager som håndterer notifications (hører innunder system service)
    NotificationManager notificationManager = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
    //bruker NotificationCompat.Builder siden API < 11
    NotificationCompat.Builder notification = new NotificationCompat.Builder(context)
        .setContentType(context.getString(R.string.app_name))
        .setContentTitle(melding)
        .setSmallIcon(ikon);
    //oppretter intent
    Intent notificationIntent = new Intent(context, MainMapActivity.class);
    //sørg for at intent ikke starter ny aktivitet
    notificationIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
    //oprett pendingintent
    PendingIntent notifyIntent = PendingIntent.getActivity(context, 0, notificationIntent,
PendingIntent.FLAG_UPDATE_CURRENT);
    //legg pendingintent inn i notification
    notification.setContentIntent(notifyIntent);
    //oppretter notification objekt og sender det til notificationmanager
    notificationManager.notify(NOTIFICATION_ID, notification.build());
} //opprettNotification

```

Klassen ServerTilkobling er klassen som håndterer til/frakobling til server. Den er grovt basert på klassen ServerUtilities i gcm-demo-client, men jeg har gjort endringer slik som at ved registrering på egen server så blir brukers enhet også lagret i databasen (SQLite). Ved tilkobling blir det også kontrollert om bruker ønsker å sende sine posisjonsdata til server. Dersom bruker ikke vil sende sine posisjonsdata, sendes en default verdi (999.0). Fra ServerTilkobling er det tre funksjoner jeg vil vise.

Den første er en funksjon som sender en offline melding til server. Offline melding er en melding som sendes til server for å fortelle at en enhet har avsluttet applikasjonen, og ikke lenger sender sine posisjonsdata. Hensikten med denne funksjonen er at dersom bruker ønsker å motta posisjonsdata etter applikasjon er avsluttet, ville server (og tilkoble enheter) aldri få vite at bruker har avsluttet. Derfor kalles denne funksjonen ved avslutning, men dersom bruker ikke vil fortsette å motta posisjonsdata blir enhet avregistrert fra server og GCM (dette gjøres i onDestroy()).

```

protected static void sendOfflineMeldingTilServer(Context context, String registrationID) {
    try {
        String serverUrl = SERVER_URL + SEND_MELDING_TIL_SERVER;
        Map<String, String> parametere = new HashMap<String, String>();
        parametere.put(PARAMETER_OFFLINE_MELDING, PARAMETER_OFFLINE_MELDING);
        parametere.put(Filbehandling.REGISTRATIONID_NOKKEL, registrationID);
        postOnServer(serverUrl, parametere);
    } catch (IOException ex) {
        String feilmelding = ex.getMessage();
        Filbehandling.oppdaterSystemLogg(context, feilmelding);
        FellesFunksjoner.visMelding(context, feilmelding, true);
    } //try/catch
} //sendOfflineMeldingTilServer

```



Den andre funksjonen gjenoppretter tilkobling til server. Dette er et behov hvis bruker har mistet tilkobling til server, men fortsatt er registrert på GCM siden da hoppes serverregistrering over<sup>19</sup>. Det er her verdt å merke seg at ved å trykke på "Start sending av posisjonsdata", så vil ikke data sendes dersom location er null (*gjelder for emulator*).

```
protected static void gjenopprettTilkoblingTilServer(Context context, String registrationID) {
    try {
        String serverUrl = SERVER_URL + SEND_MELDING_TIL_SERVER;
        String defaultVerdi = context.getString(R.string.default_enhet_navn);
        _deltPreferanse = Filbehandling.getSharedPreferences(context);
        //fyller opp Map med verdier som skal sendes til server
        Map<String, String> parametere = new HashMap<String, String>();
        parametere.put(Filbehandling.REGISTRATIONID_NOKKEL, registrationID);
        parametere.put(PARAMETER_ENHETSNAVN,
        _deltPreferanse.getString(Filbehandling.ENHETSNAVN_NOKKEL, defaultVerdi));
        Float ikkeSend = Float.parseFloat(IKKE_SEND_POSISJON);
        parametere.put(Filbehandling.LATITUDE_NOKKEL, ikkeSend.toString());
        parametere.put(Filbehandling.LONGITUDE_NOKKEL, ikkeSend.toString());
        postOnServer(serverUrl, parametere);
    } catch (IOException ex) {
        String feilmelding = ex.getMessage();
        Filbehandling.oppdaterSystemLogg(context, feilmelding);
        FellesFunksjoner.visMelding(context, feilmelding, true);
    } //try/catch
} //gjenopprettTilkoblingTilServer
```

Den tredje funksjonen som tar for seg sending av posisjonsdata til server:

```
protected static void sendKoordinaterTilServer(Context context, String registrationID) {
    try {
        String serverUrl = SERVER_URL + SEND_MELDING_TIL_SERVER;
        String defaultVerdi = context.getString(R.string.default_enhet_navn);
        _deltPreferanse = Filbehandling.getSharedPreferences(context);
        //fyller opp Map med verdier som skal sendes til server
        Map<String, String> parametere = new HashMap<String, String>();
        parametere.put(Filbehandling.REGISTRATIONID_NOKKEL, registrationID);
        parametere.put(PARAMETER_ENHETSNAVN,
        _deltPreferanse.getString(Filbehandling.ENHETSNAVN_NOKKEL, defaultVerdi));
        float defValue = Float.parseFloat(IKKE_SEND_POSISJON);
        Float lat = _deltPreferanse.getFloat(Filbehandling.LATITUDE_NOKKEL, defValue),
        lon = _deltPreferanse.getFloat(Filbehandling.LONGITUDE_NOKKEL, defValue);
        parametere.put(Filbehandling.LATITUDE_NOKKEL, lat.toString());
        parametere.put(Filbehandling.LONGITUDE_NOKKEL, lon.toString());
        postOnServer(serverUrl, parametere);
    } catch (IOException ex) {
        String feilmelding = ex.getMessage();
        Filbehandling.oppdaterSystemLogg(context, feilmelding);
        FellesFunksjoner.visMelding(context, feilmelding, true);
    } //try/catch
} //sendKoordinaterTilServer
```

<sup>19</sup> Et eksempel på et tilfelle hvor dette kan skje er hvis bruker blir frakoblet og har valgt mottak av posisjonsdata etter applikasjonen avsluttes. Da vil ikke applikasjonen forsøke å koble til server

# Server

---

Å lage en server ved å bruke Java og servlets er noe jeg aldri har gjort før, og det var interessant å lære. Grunnlaget for serveren er basert på medfølgende eksempel i `.\extras\samples\gcm-demo-server`. Noen ting verdt å merke seg ved utvikling av server er følgende:

- Dersom servlet starter med default package og package senere legges til, må pakkenavn legges foran klassenavn i `WEB-INF/web.xml`
- Servlets bruker perspektivet Java EE med TomCat
  - må laste ned alt av Web, XML og EE via Help ---> Install new software
- Ved bruk av Google Cloud Messaging (GCM) må `gcm-server.jar` legges i `WebContent\WEB-INF\lib` mappen
- Ved bruk av MySQL i servlets må `mysql-connector-java` legges i `WebContent\WEB-INF\lib`
- For å unngå SQL-Injection, må en bruke `PreparedStatement`<sup>20</sup> istedenfor `Statement` ved kjøring av spørringer basert på input fra bruker
- Android emulators localhost er ikke det samme som Servers localhost.  
Servers localhost: `http://localhost:8080/Karaktergivende_Server_KLA/`  
Emulators localhost<sup>21</sup>: `http://10.0.2.2:8080/Karaktergivende_Server_KLA`

## *Forelderklassen og arvede funksjoner*

`ForelderServlet`<sup>22</sup> er navnet på den abstrakte klassen som alle servletene arver fra. Det finnes totalt seks servlets inkludert `ForelderServlet`, hver med sin egen oppgave. Det ble anbefalt å lage en servlet for hver oppgave eller en servlet for alt. Jeg valgte da å gå for det første, siden jeg personlig syns det er mer oversiktlig. Servletenes oppgave går i hovedsak ut på å motta og sende informasjon til og fra tilkoblede enheter (*Servlet for innlogging skriver jeg om i kapittelet for nettsider*).

For at en enhet skal kunne motta posisjonsdata fra server, må den først tilkobles. Dette skjer via `RegistrerEnhet`. `RegistrerEnhet` henter ut mottatte parameter og registrerer enhet i databasen. I tillegg videresendes enhetens posisjonsdata til andre enheter, dersom faktiske koordinater er mottatt (*bruker kan tross alt velge å ikke sende egne posisjonsdata*). Enhet er tilkoblet server til a) enhet avregistrer seg fra server (*dette skjer via AvregistrerEnhet*) eller b) administrator frakobler enhet fra nettsiden (*nettsidene skrives om i eget kapittel*). Når enhet frakobles server (*likegyldig grunn*), så slettes enhet fra databasen (MySQL).

---

<sup>20</sup> Programmering i Java, Else Lervik, Vegard B. Havdal, s. 905

<sup>21</sup> <http://stackoverflow.com/questions/5495534/java-net-connectexception-localhost-127-0-0-18080-connection-refused>

<sup>22</sup> Klassen `ForelderServlet` kan du se i Vedlegg 3, s. 30

//koden under er POST metoden for registrering av enhet på server

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    String registrationID = hentParametere(request, PARAMETER_REGISTRASION_ID);
    String enhetsNavn = hentParametere(request, PARAMETER_ENHETSNAVN);
    String latitude = hentParametere(request, PARAMETER_LATITUDE);
    String longitude = hentParametere(request, PARAMETER_LONGITUDE);
    //registrer enhet med oversendte parametere
    Databehandling.registrerEnhet(registrationID, enhetsNavn, latitude, longitude);
    setSuksess(response);
    try {
        //skal registrert enhets posisjon sendes?
        if (!latitude.equals(IKKE_SEND_POSISJON) && !longitude.equals(IKKE_SEND_POSISJON)) {
            //legg ved ekstra attributter for utsending av koordinater
            request.setAttribute(MELDINGSTYPE, SEND_KOORDINATER_MELDING);
            request.setAttribute(MELDINGS_EMNE, SEND_KOORDINATER_MELDING);
            request.getRequestDispatcher("SendMeldinger").forward(request, response);
        } //if (!latitude.equals(IKKE_SEND_POSISJON) && !longitude.equals(IKKE_SEND_POSISJON))
    } catch (IOException ex) {
        ex.printStackTrace();
    } //try/catch
} //doPost
```

## Mottak og sending av melding

Server kan motta to forskjellige typer meldinger. Den ene er mottak av koordinater, den andre er at enhet har gått offline (*avsluttet applikasjon, men fortsetter å motta posisjonsdata*). Som nevnt i forrige kapittel, så mottas koordinater ved registrering. Deretter når enhet sender ut sine posisjonsdata, blir disse sendt til en servlet kalt MottaMeldingFraEnhet. Denne servleten tjener to oppgaver. Det ene er videresending av mottatte koordinater, og den andre er å videresende melding om at enhet gikk offline. Måten dette sjekkes på er ved å se på hvilke parametere som ble sendt fra enheten.

//hent ut parameter for om enhet gikk offline

```
String enhetOffline = hentParametere(request, ENHET_GIKK_OFFLINE, "");
String registrationID = hentParametere(request, PARAMETER_REGISTRASION_ID);
if (enhetOffline.equals("")) {
    //hent ut oversendte koordinater
    String enhetsNavn = hentParametere(request, PARAMETER_ENHETSNAVN);
    String latitude = hentParametere(request, PARAMETER_LATITUDE);
    String longitude = hentParametere(request, PARAMETER_LONGITUDE);
    //oppdater enhets lokasjon
    Databehandling.oppdaterKoordinater(registrationID, enhetsNavn, latitude, longitude);
    videresendKoordinater(request, response, latitude, longitude);
} else {
    //legg ved ekstra attributter for videresending
    request.setAttribute(MELDINGSTYPE, ENHET_GIKK_OFFLINE);
    request.setAttribute(MELDINGS_EMNE, ENHET_GIKK_OFFLINE);
    Databehandling.setEnhetSomOffline(registrationID);
    request.getRequestDispatcher("SendMeldinger").forward(request, response);
} //if
```

Videresending av meldinger til enheter blir gjort ved hjelp av Servleten kalt SendMeldinger<sup>23</sup>. Denne servleten er basert på eksempelkoden fra gcm-demo-server og Avilyne<sup>24</sup> (som har et ganske kjekt tutorial). For å finne ut hva slags melding som skal sendes, og hva meldingen inneholder, så henter jeg ut parameterne i doPost på SendMeldinger. Server mottar som nevnt kun to forskjellige meldinger fra enhet, koordinater eller offline melding. Foruten utsending av disse, så sender også server ut to andre typer meldinger.

Den ene typen melding som sendes ut, er melding fra innlogget administrator på serverens nettside:

```
private void opprettMeldingTilEnheter(String emne, String melding) {
    //Objektet som tar vare på dataene som skal sendes
    message = new Message.Builder()
    //hvis flere meldinger sendes med samme tittel/emne til enheter, og hvis
    //enheten var offline når tidligere meldinger ble sendt, så sørger
    //collapseKey(...) for at brukeren kun mottar den siste meldingen
    //med gitt tittel
    .collapseKey(emne)
    .timeToLive(30)
    .delayWhileIdle(true)
    .addData(SEND_MELDING_TIL_ENHETER, melding)
    .build();
    sendMeldingerTilEnhet();
} //opprettMeldingTilEnheter
```

Den andre typen er en melding som forteller valgte enheter at administrator har frakoblet de fra server. M.a.o. fjernet deres tilgang til server. Et litt kinkig problem som oppsto når jeg jobbet med frakobling av enheter var utsending til korrekt enhet om at denne enheten var blitt frakoblet server. Istedenfor å sende meldingen til frakoblet enhet, fikk alle tilkoblede enheter meldingen "Du er frakoblet".

Litt av problemet lå nok i det at jeg ønsket å ikke sende administrator vekk fra siden for frakobling<sup>25</sup>, så jeg kalte istedenfor direkte på funksjoner fra SendMeldinger. Det som skapte problemet var faktisk det at jeg hadde en global variabel for sending av meldinger, Messages. Når jeg lagde en lokal variabel inni funksjonen, så ble frakoblet melding sendt kun til enhet(er) som var frakoblet.

```
//denne kodesnutten er tatt fra klassen Databehandling
public static void kobleFraEnheter(String[] valgteEnheter) {
    //melding som skal vises til bruker
    String melding = "Du har blitt frakoblet TracknHide's server!";
    SendMeldinger sendMelding = new SendMeldinger();
    for (int i = 0; i < valgteEnheter.length; i++) {
        //send melding til enheter at om valgte enheter har gått offline
        sendMelding.opprettEnhetGikkOffline(valgteEnheter[i]);
        //send melding som forteller enheten at den har blitt frakoblet
        sendMelding.sendKobletFraServerMelding(valgteEnheter[i], melding);
        avRegistrerEnhet(valgteEnheter[i]);
    } //for
} //kobleFraEnheter
```

<sup>23</sup> Klassen SendMeldinger kan du se i Vedlegg 4, s. 31

<sup>24</sup> <http://avilyne.com/?p=267>

<sup>25</sup> Jeg kunne her ha brukt POST, men valgte å løse dette ved bruk av funksjoner som jobbet "i bakgrunnen"

```
//denne kodesnutten er fra servleten SendMeldinger
protected void sendKobletFraServerMelding(String regID, String melding) {
    try {
        //Oppretter eget objektet her siden melding kun skal sendes til en bruker
        Message message = new Message.Builder()
            .timeToLive(30)
            .delayWhileIdle(true)
            .addData(SEND_MELDING_TIL_ENHETER, melding)
            .build();
        //oppretter arraylist og legger ved regID
        ArrayList<String> utsendingsListe = new ArrayList<String>();
        utsendingsListe.add(regID);
        //Hent resultatet av masseutsendingen via GCM
        sender.send(message, utsendingsListe, 1);
    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (Exception ex) {
        ex.printStackTrace();
    } //try/catch
} //sendKobletFraServerMelding
```

En av forskjellene på min måte å sende ut meldinger på og eksemplet i samples, er at jeg bruker egen trådklasse for sending av meldinger. Jeg prøvde å sette meg inn i Executor, men fant det enklere å heller forholde meg til ting jeg kunne. Jeg baserte derfor grunnlaget til trådklassen på et eksempel fra en lærebok om Java programmering<sup>26</sup>.

Dette er også grunnen til at jeg tok i bruk synchronized på funksjonen som tar for seg den faktiske utsendingen av meldinger. Siden det er ukjent når en tråd faktisk starter og fullføres, og hvor ofte bruker sender sine posisjonsdata, kunne en risikere at enheter mottok koordinater i feil rekkefølge, noe som ville ført til alt annet enn korrekt tegning på kartet.

### *Nettsidene og hva administrator får se*

Brukernavn: Lucas

Passord: pwd

Administrator får tilgang til fire forskjellige sider. Nettsidene er lagt opp slik at administrator må være innlogget for å få tilgang til siden. Selv om en bruker uten tilgang finner linker i nettsiden, så vil bruker bare bli videresendt til index.jsp som ber bruker oppgi brukernavn og passord. For å holde bruker innlogget, så brukes HttpSession<sup>27</sup>. Dersom session objektet er tomt eller null, så vil bruker bli bedt om å logge inn, og etter 5 minutt med inaktivitet blir bruker automatisk logget ut og session objektet invalidert/ødelagt. Jeg valgte 5 minutt, siden dette verken er en side mye aktivitet eller oppgaver, og ser heller ikke behovet for at en administrator skal ha "evig" tilgang.

<sup>26</sup> Programmering i Java, Else Lervik, Vegard B. Havdal, eksempel på s. 695

<sup>27</sup> Programmering i Java, Else Lervik, Vegard B. Havdal, s. 909

<http://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpSession.html>

Nettsidene som administrator får se er:

- **index.jsp**  
Inneholder en velkomsttekst med innlogget brukers navn, kort beskrivelse av nettsiden og hva den er til
- **visonlinenheter.jsp**  
Viser enheter som er tilkoblet og deres siste registrerte posisjon
- **sendmelding.jsp**  
Gir administrator mulighet til å sende en melding til alle tilkoblede enheter
- **koblefraenhet.jsp**  
Gir administrator mulighet til å koble fra en eller flere tilkoblede enheter fra server

Det eneste jeg kan si om fargevalget foruten at de matcher basert på fargehjulet<sup>28</sup>, er egentlig at jeg liker svart og lilla.

### *MySQL og databasens innhold*

Det er ikke så mye å si om databasen, da det kun er to tabeller, og navnene er ganske selvforutsigbare. Tankene mine rundt tblAdmin var ganske enkelt at man har en eller flere administratorer med brukernavn og passord. I tillegg la jeg til et felt for adgangsnivå, med tanke på fremtidig utvikling. F.eks. skille ut administrator og moderator dersom det skulle være behov for direkte tilgang til databasen eller andre større operasjoner. Adgangsnivå er ikke blitt brukt i denne oppgaven, foruten at det ligger i databasen.

Da jeg utviklet tblKlienter trodde jeg at registrationID var unik, men som nevnt tidligere var dette en misoppfattelse fra min side. For å gjøre tilkoblede enheter litt mer identifiserbare, så valgte jeg å legge til et felt for enhetens navn og datoen enhet koblet seg til server første gang. Grunnen til at jeg tok med dato var tanken at om en enhet blir avregistrert, men melding om dette ikke blir sendt til server, så kan administrator frakoble enhet (*dersom tilkoblingsdato er langt tilbake i tid*).

Et forbedringspotensial for tblAdmin (*og nettsidene*) kunne vært å ha lagt til mulighet for endring av passord via nettsidene. I tillegg så burde administrators passord vært hashet og saltet i databasen. Dette var noe jeg ikke valgte å prioritere (*pga resterende tid*), men nevner det her i rapporten for å vise at jeg er klar over dette.

---

<sup>28</sup> Med fargehjulet tenker jeg da på grafisk brukergrensesnitt og farger som går godt sammen

# Oppsummering/konklusjon

---

Dette har vært et interessant, men og ganske slitsomt prosjekt. Jeg har lært mye nytt, bl.a. bruk av flere tabeller i SQLite, at AsyncTask ikke kaster Exceptions (*i den forstand man ikke får vite at oppgaven feilet*) og jeg har fått prøvd meg på å utvikle server ved å bruke Servlets. For ikke glemme at jeg har aldri brukt Google Maps, C2DM eller GCM.

Det har vært ting som har tatt altfor mye tid, og ting som gikk raskere enn jeg selv hadde forutsett. Det mest strevsomme var uten tvil det faktum at jeg ikke kunne gjøre for store "hopp" under utviklingen. Det jeg mener er at jeg erfarte altfor ofte de gangene jeg skrev mye kode og la inn mye funksjonalitet at ting feilet. Dette førte til tider at jeg måtte ta backup av koden og trykke mye "Ctrl + Z". Kanskje det er kun meg, men det er i hvert fall en følelse jeg sitter igjen med når jeg sammenligner mot andre større utviklingsprosjekt innenfor andre felt.

Jeg kan hvert fall si at jeg har fått god bruk for kunnskapen jeg har tilegnet meg gjennom dette semesteret. Jeg hadde ikke hatt sjanse til å fullføre denne oppgaven uten lærdommen fra de fire obligatoriske oppgavene. Det har vært en del tilbakeblikk på tidligere kode for å se hvordan jeg gjorde det da, og for å friske opp minnet for å bedre funksjonaliteten. Og som lærere jeg hadde fra før sa: "Ingen vits i å finne opp hjulet på nytt" og "If it works, don't fix it!".

Jeg må nå ærlig innrømme at jeg har lite lyst til å starte et nytt utviklingsprosjekt når det kommer til Google Maps og kart på Android, men eventuelt å fortsette med applikasjonsutvikling for Android?

Det har jeg lyst til!

Knut Lucas Andersen

14.05.2013 - Narvik

# Kilder og litteraturliste

---

## Android

Kilde	Gjelder	Funnet på side
Internett	Koble til nettverk (internett)	<a href="http://developer.android.com/training/basics/network-ops/connecting.html">http://developer.android.com/training/basics/network-ops/connecting.html</a>
Internett	Notifications	<a href="http://developer.android.com/guide/topics/ui/notifiers/notifications.html">http://developer.android.com/guide/topics/ui/notifiers/notifications.html</a>
Internett	Rett API i emulator og properties	<a href="http://stackoverflow.com/questions/11806937/java-the-import-com-google-cannot-be-resolved">http://stackoverflow.com/questions/11806937/java-the-import-com-google-cannot-be-resolved</a>
Internett	Endre farge på titteltekst i AlertDialog	<a href="http://stackoverflow.com/questions/14439538/how-can-i-change-the-color-of-alertdialog-title-and-the-color-of-the-line-under">http://stackoverflow.com/questions/14439538/how-can-i-change-the-color-of-alertdialog-title-and-the-color-of-the-line-under</a>
Internett	Datatyper SQLite	<a href="http://www.sqlite.org/datatype3.html">http://www.sqlite.org/datatype3.html</a>
Internett	Maks lengde for tekst i SQLite	<a href="http://sqlite.org/limits.html#max_length">http://sqlite.org/limits.html#max_length</a>
Internett	Google Play på ekte enhet vs. emulator	<a href="http://stackoverflow.com/questions/11154222/google-play-on-android-4-0-emulator">http://stackoverflow.com/questions/11154222/google-play-on-android-4-0-emulator</a>
Internett	Side for oppsett av api for bruk av Maps og GCM	<a href="https://code.google.com/apis/console/">https://code.google.com/apis/console/</a> <a href="https://developers.google.com/console/help/">https://developers.google.com/console/help/</a>
Internett	Ikoner brukt i applikasjon	<a href="http://developer.android.com/design/downloads/index.html">http://developer.android.com/design/downloads/index.html</a>

## Exceptions/Errors

Kilde	Gjelder	Funnet på side
Internett	Localhost i emulator	<a href="http://stackoverflow.com/questions/5495534/java-net-connectexception-localhost-127-0-0-18080-connection-refused">http://stackoverflow.com/questions/5495534/java-net-connectexception-localhost-127-0-0-18080-connection-refused</a>
Internett	Datamanipulasjon MySQL	<a href="http://stackoverflow.com/questions/1905607/can-not-issue-data-manipulation-statements-with-executequery">http://stackoverflow.com/questions/1905607/can-not-issue-data-manipulation-statements-with-executequery</a>
Internett	doInBackground() Exception	<a href="http://stackoverflow.com/questions/8597088/async-task-never-executing-doinbackground-but-does-in-debug-update-fixed">http://stackoverflow.com/questions/8597088/async-task-never-executing-doinbackground-but-does-in-debug-update-fixed</a>

## Google Cloud Messaging (GCM)

Kilde	Gjelder	Funnet på side
Internett	Om GCM	<a href="http://developer.android.com/google/gcm/index.html">http://developer.android.com/google/gcm/index.html</a>
	RegistrationID	<a href="http://developer.android.com/google/gcm/gcm.html#arch">http://developer.android.com/google/gcm/gcm.html#arch</a>
	GCM packages	<a href="http://developer.android.com/reference/gcm-packages.html">http://developer.android.com/reference/gcm-packages.html</a>
Internett	Maks lengde på registrationID	<a href="http://stackoverflow.com/questions/11668761/gcm-max-length-for-registration-id">http://stackoverflow.com/questions/11668761/gcm-max-length-for-registration-id</a>
Internett	Krasj ved bruk av onDestroy() og GCM avregistrering	<a href="http://stackoverflow.com/questions/11935680/gcmregistrar-ondestroycontext-crashing-receiver-not-registered">http://stackoverflow.com/questions/11935680/gcmregistrar-ondestroycontext-crashing-receiver-not-registered</a>



## Google Maps

Kilde	Gjelder	Funnet på side
Internett	Google Maps API lvl2 i emulator	<a href="http://nemanjakovacevic.net/blog/2012/12/how-to-make-android-google-maps-v2-work-in-android-emulator/">http://nemanjakovacevic.net/blog/2012/12/how-to-make-android-google-maps-v2-work-in-android-emulator/</a>  <a href="http://stackoverflow.com/questions/14445093/how-to-show-google-map-v2-on-android-emulator">http://stackoverflow.com/questions/14445093/how-to-show-google-map-v2-on-android-emulator</a>
Internett	Google Maps API lvl1 deprecated	<a href="https://developers.google.com/maps/documentation/android/v1/mapkey">https://developers.google.com/maps/documentation/android/v1/mapkey</a>
Internett	Google Maps API lvl2	<a href="https://developers.google.com/maps/documentation/android/index">https://developers.google.com/maps/documentation/android/index</a>  <a href="https://developers.google.com/maps/documentation/android/start">https://developers.google.com/maps/documentation/android/start</a>
Internett	Ta i bruk Google Maps Video nr. 129 - 144	<a href="http://thenewboston.org/list.php?cat=6">http://thenewboston.org/list.php?cat=6</a>
Internett	Debug.keystore	<a href="http://stackoverflow.com/questions/7296467/google-map-signed-api-key-errors-in-android">http://stackoverflow.com/questions/7296467/google-map-signed-api-key-errors-in-android</a>

## Server

Kilde	Gjelder	Funnet på side
Internett	Selekterte enheter for frakobling	<a href="http://www.roseindia.net/jsp/jsp-checkbox.shtml">http://www.roseindia.net/jsp/jsp-checkbox.shtml</a>
Internett	Utsending av meldinger fra server	<a href="http://avilyne.com/?p=267">http://avilyne.com/?p=267</a>  <a href="http://developer.android.com/reference/com/google/android/gcm/Server/Result.html">http://developer.android.com/reference/com/google/android/gcm/Server/Result.html</a>
Internett	HttpSession getSession	<a href="http://www.javaken.com/forum/showthread.php?t=2773">http://www.javaken.com/forum/showthread.php?t=2773</a>
Internett	HttpSession dokumentasjon	<a href="http://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpSession.html">http://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpSession.html</a>
Internett	TomCat	<a href="http://tomcat.apache.org/">http://tomcat.apache.org/</a>
Internett	Eclipse	<a href="http://www.eclipse.org/">http://www.eclipse.org/</a>

## Litteraturliste

Kilde	Gjelder	Funnet på side
Programmering i Java, Else Lervik, Vegard B. Havdal, tisip - Gyldendal, 2009, ISBN 978-82-05-39050-8	(Kap. 20) Tråder (Kap. 24) Databaser og MySQL (Kap. 25) Servlet og JavaServer Pages (JSP) (Vedl. F) Bruk av jar og JavaDoc	s. 685 s. 817 s. 861 s. 928 - 932
Professional Android 4 Application Development, Reto Meier, Wrox Wiley, 2012, ISBN978-1-118-10227-5	(Kap. 8) SQLite, Databaser og ContentProvider (Kap. 13) Kart o.l. (Google Maps) (Kap. 18) Google Cloud Messaging	s. 251 s. 513 s. 743
Forelesningsnotater, Werner Farstad, 2013	Kap13 - PosisjoneringOgKart-1.pdf Kap13 - PosisjoneringOgKart-3.pdf servletEclipse.pdf servletSesjoner.pdf Google Cloud Messaging.pdf	s. 4, 6 s. 4-7, 9 s. 1-6 s. 5, 12 s. 3-4

# Vedlegg 1: AndroidManifest.xml

---

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="dt.hin.android.kl_andersen"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <!-- Krav om tilgang til å sjekke om nettverk er oppe ved oppstart av applikasjon -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <!-- Krav om internett tilgang siden kart lastes ned herfra -->
    <uses-permission android:name="android.permission.INTERNET" />
    <!-- Tilgang til brukers posisjon for å vise lokasjon på kart -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <!-- GCM krever en Google konto -->
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <!-- Unngå at enhet går i hvilemodus mens meldinger/hendelser mottas -->
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <!-- Tilgang til å registrere at enhet er slått på -->
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <!--
    Egendefinert tillatelse som gjør at applikasjonen kan motta meldinger fra GCM.
    Her må pakkenavnet til applikasjonen være med!
    Eksempel: PACKAGE.permission.C2D_MESSAGE.
    -->
    <permission
        android:name="dt.hin.android.kl_andersen.permission.C2D_MESSAGE"
        android:protectionLevel="signature" />

    <uses-permission android:name="dt.hin.android.kl_andersen.permission.C2D_MESSAGE" />
    <!-- Denne applikasjonen har tillatelse til å registrere og motta meldinger -->
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <uses-library android:name="com.google.android.maps" />

        <activity
            android:name="dt.hin.android.kl_andersen.MainMapActivity"
            android:label="@string/app_name"
            android:uiOptions="splitActionBarWhenNarrow" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- Sub-aktivitet som fremviser systemlogg og mulighet for å slette denne -->
        <activity
            android:name="dt.hin.android.kl_andersen.SystemloggActivity"
            android:label="@string/activity_systemlogg" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="dt.hin.android.kl_andersen.MainMapActivity" />
        </activity>
    </application>
</manifest>
```

```

        android:value="dt.hin.android.kl_andersen.MainMapActivity" />
    </activity>

    <!-- Sub-aktivitet som fremviser lagrede ruter -->
    <activity
        android:name="dt.hin.android.kl_andersen.VisRuterActivity"
        android:label="@string/activity_visruter" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="dt.hin.android.kl_andersen.MainMapActivity" />
    </activity>

    <!-- Sub-aktivitet som fremviser innstillinger -->
    <activity
        android:name="dt.hin.android.kl_andersen.InnstillingsActivity"
        android:label="@string/activity_innstillinger" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="dt.hin.android.kl_andersen.MainMapActivity" />
    </activity>

    <!-- ContentProvider -->
    <provider
        android:name="dt.hin.android.kl_andersen.LoggingContentProvider"
        android:authorities="dt.android.rutelogging"
        android:exported="false" />

    <!-- BroadcastReceiver som mottar applikasjonshendelser og -meldinger -->
    <receiver
        android:name=".KringkastningsMottaker"
        android:enabled="true"
        android:exported="false" >
        <intent-filter>
            <action android:name="dt.hin.android.kl_andersen.INSTALLINGER_ACTIVITY" />
            <action android:name="dt.hin.android.kl_andersen.VIS_MELDING" />
            <action android:name="android.intent.action.BOOT_COMPLETED" />
        </intent-filter>
    </receiver>

    <!--
    Kringkastningsmottaker som mottar intents fra GCM services og sender det til IntentService for behandling.
    En må ha tillatelsen com.google.android.c2dm.permission.SEND slik at kun GCM service kan sende meldinger for
    applikasjonen.
    -->
    <receiver
        android:name="com.google.android.gcm.GCMBroadcastReceiver"
        android:permission="com.google.android.c2dm.permission.SEND" >
        <intent-filter>

            <!-- Mottar meldingene -->
            <action android:name="com.google.android.c2dm.intent.RECEIVE" />
            <!-- Mottar registrationID -->
            <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
            <!-- Kategori -->
            <category android:name="dt.hin.android.kl_andersen" />
        </intent-filter>
    </receiver>

    <!--
    Applikasjonsspesifikk sub-klasse av GCMBaseIntentService som håndterer mottak av meldinger.
    Service må navngis som .GCMIntentService, såfremt ikke applikasjon bruker en egendefinert
    BroadcastReceiver som redefinerer navnet.
    -->
    <service android:name=".GCMIntentService" />
</application>
</manifest>

```

# Vedlegg 2: Android - Klassen Enhet

---

```
public class Enhet {
    /** Datoformatet som rutetidspunkt blir lagret som i databasen (SQLite) */
    private final String DATO_TID_FORMAT = "yyyy-MM-dd HH:mm:ss";
    private long _identifikator;
    private String _registrationID;
    private String _enhetsNavn;
    private boolean _egenEnhet;
    private int _ruteID;
    private String _ruteTidspunkt;
    private ArrayList<GeoPoint> _koordinatListe;

    /**
     * Konstruktør for oppretting av en enhet. */
    public Enhet(long identifikator, String registrationID, String enhetsNavn, boolean egenEnhet, int ruteID,
        ArrayList<GeoPoint> koordinatListe) {
        _identifikator = identifikator;
        _registrationID = registrationID;
        _enhetsNavn = enhetsNavn;
        _egenEnhet = egenEnhet;
        _ruteID = ruteID;
        _koordinatListe = new ArrayList<GeoPoint>(koordinatListe);
    } //konstruktør

    /**
     * Konstruktør for oppretting av enhet. <br />
     * Bruksområde er hovedsaklig tenkt for fremvisning/opplisting og generering av lagrede ruter fra databasen */
    public Enhet(long identifikator, String registrationID, String enhetsNavn, boolean egenEnhet, int ruteID, String
        ruteTidspunkt, ArrayList<GeoPoint> koordinatListe) {
        _identifikator = identifikator;
        _registrationID = registrationID;
        _enhetsNavn = enhetsNavn;
        _egenEnhet = egenEnhet;
        _ruteID = ruteID;
        _ruteTidspunkt = ruteTidspunkt;
        _koordinatListe = new ArrayList<GeoPoint>(koordinatListe);
    } //konstruktør

    public long getIdentifikator() {
        return _identifikator;
    } //getIdentifikator

    public String getRegistrationID() {
        return _registrationID;
    } //getRegistrationID

    public String getEnhetsnavn() {
        return _enhetsNavn;
    } //getEnhetsnavn

    public boolean getEgenEnhet() {
        return _egenEnhet;
    } //getEgenEnhet

    public int getRuteID() {
        return _ruteID;
    } //getRuteID

    /**
     * Kun tilgjengelig ved oppretting av ruteliste og ruter basert på lagrede data.
```

```

    * @return String: Tidspunkt da ruten ble opprettet
    */
    public String getRuteTidspunkt() {
        return _ruteTidspunkt;
    } //getRuteTidspunkt

    public ArrayList<GeoPoint> getKoordinater() {
        return _koordinatListe;
    } //getKoordinater

    public void setEnhetsNavn(String enhetsNavn) {
        _enhetsNavn = enhetsNavn;
    } //setEnhetsNavn

    /**
     * Brukes ved tegning av lagrede ruter for å unngå at eksisterende rute ikke blir tegnet på kartet. <br />
     * Denne setter egenEnhet til false, likegyldig hva tidligere verdi var.
     */
    public void setEgenEnhetFalse() {
        _egenEnhet = false;
    } //setEgenEnhetFalse

    /**
     * Registrerer/legger til denne enhetens siste mottatte koordinat
     * @param koordinat - GeoPoint: Siste mottatte koordinat
     */
    public void addGeoPoint(GeoPoint koordinat) {
        _koordinatListe.add(koordinat);
    } //addGeoPoint

    @Override
    public String toString() {
        String utskrift = "";
        try {
            //konverter eksisterende tidspunkt til dato
            Locale local = Locale.getDefault();
            SimpleDateFormat formatering = new SimpleDateFormat(DATO_TID_FORMAT, local);
            Date tidspunkt = (Date) formatering.parse(getRuteTidspunkt());
            //omgjør dato til kalender
            Calendar kalender = Calendar.getInstance();
            kalender.setTime(tidspunkt);
            //konverter dato-strengen ved å bruke calendar (09/05-2013 18:00:00)
            String tid = String.format("%02d", kalender.get(Calendar.DATE))
                + "/" + String.format("%02d", (kalender.get(Calendar.MONTH) + 1))
                + "-" + kalender.get(Calendar.YEAR) + " "
                + String.format("%02d", kalender.get(Calendar.HOUR))
                + ":" + String.format("%02d", kalender.get(Calendar.MINUTE))
                + ":" + String.format("%02d", kalender.get(Calendar.SECOND));

            //opprett utskriften og returner den
            utskrift = "Rute registrert: " + tid + "\n" + "Enhetsnavn: " + getEnhetsnavn() + ", ";
            if (getEgenEnhet()) {
                utskrift += "Egen rute";
            } else {
                utskrift += "Annen brukers rute";
            } //if (_egenEnhet)
        } catch (ParseException ex) {
            ex.printStackTrace();
        } catch (Exception ex) {
            ex.printStackTrace();
        } //try/catch
        return utskrift;
    } //toString
} //Enhet

```

# Vedlegg 3: Server - ForelderServlet

---

**abstract class** ForelderServlet **extends** HttpServlet {

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    super.doGet(request, response);
} //doGet
```

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    super.doPost(request, response);
} //doPost
```

```
/**
 * Henter ut parameterverdien fra oversendt HttpServletRequest. */
protected String hentParametere(HttpServletRequest request, String parameter) throws ServletException {
    String verdi = request.getParameter(parameter);
    //inneholder parameteren verdi?
    if (erTomEllerNull(verdi)) {
        throw new ServletException("Mottatt parameter " + parameter + " ikke funnet!");
    } //if (erTomEllerNull(verdi))
    return verdi.trim();
} //hentParametere
```

```
/**
 * Henter ut parameterverdien fra oversendt HttpServletRequest. <br />
 * Dersom parameter som verdi skal hentes ut fra ikke har verdi (null/blank),
 * returneres oversendt defaultVerdi. */
protected String hentParametere(HttpServletRequest request, String parameter, String defaultVerdi) {
    String verdi = request.getParameter(parameter);
    if (erTomEllerNull(verdi)) {
        verdi = defaultVerdi;
    } //if (erTomEllerNull(verdi))
    return verdi.trim();
} //hentParametere
```

```
protected void setSuksess(HttpServletResponse response) {
    setSuksess(response, 0);
} //setSuksess
```

```
protected void setSuksess(HttpServletResponse response, int size) {
    response.setStatus(HttpServletResponse.SC_OK);
    response.setContentType("text/plain");
    response.setContentLength(size);
} //setSuksess
```

```
/**
 * Funksjon som sjekker om oversendt string har innhold. <br />
 * Det sjekkes på om string er null og om den inneholder whitespace (""). <br />
 * Dersom string har verdi returneres false, hvis string er tom returneres true. */
protected boolean erTomEllerNull(String verdi) {
    //er verdi null eller inneholder den kun space/intet?
    if (verdi == null || verdi.trim().length() == 0) {
        return true;
    } //if (verdi == null || verdi.trim().length() == 0)
    return false;
} //erTomEllerNull
```

```
} //ForelderServlet
```

# Vedlegg 4: Server - SendMeldinger

---

```
public class SendMeldinger extends ForelderServlet {
    /** Hvor mange meldinger som kan sendes på en gang via GCM */
    private static final int MAX_ANTALL_UTSENDINGER = 1000;
    private ArrayList<String> onlineEnheter;
    private Message message;
    //Instanse av com.android.gcm.server.Sender som tar for seg sending
    //av meldingen til GCM service
    private Sender sender;
    private boolean meldingSendesFraWeb = false;

    public SendMeldinger() {
        super();
        sender = new Sender(API_PROJECT_KEY);
    } //konstruktør

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException {
        String lat = "",
            lon = "",
            regID = "",
            emne = "",
            melding = "",
            enhetsNavn = "",
            utsendelsesType = "";

        try {
            meldingSendesFraWeb = false;
            onlineEnheter = Databehandling.hentAlleEnhetersRegID();
            //forsøk å hente ut utsendelsesType - hvis tom,
            //sjekk om det er oversendt som attributt
            utsendelsesType = hentParametere(request, MELDINGSTYPE, "");
            if (utsendelsesType.equals("")) {
                utsendelsesType = (String) request.getAttribute(MELDINGSTYPE);
            } //if (utsendelsesType.equals("")) {
            //er det utsending av koordinater? isåfall, hent verdiene
            if (utsendelsesType.equals(SEND_KOORDINATER_MELDING)) {
                emne = (String) request.getAttribute(MELDINGS_EMNE);
                regID = hentParametere(request, PARAMETER_REGISTRATION_ID);
                enhetsNavn = hentParametere(request, PARAMETER_ENHETSNAMN);
                lat = hentParametere(request, PARAMETER_LATITUDE);
                lon = hentParametere(request, PARAMETER_LONGITUDE);
                opprettKoordinatMelding(emne, regID, enhetsNavn, lat, lon);
            } else if (utsendelsesType.equals(ENHET_GIKK_OFFLINE)) {
                emne = (String) request.getAttribute(MELDINGS_EMNE);
                regID = hentParametere(request, PARAMETER_REGISTRATION_ID);
                opprettEnhetGikkOffline(regID);
            } else if (utsendelsesType.equals(SEND_MELDING_TIL_ENHETER)) {
                //vanlig melding som skal sendes ut
                meldingSendesFraWeb = true;
                melding = hentParametere(request, MELDINGS_INNHOLD);
                emne = hentParametere(request, MELDINGS_EMNE);
                opprettMeldingTilEnheter(emne, melding);
                //sender over verdiene slik at de fortsatt vises i meldingsfeltet
                request.setAttribute(MELDINGS_EMNE, emne);
                request.setAttribute(MELDINGS_INNHOLD, melding);
                //videresend bruker tilbake til sendmelding.jsp
                request.getRequestDispatcher("sendmelding.jsp").forward(request, response);
            } //if (utsendelsesType.equals(SEND_KOORDINATER_MELDING))
        }
    }
}
```

```

    } catch (Exception ex) {
        ex.printStackTrace();
    } //try/catch
} //doPost

/**
 * Oppretter melding som inneholder koordinatene til en enhet. <br />
 * Alle enheter som er online (foruten sender av koordinatene) vil motta
 * meldingen. */
private void opprettKoordinatMelding(String emne, String regID, String enhetsNavn, String lat, String lon) {
    //fjern enhet som sendte koordinatene, vil kun sende til andre påloggede enheter
    onlineEnheter.remove(regID);
    if (onlineEnheter.size() > 0) {
        //Objektet som tar vare på dataene som skal sendes
        message = new Message.Builder()
            .timeToLive(30)
            .delayWhileIdle(true)
            .addData(PARAMETER_REGISTRATION_ID, regID)
            .addData(PARAMETER_ENHETSNAVN, enhetsNavn)
            .addData(PARAMETER_LATITUDE, lat)
            .addData(PARAMETER_LONGITUDE, lon)
            .build();
        sendMeldingerTilEnhet();
    } //if (onlineEnheter.size() > 0)
} //opprettKoordinatMelding

/**
 * Oppretter en melding om at en enhet gikk offline (koblet seg fra server). <br />
 * Meldingen blir sendt til alle enheter som fortsatt er online. */
protected void opprettEnhetGikkOffline(String regID) {
    //kaller på denne her i tilfelle denne funksjonen blir kalt
    //pga. enhet(er) har blitt frakoblet server av administrator
    onlineEnheter = Databehandling.hentAlleEnhetersRegID();
    //sørger for at enhet ikke mottar offline melding
    //(i tilfelle forsinkelse under oppdatering av av faktiske online enheter)
    onlineEnheter.remove(regID);
    //er det fortsatt enheter online?
    if (onlineEnheter.size() > 0) {
        //Objektet som tar vare på dataene som skal sendes
        message = new Message.Builder()
            .timeToLive(30)
            .delayWhileIdle(true)
            .addData(ENHET_GIKK_OFFLINE, regID)
            .build();
        sendMeldingerTilEnhet();
    } //if (onlineEnheter.size() > 0)
} //opprettEnhetGikkOffline

```



```

/**
 * Oppretter en melding som sendes ut til enhet som ble frakoblet server. */
protected void sendKobletFraServerMelding(String regID, String melding) {
    try {
        //Oppretter eget objektet her siden melding kun skal sendes til en bruker
        Message message = new Message.Builder()
            .timeToLive(30)
            .delayWhileIdle(true)
            .addData(SEND_MELDING_TIL_ENHETER, melding)
            .build();
        //opprettet arraylist og legger ved regID
        ArrayList<String> utsendingsListe = new ArrayList<String>();
        utsendingsListe.add(regID);
        //Hent resultatet av masseutsendingen via GCM
        sender.send(message, utsendingsListe, 1);
    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (Exception ex) {
        ex.printStackTrace();
    } //try/catch
} //sendKobletFraServerMelding

/**
 * Oppretter en melding som sendes til enhetene. <br />
 * Dette er en melding som kan leses på enhet. */
private void opprettMeldingTilEnheter(String emne, String melding) {
    //Objektet som tar vare på dataene som skal sendes
    message = new Message.Builder()
        //hvis flere meldinger sendes med samme tittel/emne til enheter, og hvis
        //enheten var offline når tidligere meldinger ble sendt, så sørger
        //collapseKey(...) for at brukeren kun mottar den siste meldingen
        //med gitt tittel
        .collapseKey(emne)
        .timeToLive(30)
        .delayWhileIdle(true)
        .addData(SEND_MELDING_TIL_ENHETER, melding)
        .build();
    sendMeldingerTilEnhet();
} //opprettMeldingTilEnheter

/**
 * Sender meldinger til enhet. <br />
 * Dersom antallet enheter online er større enn grensen for hva GCM kan håndtere (pr. dags dato 1000 enheter),
 * <br /> blir meldingsutsendelsen splittet opp. <br />
 * Melding sendes først til de første 1000, deretter de resterende. <br />
 * Melding sendes ut ved bruk av trådklassen MultiUtsendingAvMeldinger. */
private void sendMeldingerTilEnhet() {
    //skal det sendes ut melding til flere enn 1000 enheter?
    if (onlineEnheter.size() > MAX_ANTALL_UTSENDINGER) {
        //hent ut totalt antall påloggede enheter
        int antallOnline = onlineEnheter.size();
        //opprett tom arraylist med plass til alle enheter som er online
        ArrayList<String> utsendingsListe = new ArrayList<String>(antallOnline);
        int teller = 0,
            antLagtTil = 0;
        for (String enhet : onlineEnheter) {
            //er 1000 enheter lagt til i arraylist/er alle enheter lagt til for utsending
            if (antLagtTil == MAX_ANTALL_UTSENDINGER || teller == antallOnline) {
                //opprett tråd som tar for seg utsending av melding til enhetene
                MultiUtsendingAvMeldinger traad = new
                MultiUtsendingAvMeldinger(utsendingsListe);
                traad.start();
            } //if (antLagtTil == MAX_ANTALL_UTSENDINGER || teller == antallOnline)

```

```

        utsendingsListe.add(enhet);
        antLagtTil = utsendingsListe.size();
        teller++;
    } //for
} else {
    //opprett tråd som tar for seg utsending av melding til enhetene
    MultiUtsendingAvMeldinger traad = new MultiUtsendingAvMeldinger(onlineEnheter);
    traad.start();
} //if (onlineEnheter.size() > MAX_ANTALL_UTSENDINGER)
} //sendMeldingerTilEnhet

/**
 * Denne funksjonen tar for seg selve utsendingen av meldingen, etterfulgt av kontroll om alt gikk bra. <br />
 * Funksjonen er synchronized for å sørge for at meldinger sendes ut i korrekt rekkefølge. <br />
 * Dette gjelder spesielt ved utsending av koordinater. */
private synchronized void sendMeldinger(ArrayList<String> utsendingsListe) {
    try {
        //Hent resultatet av masseutsendingen via GCM
        MulticastResult result = sender.send(message, utsendingsListe, utsendingsListe.size());
        //opprett liste over resultatene og sjekk resultatet
        List<Result> resultatListe = result.getResults();
        for (int i = 0; i < utsendingsListe.size(); i++) {
            //hent ut parameterne
            String registrationID = utsendingsListe.get(i);
            Result resultat = resultatListe.get(i);
            /*
             * Hent ut meldingens ID og sjekk etter følgende:
             * - Hvis meldingsID er null, så betyr det at det var en feil;
             *   kall på funksjonen getErrorCodeName()
             * - Hvis meldingsID ikke er null, betyr det at melding ble sendt;
             *   kall på getCanonicalRegistrationId()
             *   -> Hvis getCanonicalRegistrationId() returnerer null,
             *       trenger en ikke gjøre noe, hvis den har returverdi;
             *       oppdater registrationID
             *
             * Kilde:
             http://developer.android.com/reference/com/google/android/gcm/server/Result.html
             */
            String meldingsID = resultat.getMessageId();
            if (meldingsID != null) {
                String canonicalRegId = resultat.getCanonicalRegistrationId();
                if (canonicalRegId != null) {
                    //samme enhet har flere registrationID'r, oppdater ArrayList
                    Databehandling.oppdaterRegistrationID(registrationID,
canonicalRegId);
                } //if (canonicalRegId != null)
            } else {
                String feilKode = resultat.getErrorCodeName();
                if (feilKode.equals(Constants.ERROR_NOT_REGISTERED)) {
                    //enhet er ikke lenger tilkoblet GCM, fjern den fra server
                    Databehandling.avRegistrerEnhet(registrationID);
                } else {
                    if (meldingSendesFraWeb) {
                        //en mer kritisk feil oppsto, vis melding til administrator
                        String tilbakemelding = "<font color='red'>Feil under
sending av melding til "
+ registrationID + ": " + feilKode +
"</font>";
                        Databehandling.setTilbakemelding(tilbakemelding);
                    } //if (meldingSendesFraWeb)
                } //if (feilKode.equals(Constants.ERROR_NOT_REGISTERED))
            } //if (meldingsID != null)
        } //for
    }
}

```

```

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (Exception ex) {
            ex.printStackTrace();
        } //try/catch
    } //sendMeldinger

/**
 * Klasse som oppretter en tråd. <br />
 * Tråden(e) som opprettes ved bruk av klassen tar for seg utsending av melding
 * til flere enheter.
 * @author Knut Lucas Andersen
 */
private class MultiUtsendingAvMeldinger extends Thread implements Runnable {
    private final ArrayList<String> _utsendingsListe;

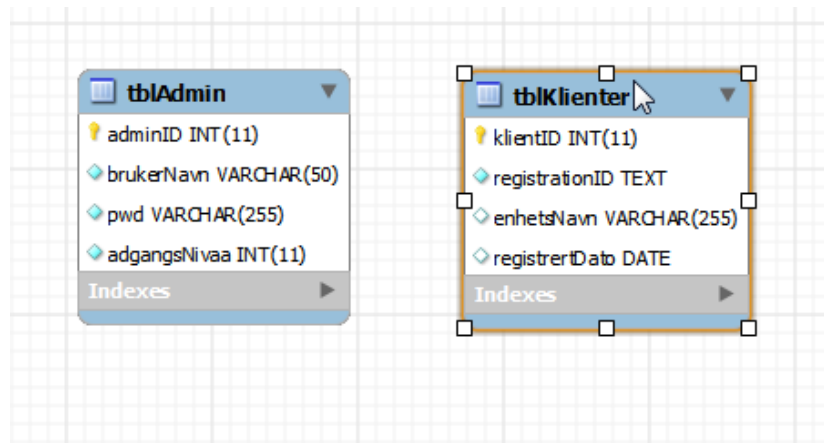
    /**
     * Konstruktør som mottar ArrayList. <br />
     * ArrayList inneholder enhetene som skal motta meldingen.
     * @param utsendingsListe - ArrayList<String>;
     */
    public MultiUtsendingAvMeldinger(ArrayList<String> utsendingsListe) {
        this._utsendingsListe = utsendingsListe;
    } //konstruktør

    public void run() {
        sendMeldinger(_utsendingsListe);
    } //run
} //MultiUtsendingAvMeldinger
} //SendMeldinger

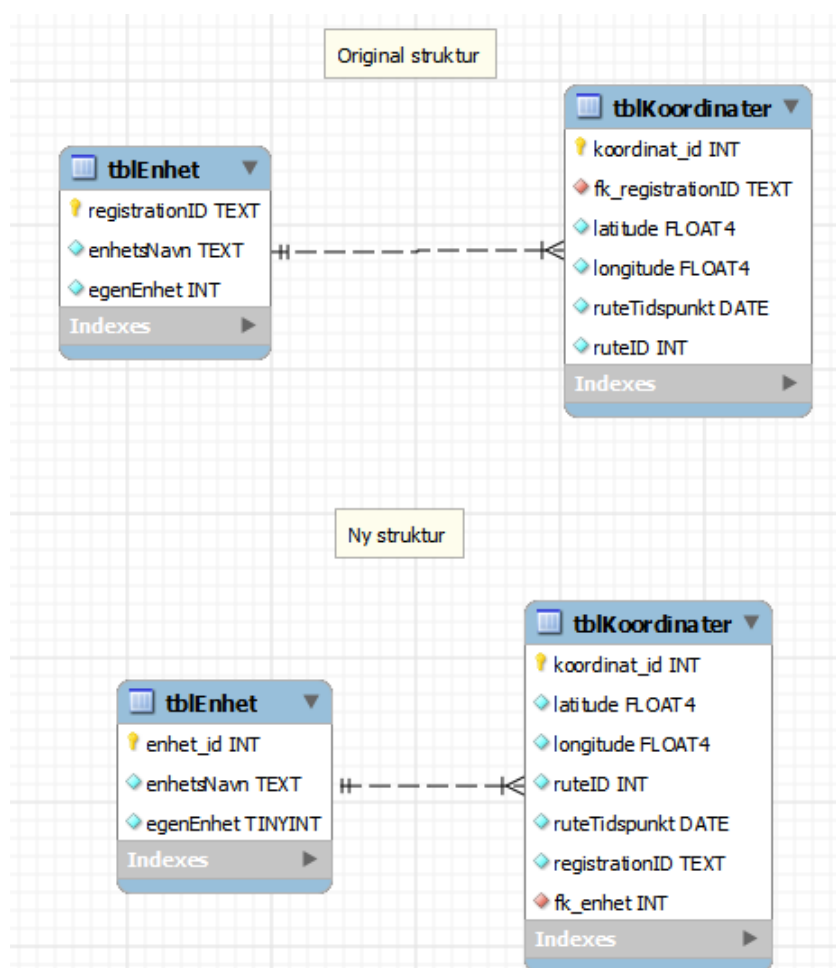
```

## Vedlegg 5: MySQL og SQLite database<sup>29</sup>

### MySQL databasen på server



### SQLite databasen på Android (klient)



<sup>29</sup> Modellene er lagd ved å bruke MySQL's modelleringsverktøy