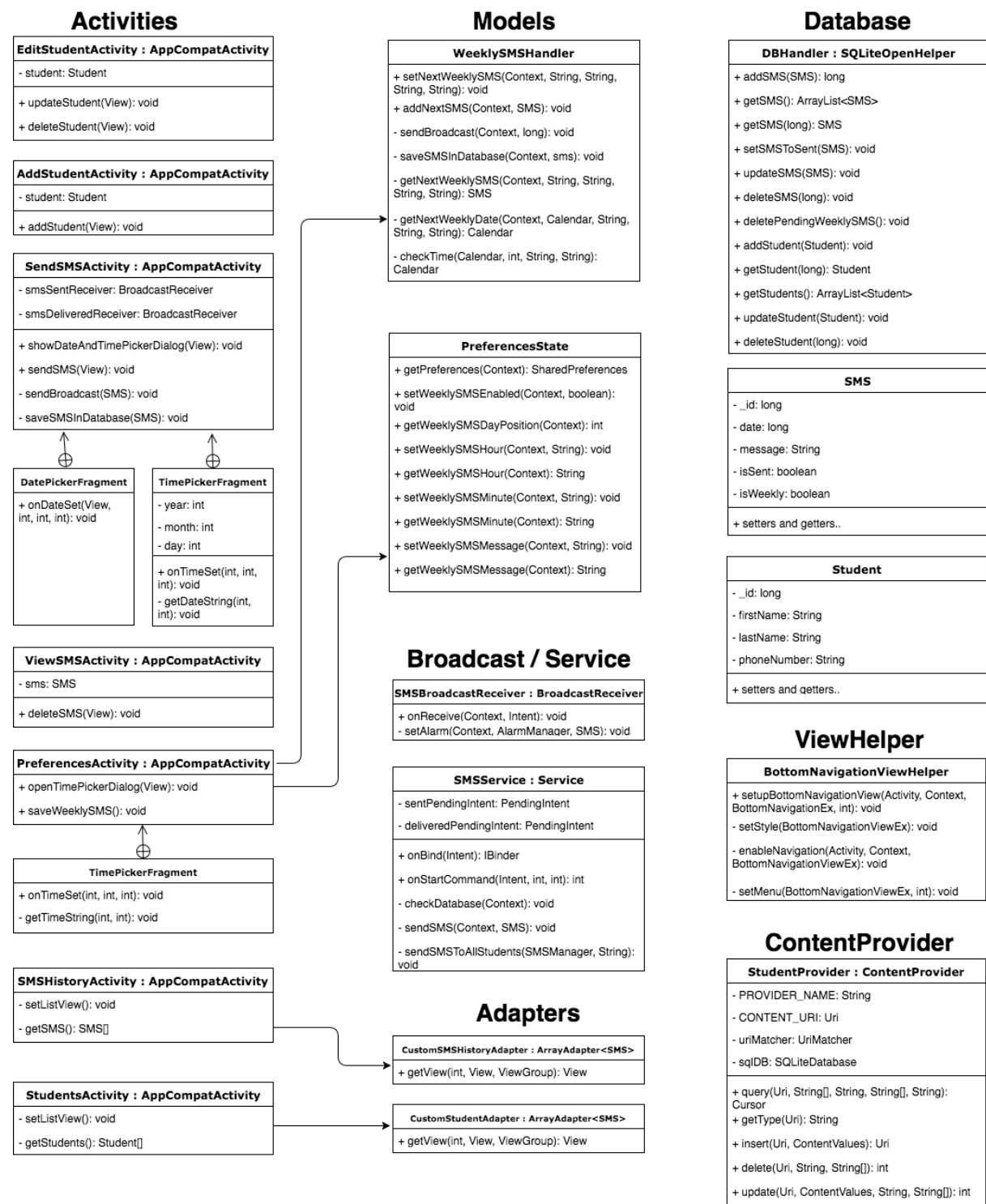


Applikasjonsutvikling (Mappe 2) - Rapport

Stian Tornholm Grimsgaard - s305537

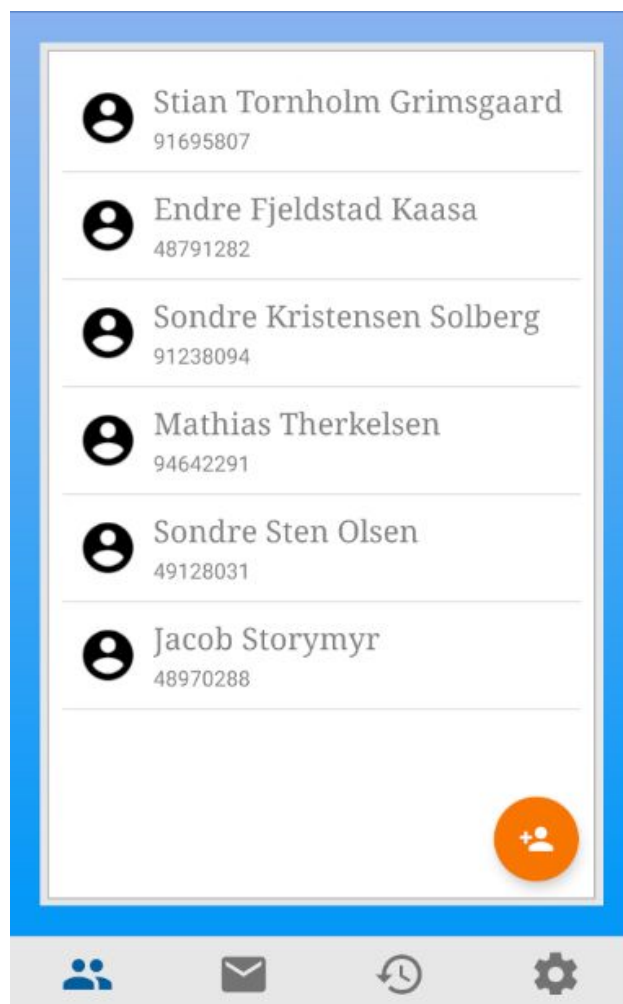
Klasser



StudentsActivity

StudentsActivity er den første aktiviteten man møter når man starter opp applikasjonen. Denne består av en liste av alle studenter som er lagt til i databasen. Listen er egendefinert (ref. *CustomStudentAdapter*) og består av et bilde, fornavn, etternavn og telefonnummer. Alle studenter blir hentet ut fra databasen ved hjelp av *DBHandler*.

Ved å klikke på en student i listen, kommer man videre til en ny aktivitet der man kan endre informasjon (*EditStudentActivity*). Aktiviteten har også en såkalt *FloatingActionButton* som navigerer deg til en ny aktivitet der man kan legge til en ny student (*AddStudentActivity*).

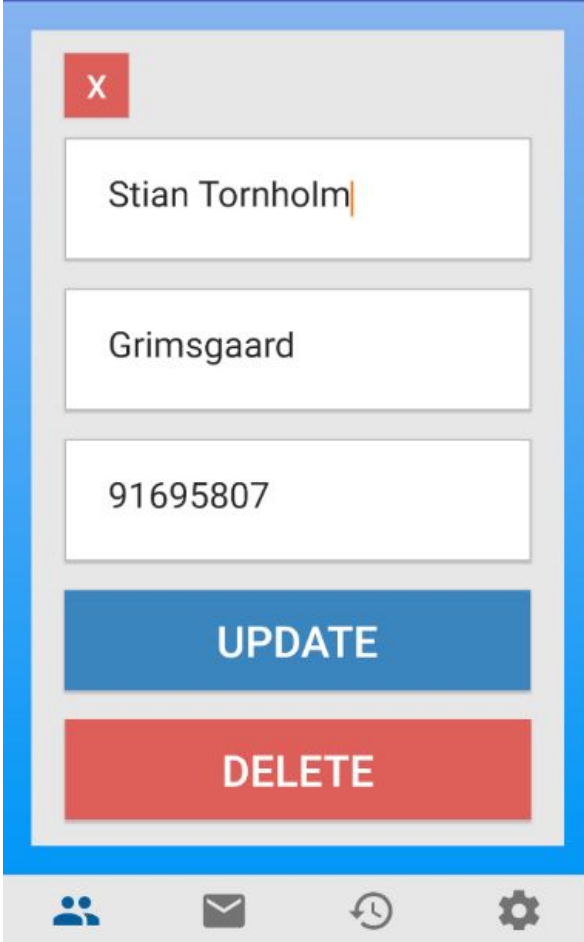


Students-aktiviteten.

EditStudentActivity

EditStudentActivity er aktiviteten man kommer til når man trykker på en students i listen beskrevet ovenfor. Her har man mulighet til å endre fornavn, etternavn og telefonnummer, for å deretter oppdatere informasjonen ved "Update"-knappen. Man kan også slette studenten ved å trykke på "Delete"-knappen. Både "Update" og "Delete" blir utført ved hjelp

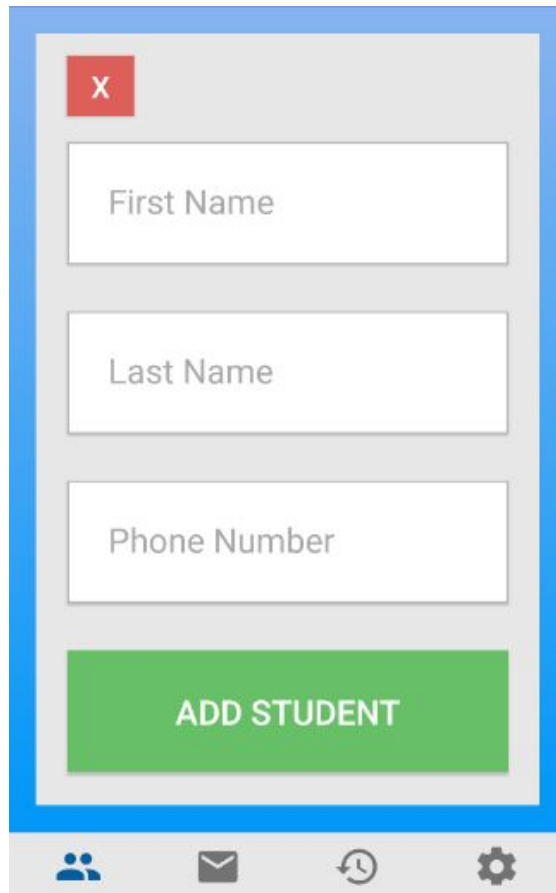
av *DBHandler*. Ved å trykke på den røde “X”-knappen øverst til venstre, kommer man tilbake til listen av studenter.



EditStudent-aktiviteten

AddStudentActivity

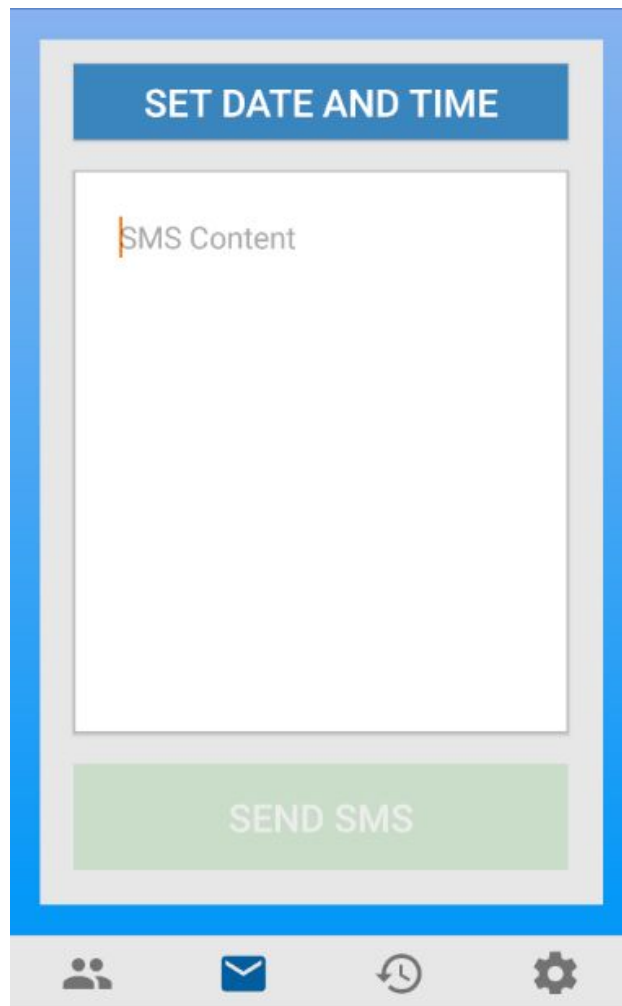
AddStudentActivity er aktiviteten man kommer til når man trykker på den oransje FloatingActionButton som beskrevet i Students-aktiviteten. Aktiviteten består av 3 EditText-widgets, hvor man må fylle inn fornavn, etternavn og telefonnummer for å få lov til å legge til studenten. Studenten blir deretter lagt til i databasen ved hjelp av *DBHandler*.



AddStudent-aktiviteten

SendSMSActivity

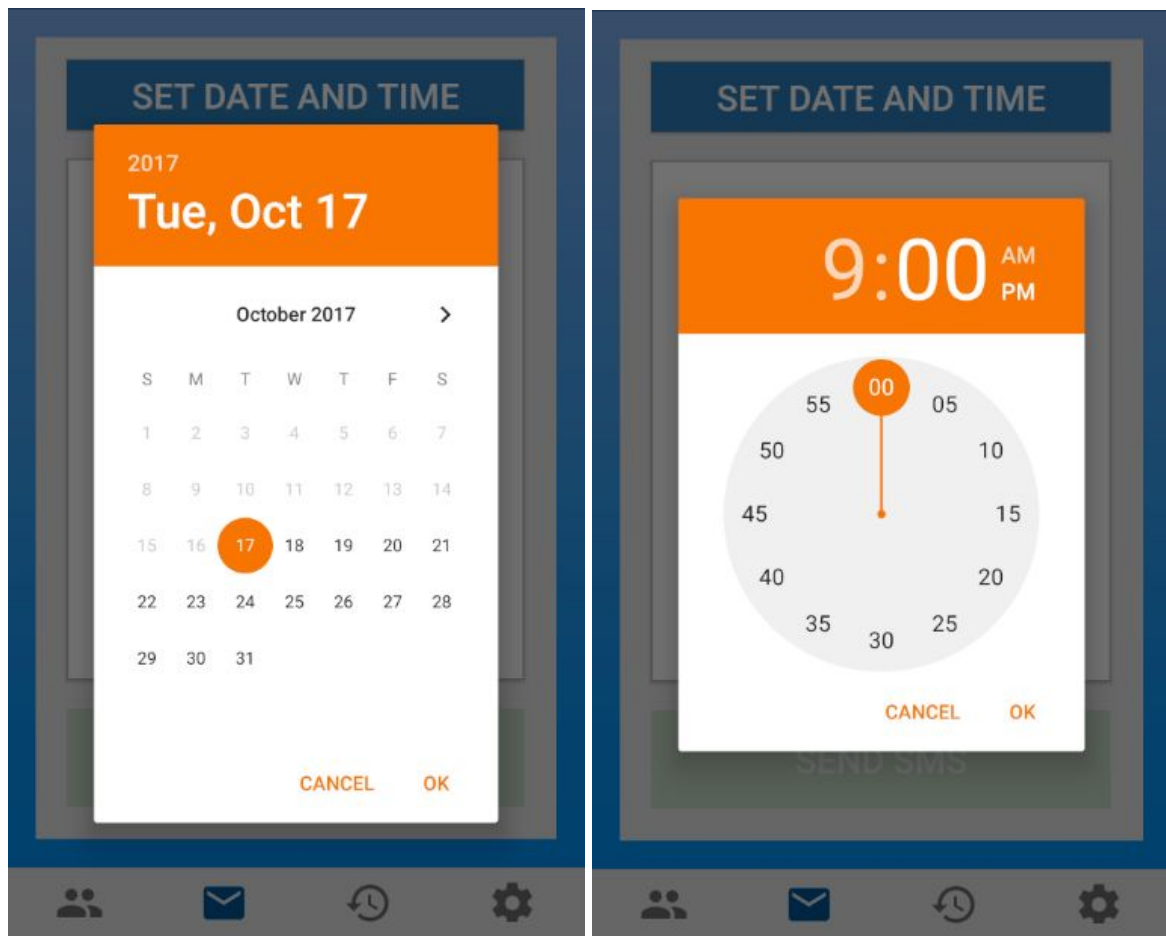
SendSMSActivity er aktiviteten der man kan sende SMS til alle studenter på et gitt tidspunkt. For å få tilgang til "Send SMS"-knappen, må man både ha satt en dato og tid, samt lagt til en melding å sende ut. Skjermbildet under viser hvordan aktiviteten ser ut når man først åpner den ved hjelp av navigasjonsmenyen nederst.



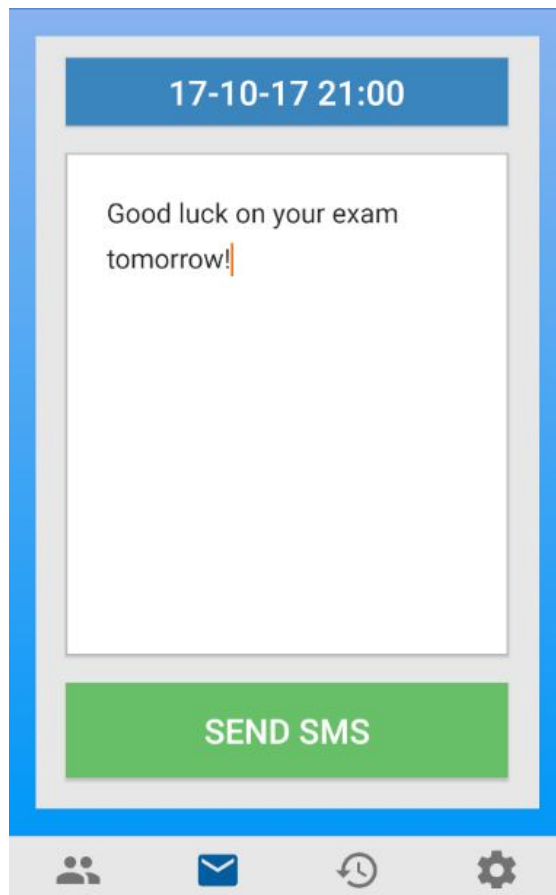
SendSMS-aktiviteten

Man kan deretter sette dato og tid ved å trykke på "Set date and time"-knappen. Da vil en *DatePickerDialog* åpnes, hvor man kan velge en dato. Man har kun lov til å velge gjeldende og fremtidige datoer.

Etter man har valgt dato, trykker man på "Ok"-knappen. Da vil en *TimePickerDialog* åpnes, hvor man kan sette en tid.



Når både dato og tid er satt ved hjelp av dialogene, vil knappen bli oppdatert med valgt dato og tid. Når man i tillegg skriver inn meldingen som skal sendes, vil "Send SMS"-knappen bli tilgjengelig og man kan sende SMS-en.

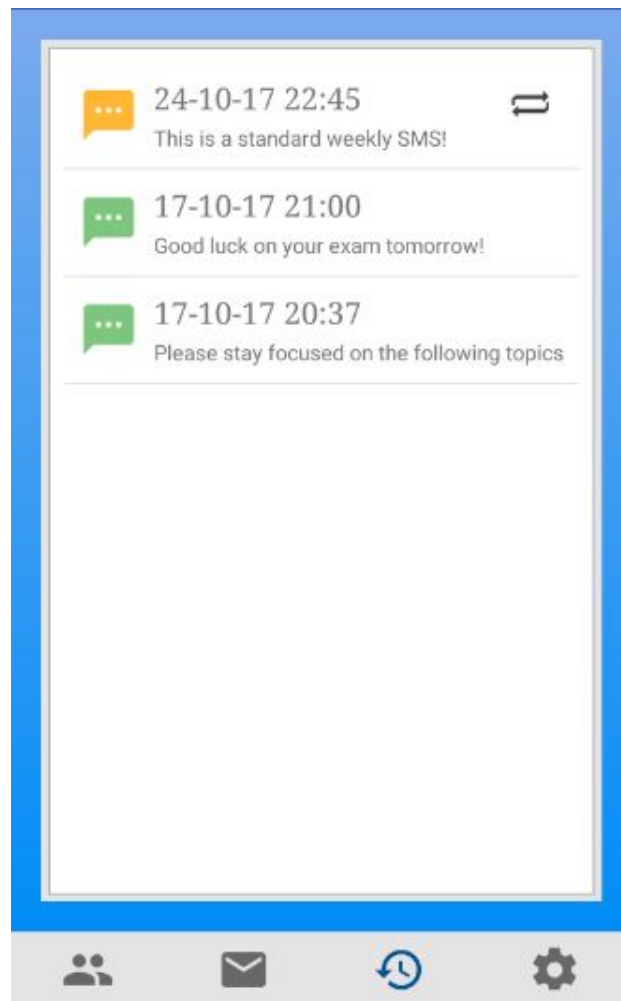


SendSMS-aktiviteten

Når man sender SMS-en blir man første gang bedt om å gi applikasjonen nødvendige tillatelser. Bak kulissene blir SMS-en lagret i databasen, før applikasjonen sender en melding om å starte *Broadcast Receiveren* “*SMSBroadcastReceiver*”, gangen herfra blir beskrevet senere i rapporten.

SMShistoryActivity

SMShistoryActivity er aktiviteten som viser historikken over alle SMS som er sendt, samt alle SMS som venter på å bli sendt. Aktiviteten består av en liste av SMS hentet fra databasen, tilnærmet likt som listen av studenter. Listen er her også egendefinert og består av et bilde, dato, tid og meldingen. Ved å trykke på en SMS i listen kommer man videre til aktiviteten “*ViewSMSActivity*”, der man kan se hele meldingen, samt slette den.

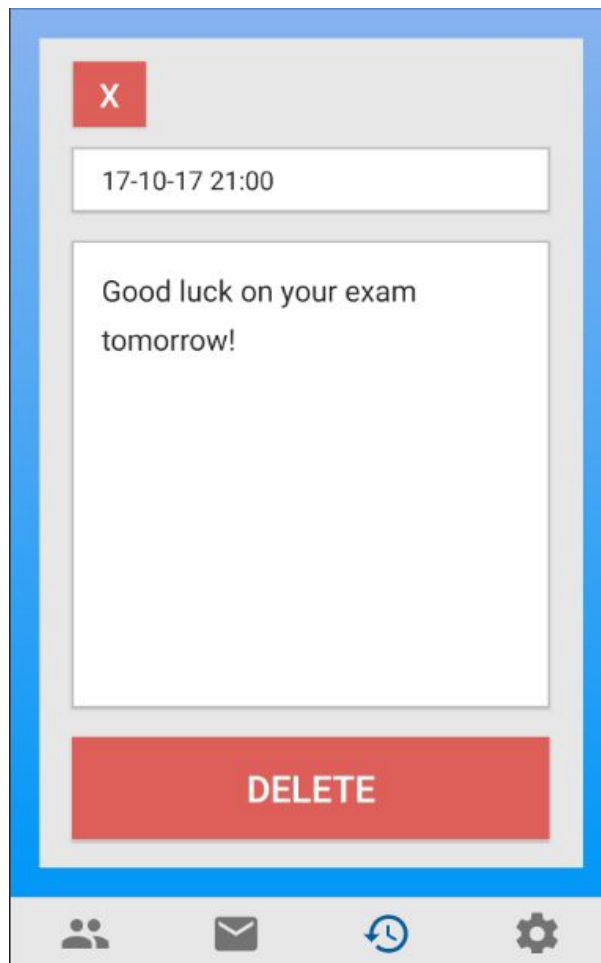


SMSTHistory-aktiviteten

Som man kan se i skjermbildet over, er “SMS”-bildet representert i både **oransje** og **grønt**. Et oransje ikon vil si at SMS-en ikke har blitt sendt enda, mens et grønt ikon tilsier at SMS-en har blitt sendt. I tillegg vil alle ukentlig SMS ha et gjentakelse-symbol øverst til høyre, som tilsier at SMS-en blir sendt ukentlig. Den neste ventede SMS-en vil også alltid ligge øverst i listen, for å opprettholde en god oversikt.

ViewSMSActivity

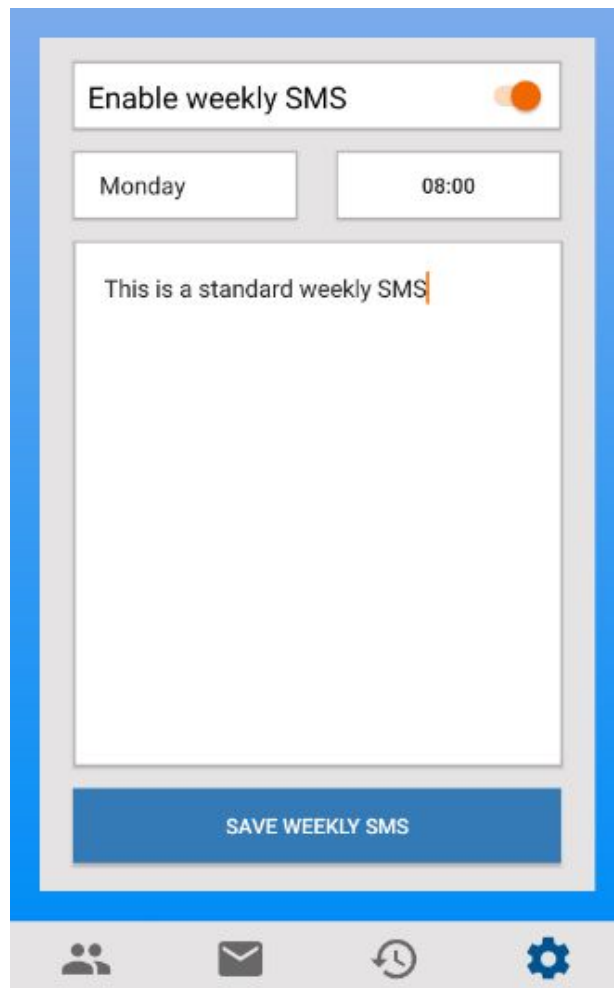
ViewSMSActivity er aktiviteten man kommer til når man trykker på en SMS i listen, som nevnt over. Her har man mulighet til å se den fulle meldinger, samt slette den ved hjelp av “Delete”-knappen.



ViewSMS-aktiviteten

PreferencesActivity

Den siste aktiviteten er *PreferencesActivity*. Her har man mulig til å velge om man vil godkjenne sending av SMS, samt muligheten til å legge til en ukentlig SMS som vil bli sendt til alle studenter.



Dagene kan velges ved hjelp av en “*Spinner*”-widget, hvor man kan velge alle dagene ved hjelp av en dropdown-list. Tidspunkt blir valgt ved hjelp av *TimePickerDialog*, likt som når man sender en vanlig SMS. Hvis enten dag, tidspunkt eller melding er endret, gir det mulighet til å trykke på “Save Weekly SMS”-knappen.

Alt i denne aktiviteten blir lagret i *SharedPreferences* i klassen ***PreferencesState***. Dermed vil innholdet i weekly SMS være det samme som lagret sist, når man går inn på aktiviteten. Ukentlig SMS fortjener en forklaring på hvordan den er bygd opp. Når man trykker “Save Weekly SMS”-knappen blir først angitt dag, tidspunkt og melding satt på nytt i *PreferencesState*. Deretter opprettes en instans av *WeeklySMSHandler*, hvor det blir satt en ny ukentlig SMS. Mer om dette under.

WeeklySMSHandler

WeeklySMSHandler tar seg av alt som skjer bak kulissene når brukeren lagres en ukentlig SMS i aktiviteten beskrevet ovenfor. Hvis den lagrede SMS-en er en oppdatering av en

allerede opprettet ukentlig SMS, vil den kun oppdatere SMS-en i databasen med nye parametre. Hvis det ikke allerede er opprettet en ukentlig SMS, vil den legge den nye SMS-en i databasen. All ukentlig SMS blir altså lagret i databasen, likt som ordinære SMS.

Jeg har også testet muligheten for å lagre SMS-en i *SharedPreferences*, men velger heller å legge den i databasen. Dette gjøres for å blant annet enkelt kunne hente den ut i listen av SMS-er, sammen med de ordinære SMS-ene.

I *WeeklySMSHandler* ligger metoden *getNextWeeklySMS()* som returnerer en ny SMS. Denne metoden bruker videre en metode *getNextWeeklyDate()* og *checkTime()* som finner riktig dato og tid i henhold til hva brukeren har valgt. Hvis brukeren for eksempel har valgt at den ukentlige SMS-en skal sendes hver mandag, finner metodene den riktige datoen i henhold til valgt dag og tid og returnerer denne.

Hvis jeg setter at den ukentlige SMS-en skal sendes hver mandag klokken 09.00, og dagens dato er en mandag klokken 10:00, vil den gi den SMS-en en datoen til neste mandag klokken 10:00.

Hver gang en ukentlig SMS blir sendt, vil den også automatisk opprette en ny SMS i databasen en uke frem i tid ved hjelp av metoden *addNextSMS()*.

SMSBroadcastReceiver

SMSBroadcastReceiver blir kalt på ved to tilfeller, og er implementert deretter.

- 1) Hver gang telefonen starter opp (*Boot Completed*).
- 2) Hver gang man sender en ny SMS.

- Når telefonen starter opp, går den igjennom alle SMS som er lagret i databasen. Hver gang den finner en SMS som ikke allerede er sendt, så setter den en Alarm på SMS-en med tiden den har i databasen. Dermed vil *AlarmManager* og alle SMS bli satt på nytt hver gang telefonen starter opp.
- Hver gang man sender en SMS vil også *SMSBroadcastReceiver* bli kalt på. Forskjellen er at her vil SMS-en sin *ID* bli sendt med i en *Bundle*, og dermed vil det kun bli satt en Alarm for denne ene SMS-en.

Når en alarm blir utløst, vil en *PendingIntent* starte serviceklassen *SMSService*.

SMSService

SMSService er prosjektet sin eneste serviceklasse, og blir kun instansiert når en alarm blir utløst. Klassen sin hovedoppgave er å sende SMS til alle studenter.

Når klassen blir startet, går den gjennom SMS-ene i databasen, og sender alle SMS der tiden (representert ved en *long*) er mindre enn tiden nå (*System.currentTimeMillis()*) og SMS-en ikke allerede er sendt. Hver eneste student i databasen vil motta meldingen, ved hjelp av telefonnummeret som står lagret på den enkelte student.

DBHandler

DBHandler er klassen som tar seg av alt som har med databasen å gjøre. Klassen arver fra *SQLiteOpenHelper* og overstyrer nødvendige metoder som *onCreate()* og *onUpgrade()*. I

DBHandler lager en database ved navn "*studentmanager.db*", og deretter opprettes det en tabell for student og en tabell for SMS. Klassen inneholder en rekke metoder for hvordan man kan legge til, oppdatere, slette og hente ut data fra disse to tabellene. Dataene som ligger i Student og SMS blir representert ved hver sin klasse, som blir beskrevet under.

```
private static final String CREATE_STUDENT_TABLE = "CREATE TABLE " + STUDENT_TABLE_NAME + "(" +
    STUDENT_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
    STUDENT_FIRST_NAME + " TEXT," +
    STUDENT_LAST_NAME + " TEXT," +
    STUDENT_PHONE_NUMBER + " TEXT" +
    ")";
private static final String CREATE_SMS_TABLE = "CREATE TABLE " + SMS_TABLE_NAME + "(" +
    SMS_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
    SMS_DATE + " INTEGER," +
    SMS_MESSAGE + " TEXT," +
    SMS_IS_SENT + " INTEGER," +
    SMS_IS_WEEKLY + " INTEGER" +
    ")";
```

Student

Studentklassen er en representasjon av student-tabellen vist ovenfor. Den inneholder følgende variabler, samt nødvendige konstruktører, setttere og gettere:

```
private Long _id;
private String firstName;
private String lastName;
private String phoneNumber;
```

Noe man lett kan bemerke seg er at det brukes en *INTEGER* i databasen og en *Long* i java klassen. Dette er fordi *SQLite* ikke har kompatibilitet med *Long* verdier. Derimot vil både en *INTEGER* og en *Long* verdi lagre verdier av 8 bytes, så de passer som hånd i hanske.

SMS

SMSSklassen er en representasjon av SMS-tabellen vist tidligere. Den inneholder følgende variabler, samt nødvendige konstruktører, setttere og gettere:

```
private Long _id;
private Long date;
private String message;
private boolean isSent;
private boolean isWeekly;
```

Her kan man bemerke seg at datoen er representert ved en *Long* verdi, og ikke *Date* som er vanlig i databaser. Dette er også fordi *SQLite* ikke har kompatibilitet med *Date*. Java sin innebygde *Date*-klasse har dog en metode som gjør om datoen til en *Long* verdi, så dette byr ikke på store problemer.

SMS-klassen har også to boolean verdier som brukes til å sjekke om meldingen er sendt og om den er ukjentlig. Da jeg har implementert den ukentlige SMS-en i databasen, var dette et nødvendig felt.

StudentProvider

StudentProvider er applikasjonen sin *Content Provider*. Denne lar andre applikasjoner få tilgang til student-dataene på en sikker måte. Man kan få tilgang til *StudentProvider* ved å parse følgende *URI*:

"content://com.example.stiantornholmgrimsgaard.mappe2_s305537.ContentProvider.StudentProvider/studentManager".

Man kan for eksempel legge til en student fra en annen aktivitet ved å kjøre følgende kodelinjer (samt gi rettigheter i Manifest):

```
ContentValues contentValues = new ContentValues();
contentValues.put("firstName", "Ola");
contentValues.put("lastName", "Nordmann");
contentValues.put("phoneNumber", "91695807");
Uri uri = getContentResolver().insert(CONTENT_URI, contentValues);
```

Man kan også for eksempel oppdatere en id (1 i dette tilfellet) med følgende kode:

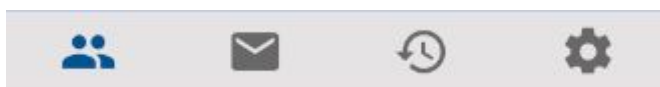
```
contentValues.put("firstName", "Stian");
Uri uri = ContentUris.withAppendedId(CONTENT_URI, 1);
getContentResolver().update(uri, contentValues, null, null);
```

Design

Da jeg har blitt ferdig med det funksjonelle relativt tidlig, har jeg fått brukt mye tid på å lage et brukervennlig design.

BottomNavigationViewHelper

I applikasjonen er det blitt implementert en *BottomNavigationViewEx* for å navigere seg mellom de forskjellige hovedaktivitetene. Dette er et eksternt bibliotek som gir litt flere muligheter enn Android sitt innebygde *BottomNavigationView*, blant annet en navigasjon med kun ikoner. Navigasjonsmenyer inneholder 4 ikoner som representerer studenter, meldinger, historikk og innstillinger.



BottomNavigationViewEx


CustomStudentAdapter

I applikasjonen finnes det to lister. Den første listen representerer studenter og henter ut en egendefinert liste ved hjelp av klassen *CustomStudentAdapter*. Studentlisten består av fullt navn, telefonnummer og et "person-ikon".



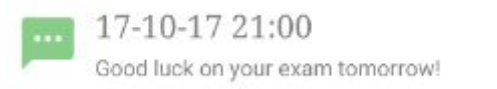
Egendefinert studentliste

CustomSMSHistoryAdapter

Det andre listen i applikasjonen er listen som representerer SMS. Denne er implementert med samme fremgang som ovenfor og inneholder alle SMS i databasen. Listen inneholder et "sms-ikon" som er **grønn** hvis meldingen er sendt, og **oransje** hvis den venter på å bli sendt. Hvis meldingen er ukjentlig så inneholder den et "repetisjons-ikon"  øverst i høyre hjørne. Listen inneholder også tidspunkt samt meldingen sitt innhold.



En ukjentlig SMS som venter på å bli sendt.



En vanlig SMS som har blitt sendt.

Fargebruk

I applikasjonen blir det brukt hovedsakelig 4 farger som er med på å gi brukeren en god brukeropplevelse, samt en følelse av kvalitet:

Blå - Applikasjonen sin hovedfarge. Blir brukt i alle knapper som har med oppdatering å gjøre. For eksempel oppdater student eller oppdater tidspunkt for SMS. I bakgrunnen i alle aktiviteter blir det også brukt en gradient av blåfarger for å gi brukeren en god brukeropplevelse.



Blå blir brukt i knapper som oppdatere.

Grønn - Blir brukt i alle knapper som legger til noe nytt. For eksempel legg til ny student eller send en ny SMS.



Grønn blir brukt i knapper som legger til noe nytt.

Oransje - I applikasjonen blir det brukt en oransje farge som er den blå fargen sin komplementær farge. Dette gjøres for da komplementærfarger gir gode kombinasjoner i riktige bruksområder.



Komplementær hjul.

Rød - Blir brukt i alle knapper som sletter eller krysser ut noe. Det blir for eksempel brukt i alle knapper som sletter noe, samt når man krysser seg ut av en aktivitet. Som man kan se i hjulet ovenfor, er også rød komplementærfargen til oransje.



Rød blir brukt i knapper som avslutter en aktivitet.

Designvalg

Innenfor design er det viktig å gjøre det enklest mulig for brukeren, som har vært et fokus i denne applikasjonen. Det er for eksempel brukt så lite tekst som mulig for å gi ut informasjon, med heller vært fokus på ikoner. Dette gjør applikasjonen mer språkuavhengig som er en viktig faktor.

Applikasjonen har et utseende som er i tråd med andre listeapplikasjoner i Android. Den har en *FloatingActionButton* som legger til nye studenter, tilnærmet lik kontakter i Android. Hvis man trykker på et element i en liste kommer man til en nytt vindu med mer informasjon om elementer, som også er svært vanlig innenfor listeapplikasjoner.

Alle aktiviteter har en en bakgrunnsfarge i kantene av blå nyanser som er svært behagelig å se på. I tillegg ligger alle Widgets på en grå bakgrunn som gir en svært god brukeropplevelse.

Lenker

- <https://github.com/ittianyu/BottomNavigationViewEx>