

BookMeetEat

Intro

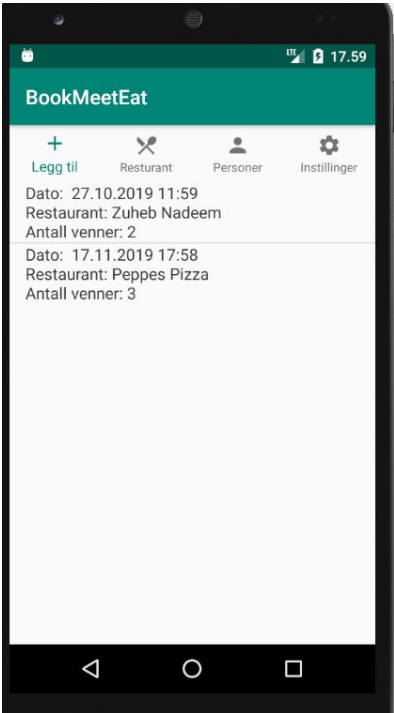
BookMeetEat er en applikasjon hvor brukeren kan velge hvilke venner han har lyst til å invitere til et spisested. Den skal både være enkel og gjøre det lettere for brukeren å sosialisere seg. Gjennom å bruke et navigasjonsbar, vil en bruker kunne lett hoppe seg gjennom de forskjellige aktivitetene.

Appen inneholder tre forskjellige hovedsider: Bestilling, Restaurant, Personer og en innstilling-side.

Appen er lett og rett fram og brukeren skal i all tid vite hva han gjør og hvor han er.

Appen er tom for data ved start. Men dette kan lett legges til. ved bruk av de ulike knappene i navigasjonsbaren.

Navigasjonsmuligheter

BESKRIVELSE	SKJERMBILDE
<h3>Hjem</h3> <p>Her kan man se en liste over bestillinger som er registrert. Her vil brukeren få fire muligheter i navigasjonsbar: Restaurant, person, innstillinger og en pluss-knapp.</p> <p>Restaurant: Navigerer deg til restaurant sin hovedside.</p> <p>Person: Navigerer deg til person sin hovedside.</p> <p>Innstillinger: Navigerer deg til innstillinger sin hovedside.</p> <p>Legg til bestillinger(+): Navigerer deg til en ny side hvor man har mulighet til å legge til nye bestillinger.</p>	<p>Forsiden, med kommende bestillinger:</p>  <p>The screenshot shows the BookMeetEat app interface on a smartphone. At the top is a green header with the app name. Below it is a navigation bar with four icons: a plus sign (labeled 'Legg til'), a fork and knife (labeled 'Restaurant'), a person icon (labeled 'Personer'), and a gear icon (labeled 'Innstillinger'). The main content area displays a list of two upcoming orders. The first order is dated 27.10.2019 11:59, from restaurant Zuheb Nadeem, with 2 friends. The second order is dated 17.11.2019 17:58, from restaurant Peppes Pizza, with 3 friends. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.</p>

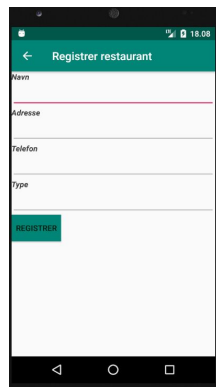
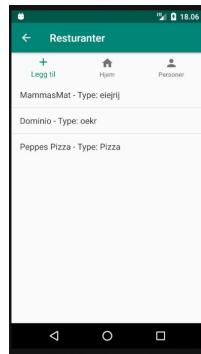
Restaurant

Her kan man se en liste over restauranter som er registrert i en liste. Her vil brukeren få tre muligheter i navigasjonsbar: Hjem, person og en pluss-knapp.

Hjem: Navigerer deg til hjemmesiden.

Person: Navigerer deg til person sin hovedside.

Legg til(+): Navigerer deg til ny side hvor du har mulighet til å legge til nye restauranter med navn, adresse, telefonnummer og type restaurant.



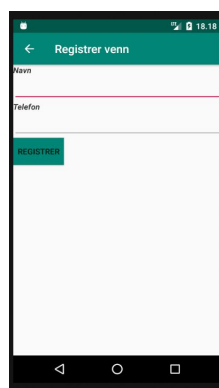
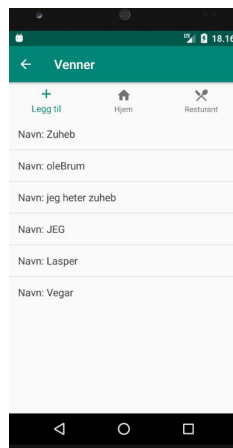
Person

Her kan man se en liste over personer som er registrert i en liste. Her vil brukeren få tre muligheter i navigasjonsbar hjem, restaurant og en pluss-knapp.

Hjem: Navigerer deg til hjemmesiden.

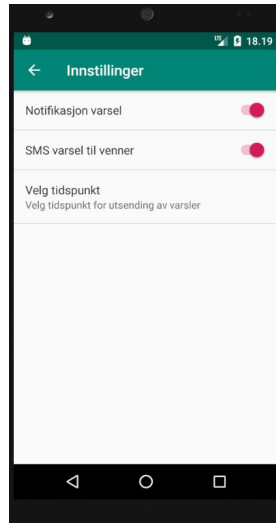
Restaurant : Navigerer deg til restaurant sin hovedside.

Legg til(+): Navigerer deg til ny side hvor du har mulighet til å legge til nye personer med navn og telefonnummer.



Innstillinger

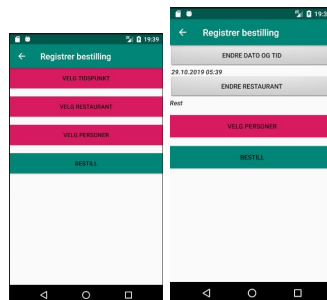
Brukeren kan tilpasse innstillingene til applikasjonen etter deres ønske. Her kan du velge om notifikasjon og SMS - varsel til vennene skal være av eller på. Samtidig kan man selv velge når både Notifikasjon og sms varsel skal sendes ut.



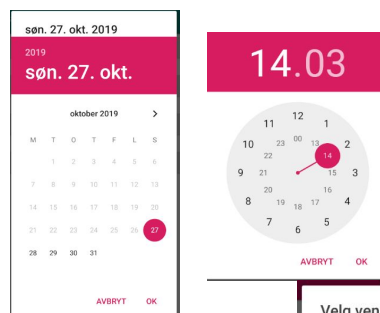
Registrere bestillinger

Her vil man kunne legge til dato og tid, restaurant og antall venner man ønsker å ta med.

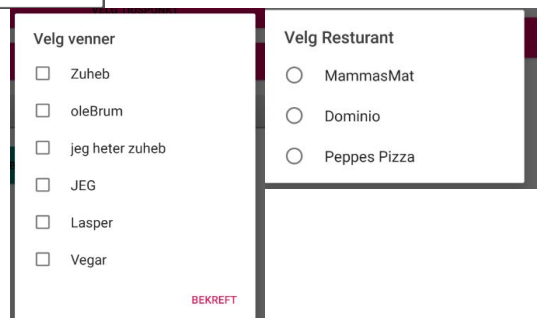
Når noe blir valgt, blir fargen på knappene endret fra rødt til grått for å symbolisere at du har valgt denne og teksten endrer seg fra "velg" til "endre".



Mulighet for å velge tid og dato:



Mulighet for å velge restaurant og venner :



Ikoner og farger

Navigasjonsbaren har ikoner som valg. Ved hjelp av disse kan brukeren navigere seg rundt i applikasjonen. Ikonene som er brukt symboliserer hvilken aktivitet man kommer til. Det har blitt brukt tekst med ikonene, i tilfelle noen brukere ikke skulle forstå ikonet. Vi har valgt å la bakgrunnsfargen være hvit fordi det er enklere å lese tekst med hvit bakgrunn.



Logikk

Applikasjonen er implementert med hensyn til developer.android.com og forelesningsnotater. Vi har tre klasser som brukes gjennom programmet Bestilling, Person og Resturant.

ButtomNavigationView

Alle sidene på applikasjonen utenom innstillinger-siden tar i bruk en navigasjonsbar. Navigasjonsbaren er lagd gjennom ButtomNavigationView som bruker app:menu for å fortelle hvilket innhold navigasjonsbaren skal ha. Det er brukt forskjellig app:menu i hver navigasjonsbar, fordi vi har valgt å endre innholdet i navigasjonsbar i forhold til hvor brukeren befinner seg.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android">
3   <item
4     android:title="Resturant"
5     android:id="@+id/hjemResturantNavBar"
6     android:icon="@drawable/ic_restaurant_menu_black_24dp" />
7
8   <item android:title="Personer"
9     android:id="@+id/hjemPersonerNavBar"
10    android:icon="@drawable/ic_person_black_24dp" />
11
12  <item android:title="Instillinger"
13    android:id="@+id/hjemInstillingNavBar"
14    android:icon="@drawable/ic_settings_black_24dp"/>
15
16  <item android:title="Legg til"
17    android:id="@+id/hjemLeggTilBestillingNavBar"
18    android:icon="@drawable/ic_add_black_24dp"/>
19 </menu>

```

Videre har vi brukt `OnNavigationItemSelectedListener` for å sjekke hva som ble trykket på i navigasjonsbaren. Brukeren blir deretter sendt videre til en ny aktivitet. Under er koden for `Navigeringsbar` i `MainActivity`, den er gjenbrukt på personer og restaurant sin hovedside, men med endringer i valgene på navigasjonsbaren.

```

//NAVIGERINGSBAR
public void NavigerBaringsbar() {
    BottomNavigationView navigationView = findViewById(R.id.HjemNavnID);

    BottomNavigationView.OnNavigationItemSelectedListener navListener =
        new BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.hjemResturantNavBar:
                        Intent restaurantNavBar = new Intent( packageContext: MainActivity.this, MainRestaurant.class);
                        startActivity(restaurantNavBar);
                        break;
                    case R.id.hjemPersonerNavBar:
                        Intent personNavBar = new Intent( packageContext: MainActivity.this, MainPerson.class);
                        startActivity(personNavBar);
                        break;
                    case R.id.hjemInstillingNavBar:
                        Intent instillingNavBar = new Intent( packageContext: MainActivity.this, Preferanser.class);
                        startActivity(instillingNavBar);
                        break;
                    case R.id.hjemLeggTilBestillingNavBar:
                        Intent leggTilNavBar = new Intent( packageContext: MainActivity.this, RegBestilling.class);
                        startActivity(leggTilNavBar);
                        break;
                }
                return false;
            }
        };
    navigationView.setOnNavigationItemSelectedListener(navListener);
}

```

ListView

I bestilling, person og restaurant er det brukt `ListView` for å liste ut verdiene fra databasen. Under kan man se kodene som er gjenbrukt på de tre sidene, endringene ligger kun i `putExtra`(hvilke verdier vi velger å ta med videre til neste aktivitet). Denne koden sørger for å åpne en ny aktivitet når vi velger noe fra `listview`'et og sender med verdiene til valgte `listview`.

```

public void ListViewListener() {
    ListAllePersoner.setOnItemClickListener((adapterView, view, i, 1) -> {

        Person person = (Person)adapterView.getAdapter().getItem(i);
        int verdi = person.getId();
        String navn = person.getNavn(); //vil gi oss den verdien vi trykker på
        String tlf = person.getTelefon();

        //Starter ny intent, når en av navnene trykkes i ListViewet
        Intent endrePersonen = new Intent( packageContext: MainPerson.this, EndrePerson.class);
        //velger å sende med verdier til den nye intent
        endrePersonen.putExtra( name: "ID",verdi);
        endrePersonen.putExtra( name: "Navn",navn);
        endrePersonen.putExtra( name: "Telefon",tlf);
        startActivity(endrePersonen);

    });
}

```

Når vi går til en ny aktivitet, blir de verdiene vi sendte med lagt tilbake i input-feltene. Da kan brukeren få en oversikt over hva som allerede ligger registrert i valgte listview'et og deretter endre etter hans ønsker.

```

//henter verdiene vi valgte å sende med i intent fra forrige java-klasse(Mainactivity)
Intent hentIntentVerdier = getIntent();
valgtID = hentIntentVerdier.getIntExtra( name: "ID", defaultValue: 0); //henter ut ID som ble valgt
navnFeltTxt.setText(hentIntentVerdier.getStringExtra( name: "Navn"));
telefonFeltTxt.setText(hentIntentVerdier.getStringExtra( name: "Telefon"));

```

Registering, endringer og sletting til databasen

Brukeren har i applikasjon mulighet til å registrere og slette personer, restauranter og bestillinger. Det er samtidig mulig å endre verdiene i personer og restauranter. Før en bruker kan registrere verdier til databasen, blir verdiene i input-felter sjekke opp med regex. Vi sørger dermed at feil eller tomme felt ikke lagres til databasen. Databasen vår er i 3. normalform og vi har 4 tabeller:

Personer: Id, Navn, Tlf

Restaurant: Id, Navn, Adresse, Tlf

Bestilling: Id, Dato, Restaurant id

PersonDeltakelse: Person Id , Bestilling Id

Det er de samme kodene som brukes både ved registrering av restaurant og bestillingen, forskjellen ligger kun i at man legger til en annen database tabell og at input-feltene valideres annerledes tilpasset forventet verdi. Her sjekker vi om input-feltene er som vi forventer (f.eks. kun 8 siffer i telefonnummer) og lar brukeren registrere seg. Dersom input-feltene ikke stemmer med forventet verdi vil brukeren få opp en melding om dette. Da vil brukeren bli værende igjen på samme side og ingenting vil bli registrert.

```
//Registrer en Person-klasse til databasen
public void RegistrerPerson(View view) {

    String innNavnet = navnTxt.getText().toString().trim();
    String innTlfnr = tlfTxt.getText().toString().trim();

    //Regex håndtere input-felter
    Pattern navnSjekk = Pattern.compile("[a-zAÅå A-Zåå]{3,30}");
    Pattern tlfSjekk = Pattern.compile("[0-9]{8}");

    if(navnSjekk.matcher(innNavnet).matches() && tlfSjekk.matcher(innTlfnr).matches() ) {
        Person person = new Person(navnTxt.getText().toString(),tlfTxt.getText().toString());
        DB.LeggTilPerson(person);
        finish();
    }
    else {
        Toast.makeText( context: RegPerson.this, text: "Du må fylle ut feltene riktig",Toast.LENGTH_SHORT).show();
    }
}
}
```

Ved endring i restaurant, vil brukeren se tidligere verdier i input-feltene. Hvis input-feltene stemmer med forventet verdi(regex valideringen), vil det være mulig å registrere endringen. Hvis ikke lages en feilmelding til skjermen for brukeren. Dette gjelder også ved endring av personer.

```
public void endreRestaurant(View view) {

    String innNavnet = navnFeltTxt.getText().toString().trim();
    String innAdresse = adresseFeltTxt.getText().toString().trim();
    String innTlfnr = telefonFeltTxt.getText().toString().trim();
    String innType = typeFeltTxt.getText().toString().trim();

    //Regex håndtere input-felter
    Pattern navnSjekk = Pattern.compile("[a-zAÅå A-Zåå]{3,30}");
    Pattern tlfSjekk = Pattern.compile("[0-9]{8}");
    Pattern adresseSjekk = Pattern.compile("[a-zAÅå A-Zåå 0-9]{3,50}");
    Pattern typeSjekk = Pattern.compile("[a-zAÅå A-Zåå]{3,30}");

    if(navnSjekk.matcher(innNavnet).matches() && tlfSjekk.matcher(innTlfnr).matches()
        && adresseSjekk.matcher(innAdresse).matches() && typeSjekk.matcher(innType).matches() ) {
        Restaurant restaurant = new Restaurant(
            Integer.valueOf(valgtID),
            navnFeltTxt.getText().toString(),
            adresseFeltTxt.getText().toString(),
            telefonFeltTxt.getText().toString(),
            typeFeltTxt.getText().toString());

        DB.oppdaterRestaurant(restaurant); //oppdaterer databasen med den verdien som blir skrevet inn til den ID som
        finish();
    }
    else {
        Toast.makeText( context: EndreRestaurant.this, text: "Du må fylle ut feltene riktig",Toast.LENGTH_SHORT).show();
    }
}
}
```

Sletter valgt verdi fra listviewet. Verdiene vises i input-feltene før de slettes. Dette gjelder også ved slett av personer og bestillinger.


```

public void slettRestaurant(View view) {
    DB.SlettRestaurant(Integer.valueOf(valgtID));
    finish();
}

```

Notifikasjon

I vår app gir vi notifikasjon til brukeren dersom en bruker har en reservasjon neste dag. Notifikasjonen sendes ut som en varsel i form av SMS til hans venner og som en notifikasjon på hans egen mobil. Under innstillinger har brukeren mulighet til å slå notifikasjonstjenesten og SMS tjenesten av og på. Brukeren har også mulighet til å velge tid utsendelse for SMS. Om SMS og notifikasjons varsel er av eller på sjekkes i innstillinger, og deres verdier blir lagret og sendt ut til aktivitetene gjennom SharedPreferences.

Denne koden sjekker om notifikasjon varsel er på. Dersom den er på sjekkes det om det ligger noen reservasjoner for neste dag, hvis det gjør det sendes det ut en notifikasjon til bruker sin mobil. Innholdet i notifikasjonen vil fortelle deg at du har en reservasjon i morgen og ved trykk på notifikasjonen blir du sendt til hjemmesiden til applikasjonen. Denne koden sjekker også om SMS varsel er av eller på. Dersom SMS varsel er på og brukeren har valgt venner ved bestilling av et bord vil SMS sendes til valgte venner med en melding som kommer fra SharedPreferences.

```

if(notifikasjonBool){
    //Lager notifikasjon for når AlarmManager settes
    NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    Intent i = new Intent( packageContext, this, MainActivity.class);
    PendingIntent pendingIntent = PendingIntent.getActivity( context, this, requestCode: 0,i, flags: 0);

    if(liste != null && liste.size() != 0){
        NotificationCompat.Builder( context, this)
            .setContentTitle("Reservasjon i morgen!")
            .setContentText("Du har " + liste.size() + " bestilling(er) som nærmer seg!")
            .setSmallIcon(R.mipmap.ic_launcher)
            .setContentIntent(pendingIntent)
            .build();

        notification.Flags = Notification.FLAG_AUTO_CANCEL;
        notificationManager.notify( id: 0,notification);
    }else{
        Log.d( tag: "Notifikasjon", msg: "INGEN BESTILLINGER SOM SKAL VARSLES");
    }
}

if (smsBool){
    if(liste != null && liste.size() != 0){
        for (Bestilling b: liste) {
            if(b.Deltakelse != null && b.Deltakelse.size() != 0){
                for (Person p: b.Deltakelse) {
                    SendSMS(p.Telefon,message);
                }
            }
        }
    }
}

```

Broadcast og Service

Applikasjonen bruker Broadcast og Service til å gi ut notifikasjon og sende ut Sms en gang daglig , med brukerens valgte tidspunkt.

Alarm manageren vil funke selv om applikasjonen eller telefonen slås av. Dette gjør vi ved å gi broadcast en action:

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

```
<receiver android:name=".Service.MinBroadCast">
    <intent-filter>
        <action android:name="com.example.mappe2.Service.MinBroadCast" />
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
```

Ellers vil servicen som daglig sjekker om det er bestillinger sørge for å gjøre jobben sin, så lenge brukeren ikke går inn på innstillinger og slår av både SMS og Notifikasjons-varslers.

```
public void serviceManager(){
    boolean smsBool = getSharedPreferences( name: "PREFERENCE", MODE_PRIVATE).getBoolean( s: "ValgtSMS", b: true);
    boolean notifikasjonBool = getSharedPreferences( name: "PREFERENCE", MODE_PRIVATE).getBoolean( s: "ValgtNotifikasjon");

    if(smsBool || notifikasjonBool){
        Intent i = new Intent();
        i.setAction("com.example.mappe2.Service.MinBroadCast");
        sendBroadcast(i);
    }else {
        Intent i = new Intent( packageContext: this, MinService.class);
        PendingIntent pIntent = PendingIntent.getService( context: this, requestCode: 0,i, flags: 0);

        AlarmManager alarm = (AlarmManager) getSystemService(Context.ALARM_SERVICE);

        if(alarm != null){
            alarm.cancel(pIntent);
            stopService(i);
            Log.d( tag: "StopService: ", msg: "Stoppet service - SMS: " + smsBool + " og Notifikasjon: " + notifikasjonB
        }
    }
}
```

Da stopper vi både Alarm Manager og Service.

ContentProvider

```
<provider
    android:authorities="com.example.mappe2"
    android:name="com.example.mappe2.DBHandler.ContentProvider"
    android:enabled="true"
    android:exported="true"
    android:readPermission = "com.example.mappe2.permission"
    android:writePermission="com.example.mappe2.permission">

</provider>
```

Vi har tatt med content provider slik at appen vår skal ha mulighet til å dele data med andre apper. Først gir vi tilgang til å dele data gjennom AndroidManifest.xml. Over kan man se at vi lager en provider i form av en klasse som holder styr på hva andre apper kan gjøre med vår data.

Content provideren er lagt til Mappen som cp_BookMeetEat. I denne er det mulig å både legge til og liste ut alle restauranter som finnes i databasen i hovedprosjektet BookMeetEat.

