

HAPPYMAP

Mappe 1 og 2 gjorde vi individuelt, men for å samle kunnskapen valgte vi å samarbeide på den siste mappeinnleveringen. Som de to forrige, vil denne være utviklet med Kotlin og Compose. Til versjonskontroll brukte vi Github.

Vi startet med å koble oss til server, og opprette table Places med feltene description, address, latitude og longitude.

Webtjenester:

```
apple ~ /www > cat jsonout.php
<?php
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
$con = mysqli_connect('localhost', 's375128', '', 's375128');
$sql = "SELECT * FROM Places";
$tabell = mysqli_query($con, $sql);
mysqli_close($con);
while ($row = mysqli_fetch_assoc($tabell)) {
    $output[] = $row;
}
print(json_encode($output));
?>
```

```
apple ~ /www > cat jsonin.php
<?php
$con = mysqli_connect('localhost', 's375128', '', 's375128');
getDescription = $_REQUEST['description'];
$address = $_REQUEST['address'];
$latitude = $_REQUEST['latitude'];
$longitude = $_REQUEST['longitude'];
$sql = mysqli_query($con, "INSERT INTO Places (description, address, latitude, longitude) VALUES ('$description', '$address', $latitude, $longitude');");
mysqli_close($con);
?>
```

For å jobbe med nettverkskommunikasjon valgte vi å gå for Retrofit. Retrofit støtter både synkrone og asynkrone forespørsler. I vårt tilfelle har vi valgt å bruke Kotlin Coroutines for å gjøre nettverkskallene asynkrone, noe som tillater at de kjøres i bakgrunnen uten å blokkere hovedtråden. Android Developer hadde en tutorial på dette som hjalp oss:

<https://developer.android.com/codelabs/basic-android-kotlin-compose-getting-data-internet?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-5-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-getting-data-internet#0>

For å sjekke om API-et fungerte som forventet brukte vi Postman:

```

1  [{"id": "1", "description": "Park med fin utsikt", "address": "Parkveien 12,
2  Oslo", "latitude": "59.9139", "longitude": "10.7522"}, {"id": "2", "description": "Kaf med god kaffe",
3  "address": "Karl Johans
4  gate 20, Oslo", "latitude": "59.9115", "longitude": "10.7413"}, {"id": "3", "description": "Teststed",
5  "address": "Gateveien
6  123", "latitude": "59.92", "longitude": "10.75"}, {"id": "4", "description": "Park", "address": "Oslo",
7  "latitude": "59.9139", "longitude": "10.7522"}]

```

Når dette var på plass kunne vi gå videre med å få inn Google Maps. Her fant vi to nyttige tutorials fra Android Developer som var nyttige:

Tutorial “Map with marker”:

<https://developers.google.com/maps/documentation/android-sdk/map-with-marker>

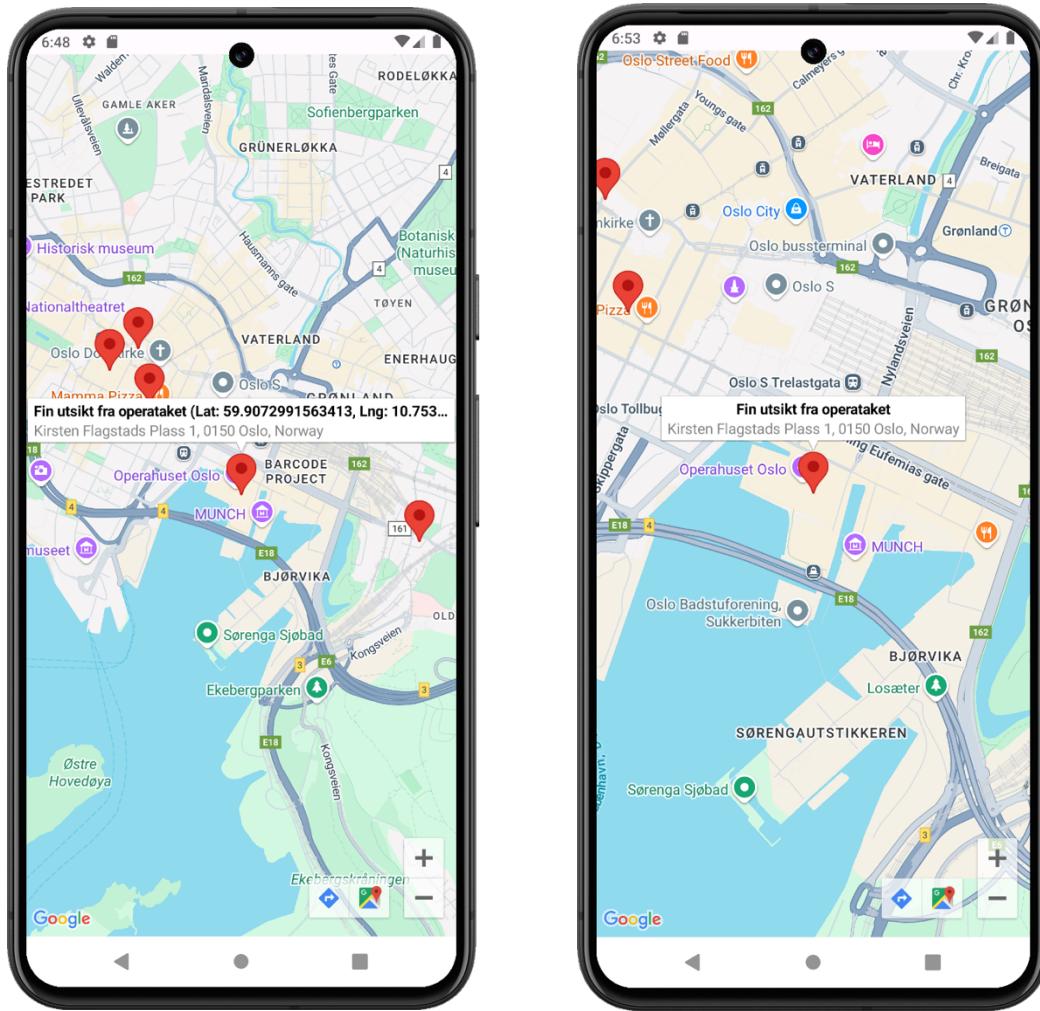
Tutorial: “Add a map to your android app”:

<https://developers.google.com/codelabs/maps-platform/maps-platform-101-compose#0>

API key-en er lagret i secrets.properties. Dette for å holde den utenfor kildekoden og ikke dele den unødvendig over github. Vi møtte derimot et problem underveis, som vi brukte noen dager på å finne ut av. Vi fikk feilmelding på API-token.

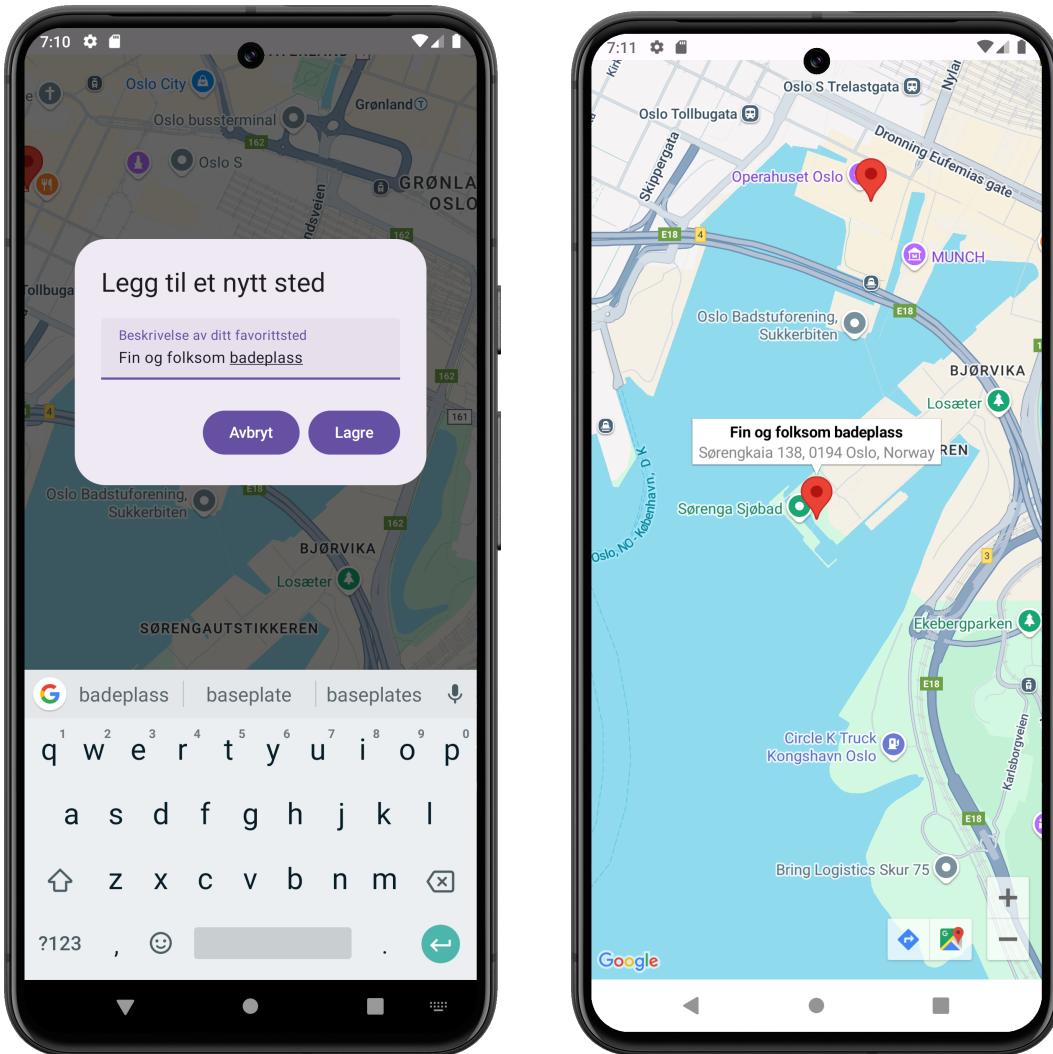
Vi antok at dette hadde med noen innstillinger inne på Google Cloud og API-key å gjøre, eller problemer med VPN, men etter mye logging og googlesøk kom vi frem til at emulatorenens performance måtte settes på Cold Boot. Denne stod på Quick Boot opprinnelig. Ved å bytte til Cold Boot fikk emulatoren en full oppstart og tjenester som nettverk og Google Play ble initialisert på nytt, og vi fikk inn Google Maps uten problem.

Vi brukte Geocoding API ved å lage en funksjon som henter adresse fra koordinater ved hjelp av Retrofit. Funksjonen håndterte veldig godt respons ved å returnere adresse, og feilhåndtering ble lagt til for å unngå krasj. Adressen ble deretter brukt i UI-et for å vise adressen når brukeren trykket på en markør.



Vi hadde først visningen som bildet over til venstre, men da gikk Lat og Lng utenfor infoboksen, da denne har begrenset med plass. Vi valgte derfor å fjerne Lat og Lng fra visningen, og heller kun ha description og adresse. Visningen ble mye mer brukervennlig, da koordinater sjeldent blir brukt.

Bruker legger inn nytt sted ved å trykke på kartet der man ønsker å få inn markør. Deretter får man opp et tekstmfelt der man legger inn beskrivelse. Man kan deretter avbryte handlingen eller lagre.



Etter lagring kommer markøren opp og ved å trykke på den får man opp beskrivelse og adresse.

I MariaDB kan man se at description, address, latitude og longitude blir lagret:

MariaDB [s375128]> SELECT * FROM Places;				
id	description	address	latitude	longitude
37	God kollektittransport	Stenersgata 1, Oslo, Norway	59.911	10.746
38	Bra shoppingmuligheter	Nedre Slottsgate 13-15, 0157 Oslo, Norway	59.91240973907075	10.742727555334568
39	Koselig sted for vandring	Henrik Ibsens gate 1, 0010 Oslo, Norway	59.9183443431413	10.728753246366978
40	Kul scatepark	Bispegata 4, 0192 Oslo, Norway	59.90537544508	10.768187083303928
41	Fine glass her	Stortorvet 9, 0155 Oslo, Norway	59.913302913465095	10.745072811841965
42	Fin utsikt fra operataket	Kirsten Flagstads Plass 1, 0150 Oslo, Norway	59.9072991563413	10.753552615642548
43	Fin og folksom badepllass	Sørenskai 138, 0194 Oslo, Norway	59.90103319405905	10.751420930027962

7 rows in set (0.000 sec)