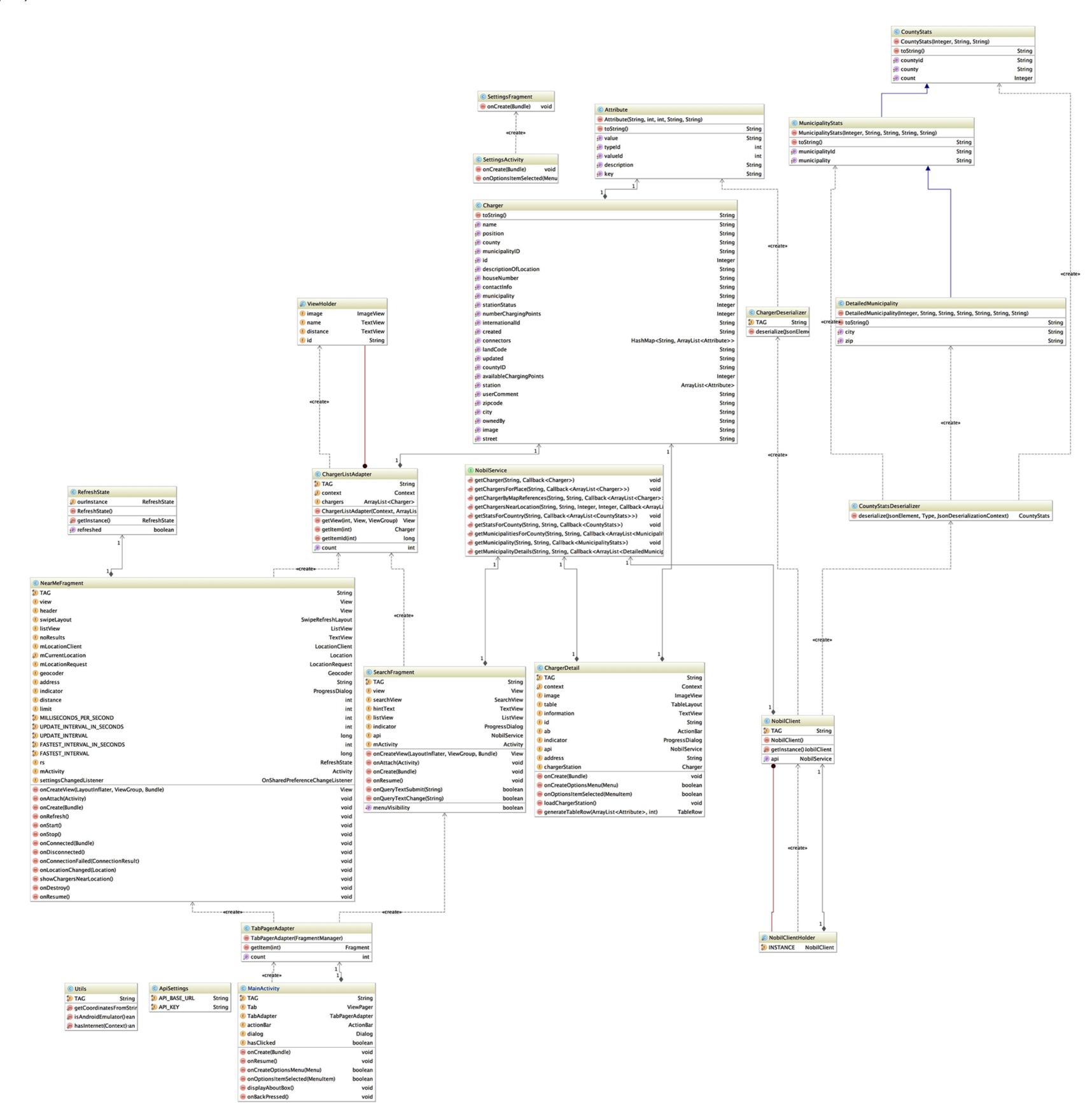
# Ladestasjoner

Utført og levert av Even Holthe <s189124@stud.hioa.no>

## 1.0 Introduksjon

Dette dokumentet dokumenter mappeinnlevering #3 i faget Applikasjonsutvikling (DAVE3600). I dette dokumentet finner du generelle begrunnelser på design- og implementasjonsdetaljer, samt en oversikt over de forskjellige komponentene i applikasjonen.



Klassediagram av Ladestasjoner-prosjektet

### 2.0 Ladestasjoner

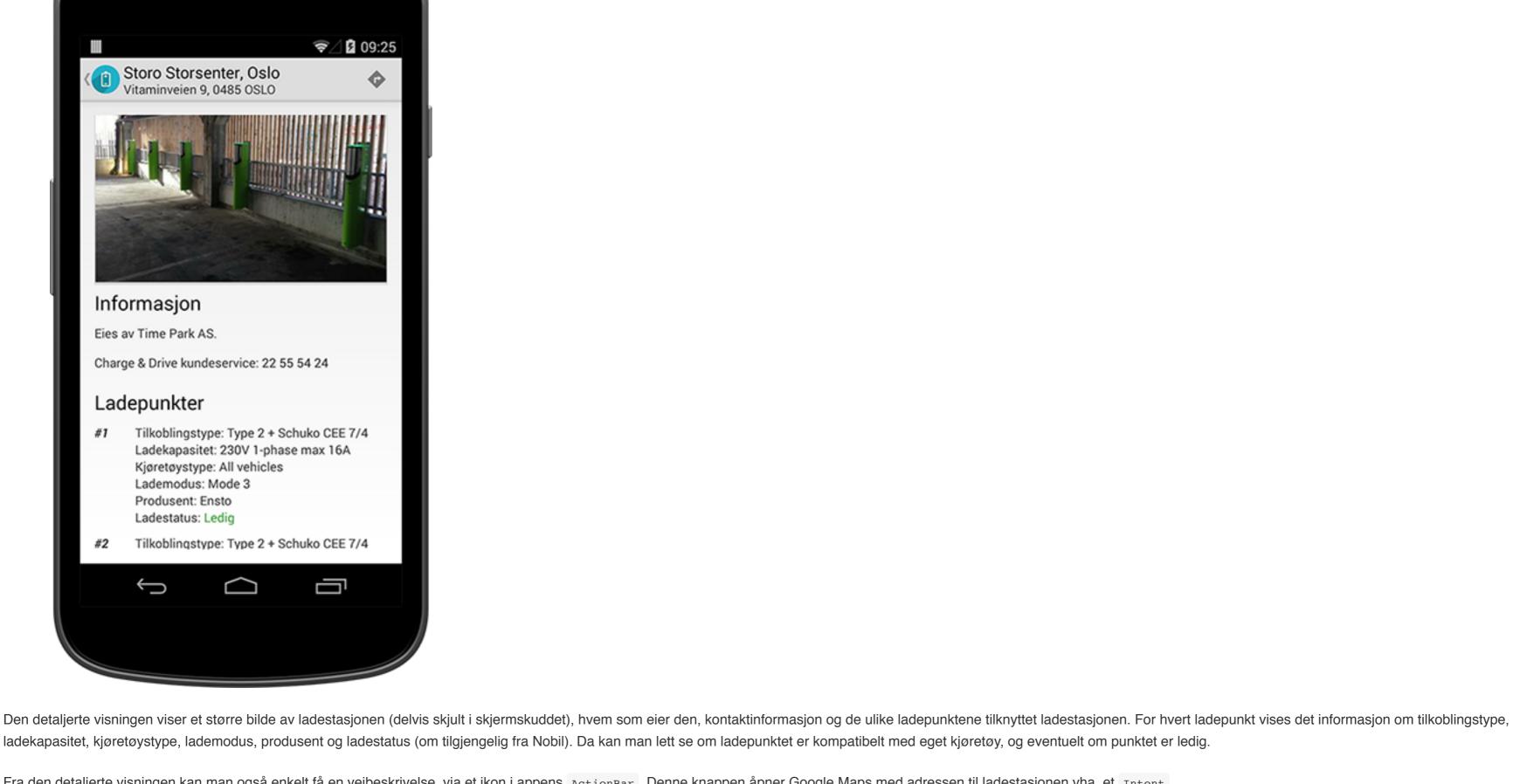
1. Kunne vise ladestasjoner i brukerens nærhet

Idéen kom tidlig med å lage en app som kunne vise ladestasjoner for elbiler. Til dette har Nobil et API som tilbyr data under en liberal lisens. Jeg bestemte meg da for at applikasjonen skal tilby to hovedfunksjoner:

- Til dette skulle brukes lister, da dette er en hensiktsfull måte å presentere data på. For hver oppføring i listen kan man få se en detaljert visning.

2. Kunne søke opp geografiske steder og vise ladestasjoner

2.1 Detaljert visning



Fra den detaljerte visningen kan man også enkelt få en veibeskrivelse, via et ikon i appens ActionBar. Denne knappen åpner Google Maps med adressen til ladestasjonen vha. et Intent. 2.2 I nærheten



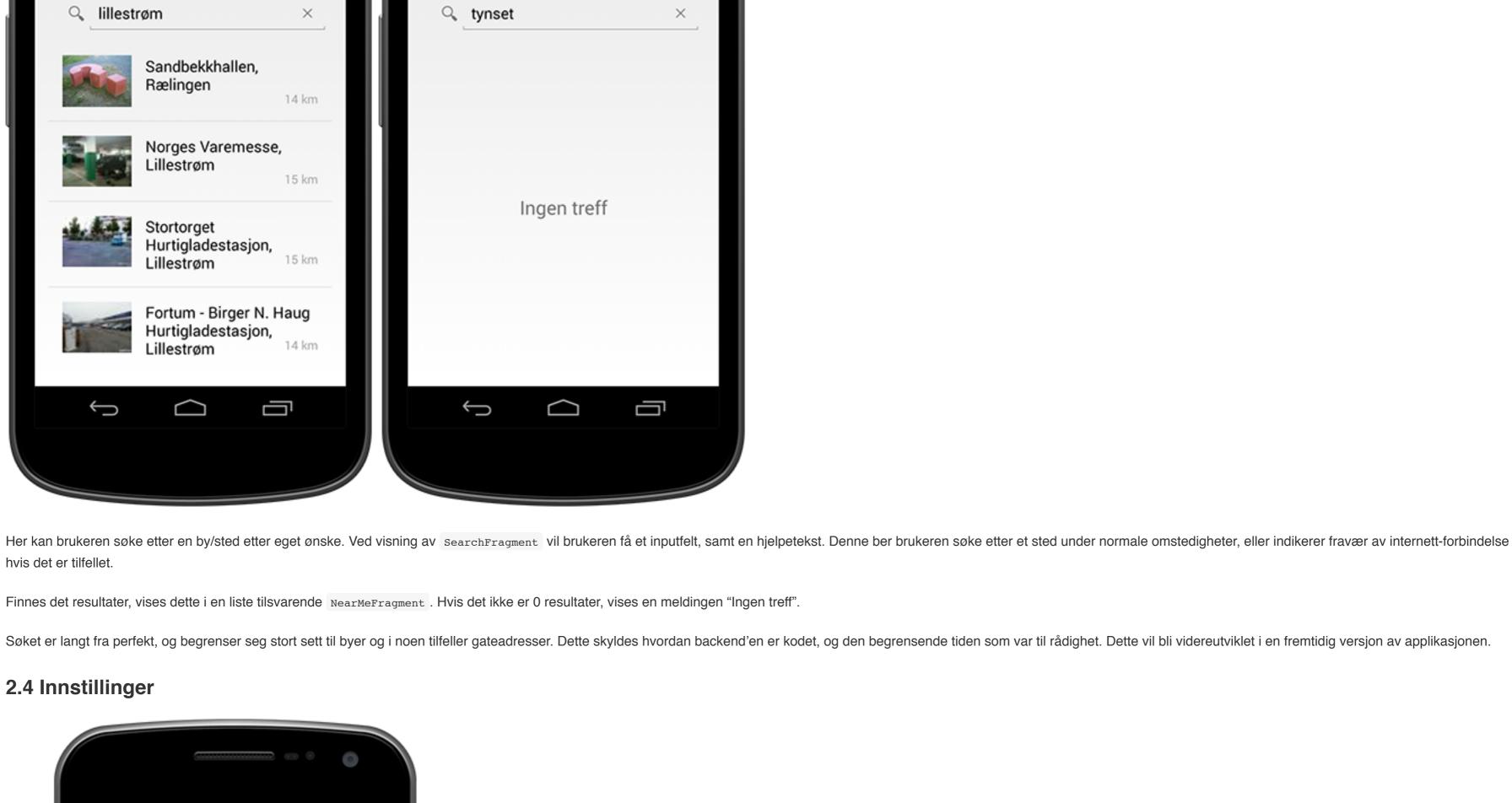
Q søk

**■** INÆRHETEN

2.3 Søk

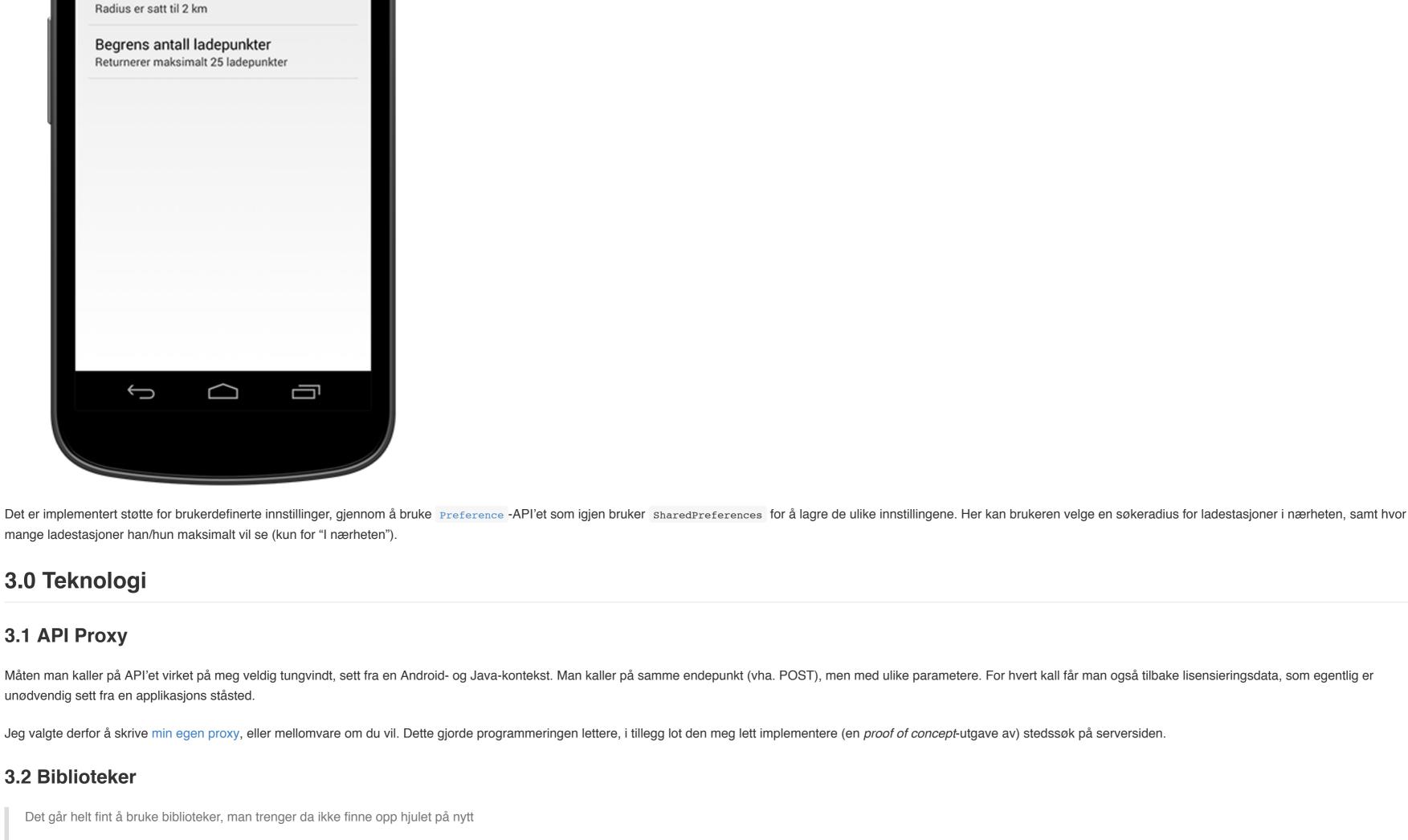
🛜 🛭 09:23 🛜 🛮 13:04 Ladestasjoner Ladestasjoner

Q søk



INÆRHETEN

令⊿ 🛭 09:36



Innstillinger

INÆRHETEN

Søkeradius

Måten man kaller på API'et virket på meg veldig tungvindt, sett fra en Android- og Java-kontekst. Man kaller på samme endepunkt (vha. POST), men med ulike parametere. For hvert kall får man også tilbake lisensieringsdata, som egentlig er unødvendig sett fra en applikasjons ståsted. Jeg valgte derfor å skrive min egen proxy, eller mellomvare om du vil. Dette gjorde programmeringen lettere, i tillegg lot den meg lett implementere (en proof of concept-utgave av) stedssøk på serversiden.

I applikasjonen benytter jeg meg av noen biblioteker. Dette gjøres for å spare kodeskriving, og for å spare systemets ressurser. På en mobil enhet er ytelse viktig, da det er svært begrensende ressurser tilgjengelig til hver applikasjon. Benyttes for nettverkskall, da dette biblioteket tilyr funksjonalitet for å kalle REST-API'er på en lettfattelig og sikker måte.

Brukes for å dynamisk laste inn bilder ettersom de trengs å vises i et Listview. Picasso cacher bildedata, så de kan kjappere lastes inn senere. Brukes for SwipeRefreshLayout ("Pull to refresh"-funksjonalitet), FragmentActivity og ViewPager (for å kunne bla gjennom faner).

Log.i(TAG, "Detected emulator, simulating location...");

mockLocation.setLatitude(59.919570); mockLocation.setLongitude(10.735562);

mLocationClient.setMockLocation(mockLocation);

mCurrentLocation = mockLocation;

onLocationChanged(mockLocation);

Location mockLocation = new Location("Pilestredet 35 Mock Location Provider");

### I emulatoren hadde jeg problemer med å få en geografisk posisjon når jeg brukte Google Play Services. Man kan komme rundt dette ved å sende en geo fix -kommando vi adb eller gjennom telnet, men det krever at man gjør det for hver instans av applikasjonen som skal kjøres i emulatoren. Dette viste seg å være tungvindt i testing, så jeg konsulterte med faglærer og vi ble enige om å prøve en alterativ måte.

Lokasjon gjennom Google Play Services

if (Utils.isAndroidEmulator()) {

3.3 Utfordringer

- Torunn Gjester, faglærer

Slik det fungerer nå, detekteres det om det er emulatoren som kjører. Hvis det er tilfellet, opprettes en fixed posisjon ved hjelp av følgende kodesnutt (implementert i NearMeFragment): // Provide a fixed location for the emulator

```
4.0 Kilder

   Ikon hentet fra IconArchive

• Ladestasjondata fra Nobil under Creative Commons 3.0 Unported-lisensen.
```

For å konvertere JSON-data til Java-objekter bruker jeg Google's Gson-bibliotek. Dette besparer tid når det kommer til programmering og ytelsestuning.