Sprawozdanie z Projektu z przedmiotu Aplikacje Mobilne

Prowadzący: por. mgr inż. Michał Sobolewski

Przygotował: Mateusz Gajda

Grupa: WCY20IJ1S1

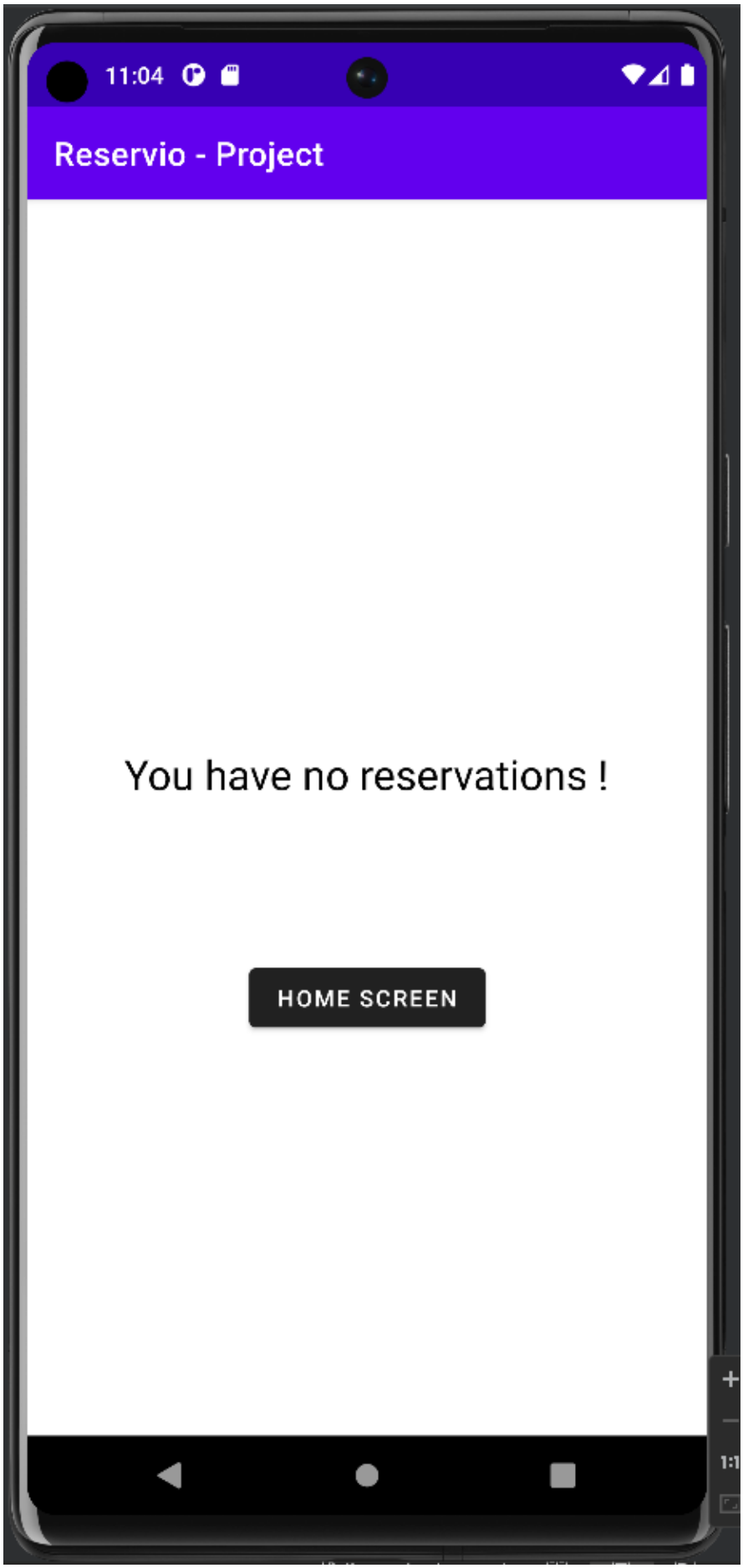
Krótki Opis Aplikacji:

Reservio to aplikacja mobilna, która umożliwia tworzenie rezerwacji w restauracjach znajdujących się w pobliżu użytkownika po podaniu zasięgu w którym aplikacja ma szukać tych restauracji. Restauracje pobierane są z API, a rezerwacje są zapisywane w bazie danych. Użytkownik po wybraniu restauracji ustawia sobie datę na kiedy chce zarezerwować stolik oraz godzinę. Dodatkowo wybiera on numer stolika oraz na ile osób będzie zarezerwowany ten stolik. Po zarezerwowaniu użytkownik może przeglądać swoje rezerwacje które są posortowane od najwcześniejszych do najpóźniejszych. Oprócz tego po kliknięciu w taką rezerwacje może ją edytować lub usuwać. Cała aplikacja została napisana w języku Java w Android Studio.

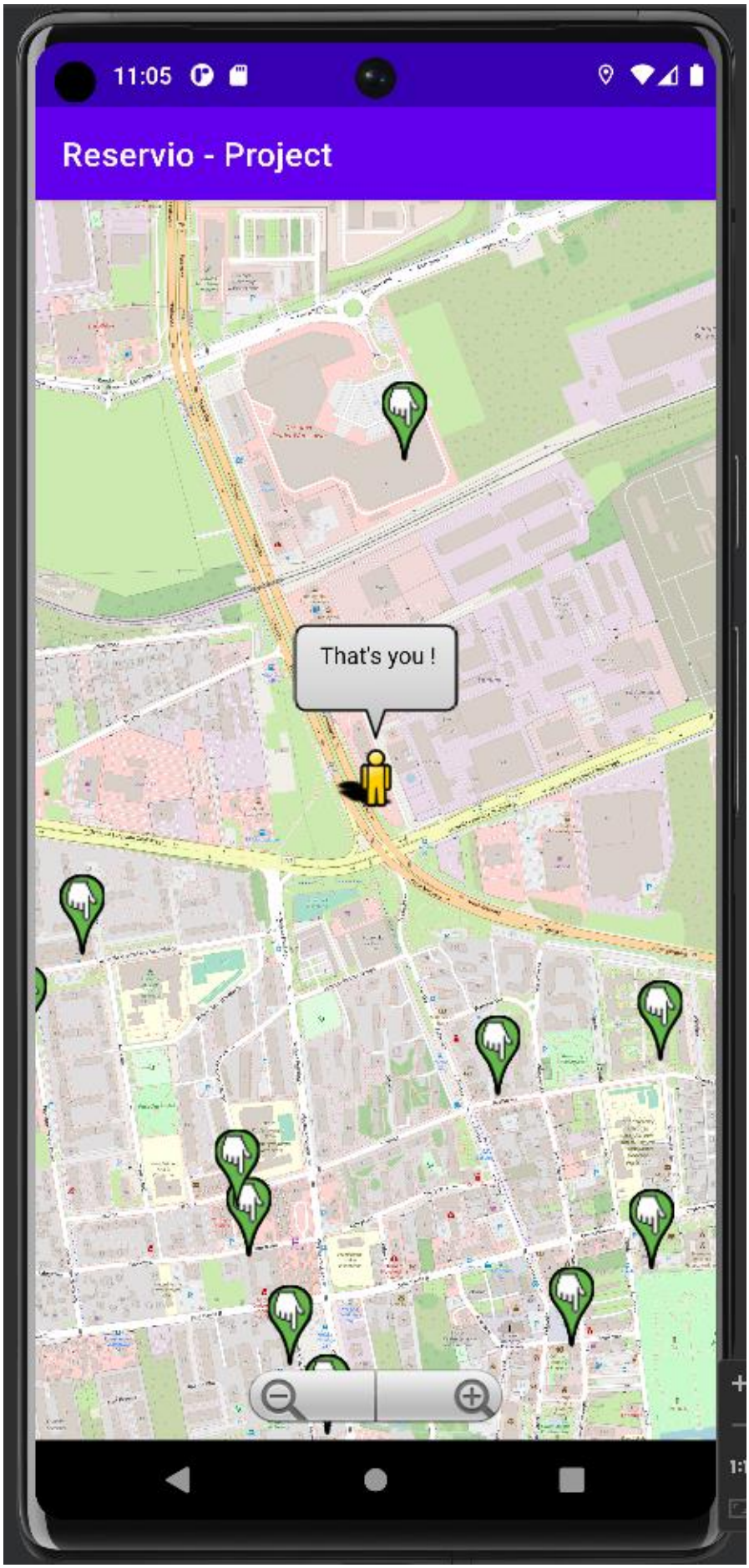
Opis wykorzystanych Technologii:

- XML - język znaczników, który jest często używany do tworzenia układów interfejsu użytkownika w aplikacjach mobilnych
- SQLite - to niewielka i szybka baza danych, która jest zintegrowana z systemem Android i można jej używać bez instalowania dodatkowych bibliotek. Jest często używana w aplikacjach mobilnych do przechowywania danych, takich jak ustawienia aplikacji, informacje użytkownika, a także do przechowywania danych offline
- Android Jetpack - to zestaw bibliotek, narzędzi i wskazówek projektowych, które pomagają w tworzeniu aplikacji mobilnych
- Android SDK - zestaw narzędzi programistycznych, które umożliwiają tworzenie aplikacji mobilnych na platformę Android. Zawiera biblioteki, narzędzia programistyczne i emulator, który pozwala na uruchamianie i testowanie aplikacji na różnych urządzeniach z Androidem
- osmdroid - biblioteka do tworzenia map w aplikacjach mobilnych na platformę Android. Umożliwia wyświetlanie map z różnych źródeł, np. OpenStreetMap, dodawanie znaczników na mapie i interakcję z użytkownikiem
- JUnit to biblioteka do testowania jednostkowego w języku Java. Jest wykorzystywana do pisania testów jednostkowych w aplikacjach Androidowych
- Dagger - biblioteka do wstrzykiwania zależności (**dependency injection**) w aplikacjach Java i Androidowych. Pozwala na łatwe zarządzanie zależnościami między komponentami aplikacji
- JSON - format danych do przesyłania informacji pomiędzy aplikacjami

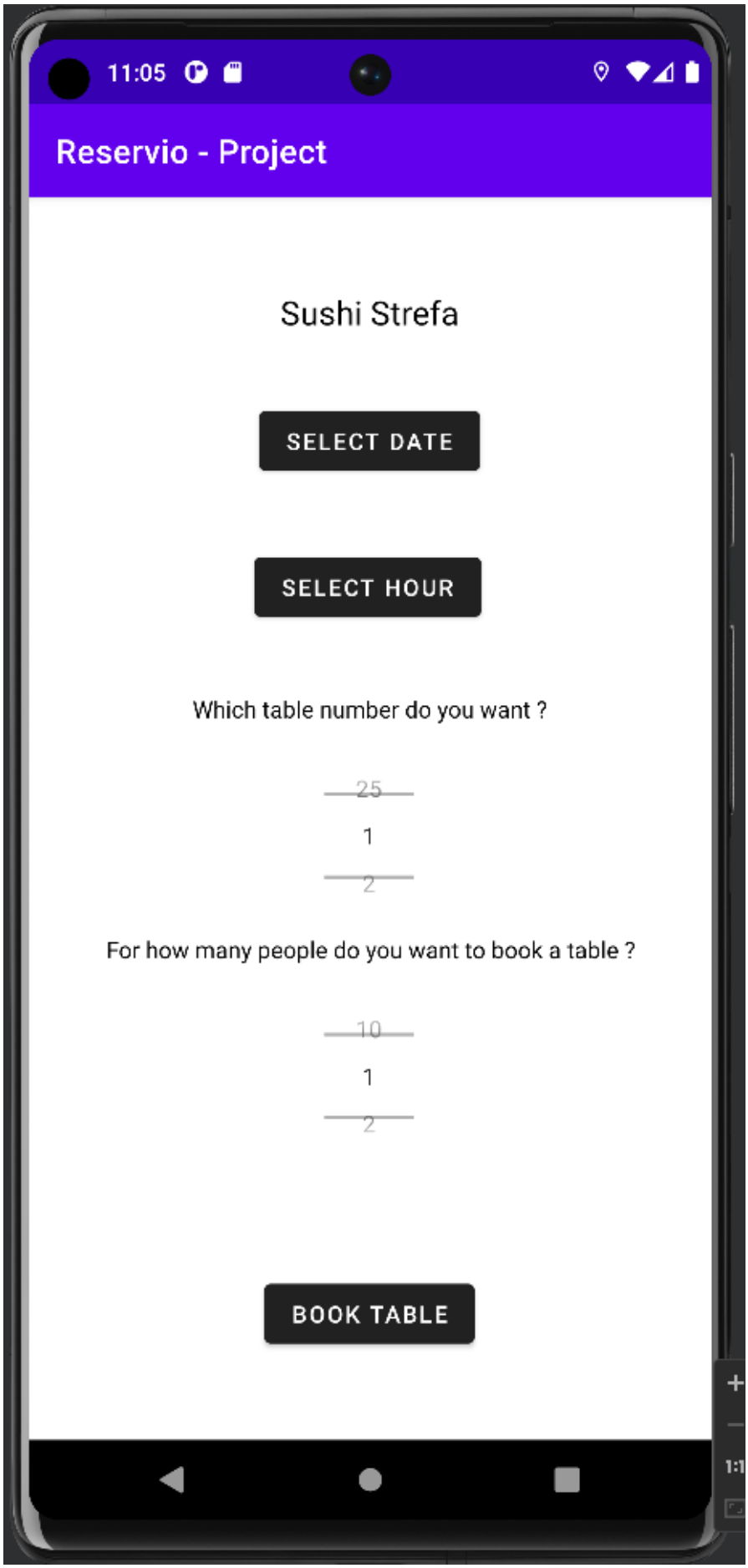
Screeny z Aplikacji:

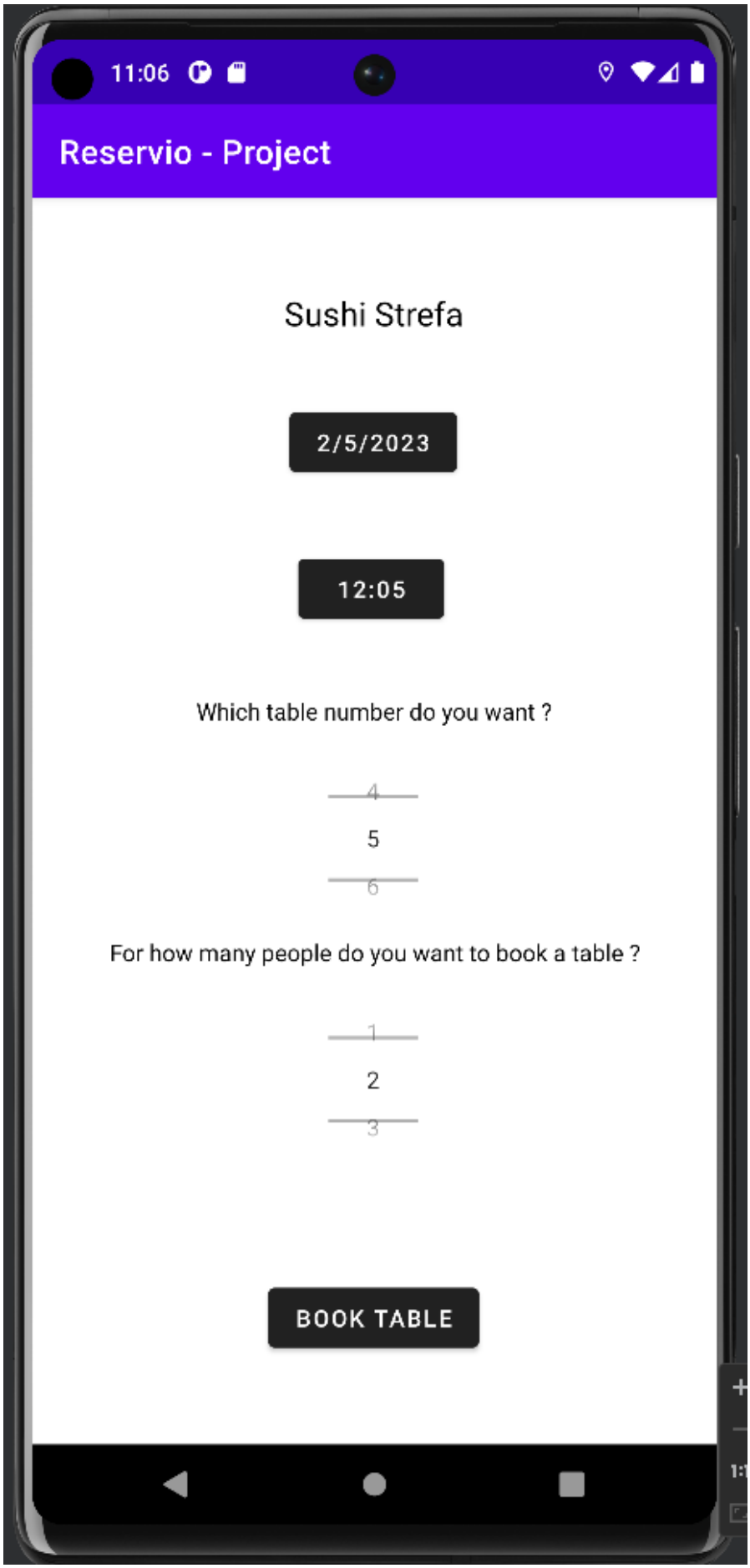

11:04
Reservio - Project
You have no reservations !
HOME SCREEN

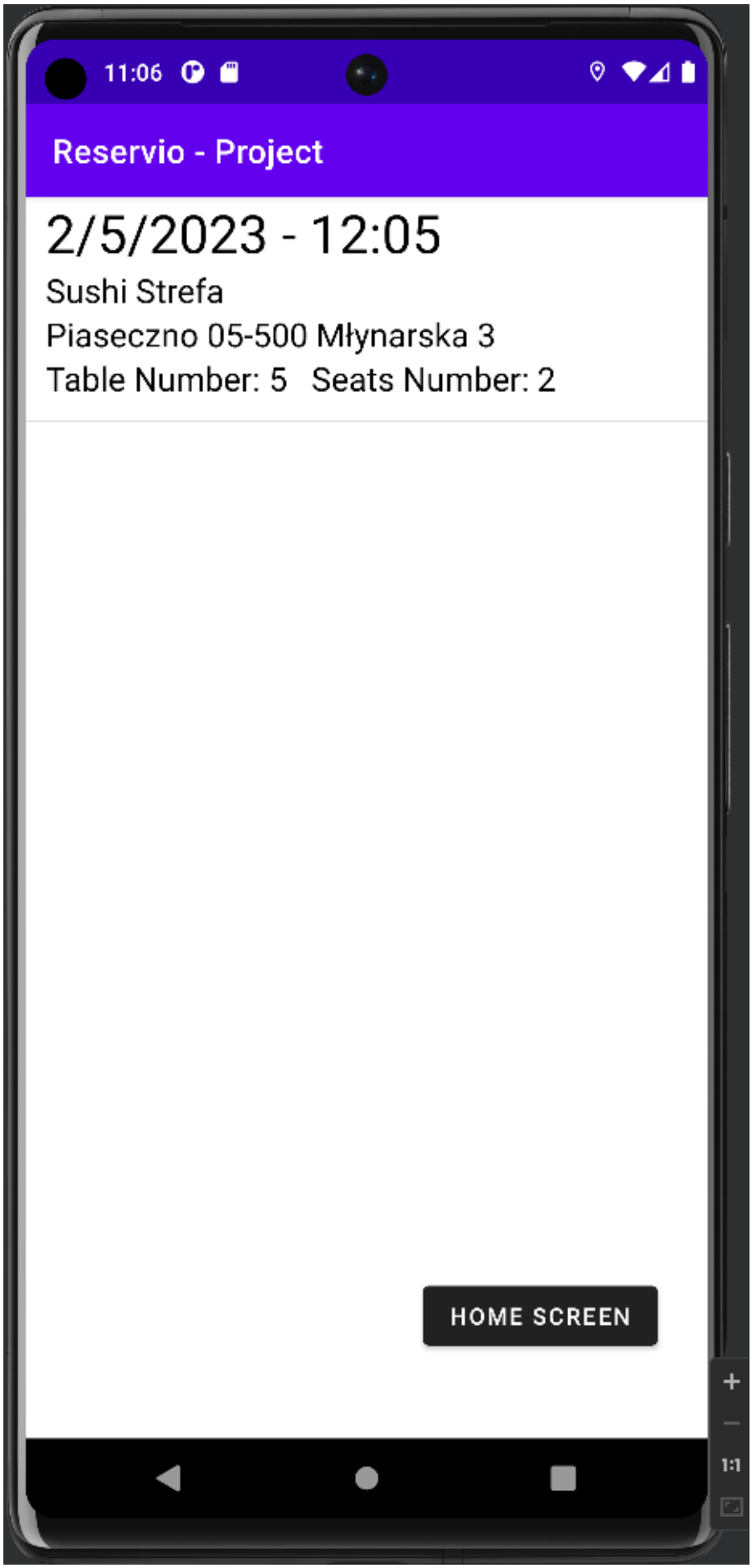

11:04
Reservio - Project
Specify The Range In Which To Search For Restaurants:

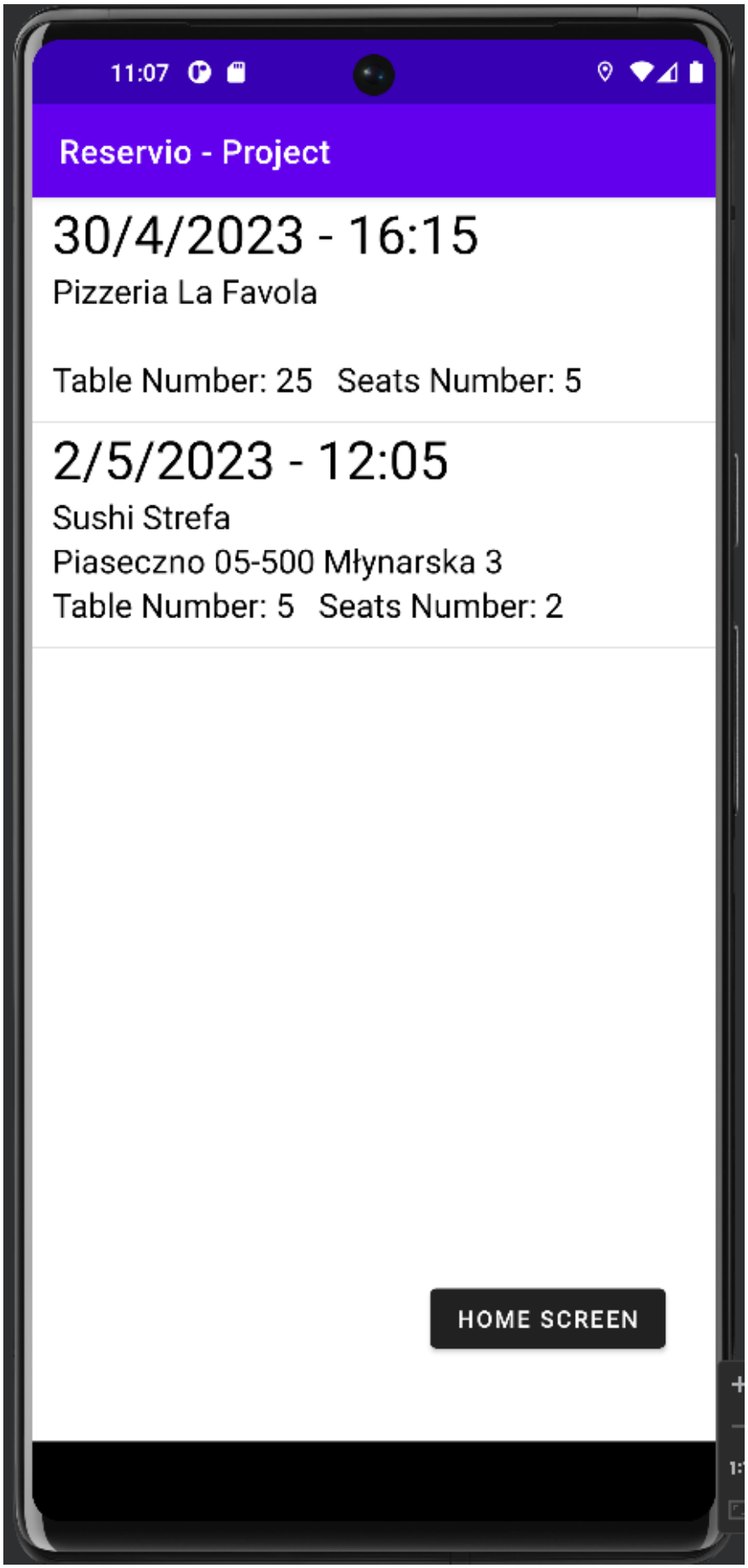

11:05
Reservio - Project
That's you !

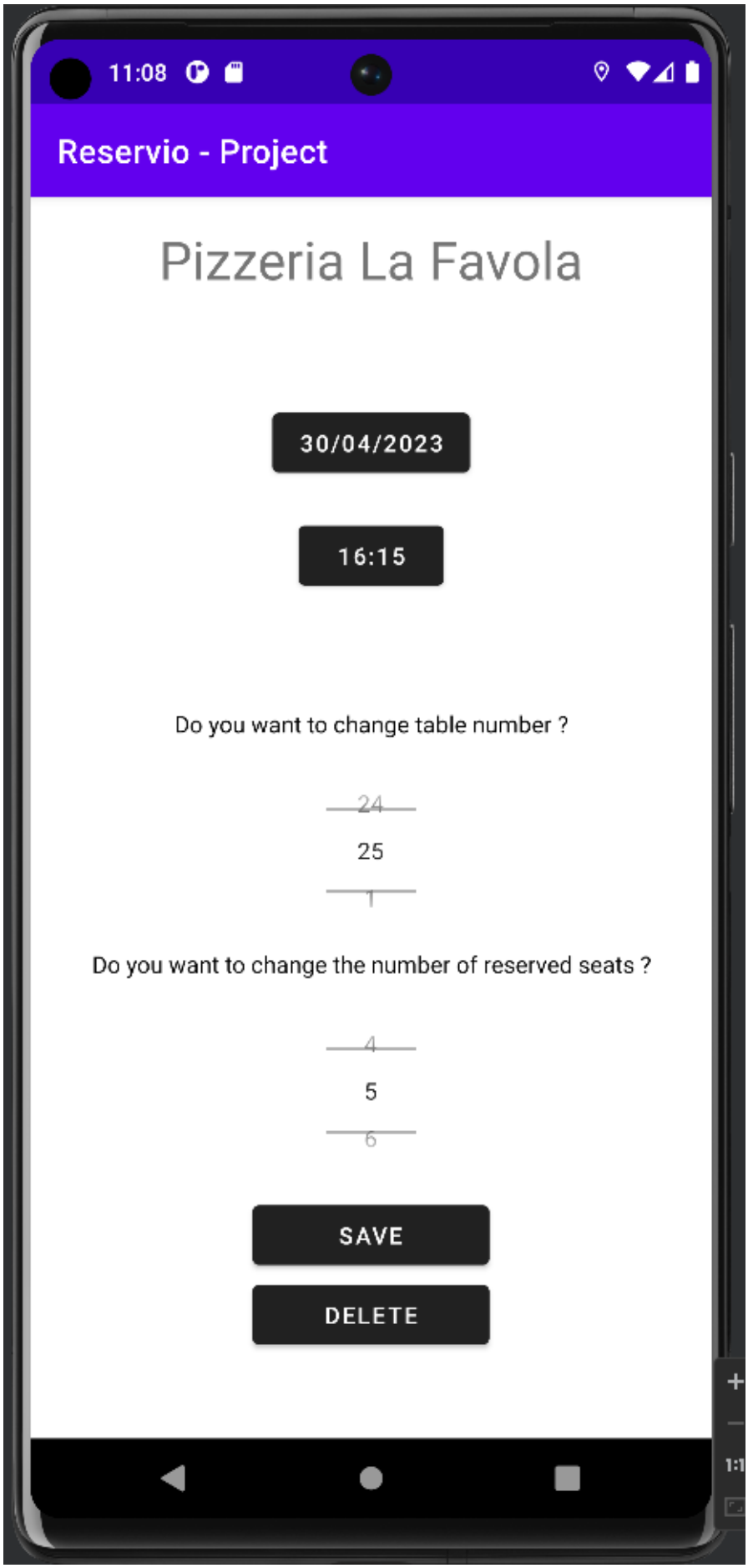

11:05
Reservio - Project
Sushi Strefa
sushi
RESERVE
1:1

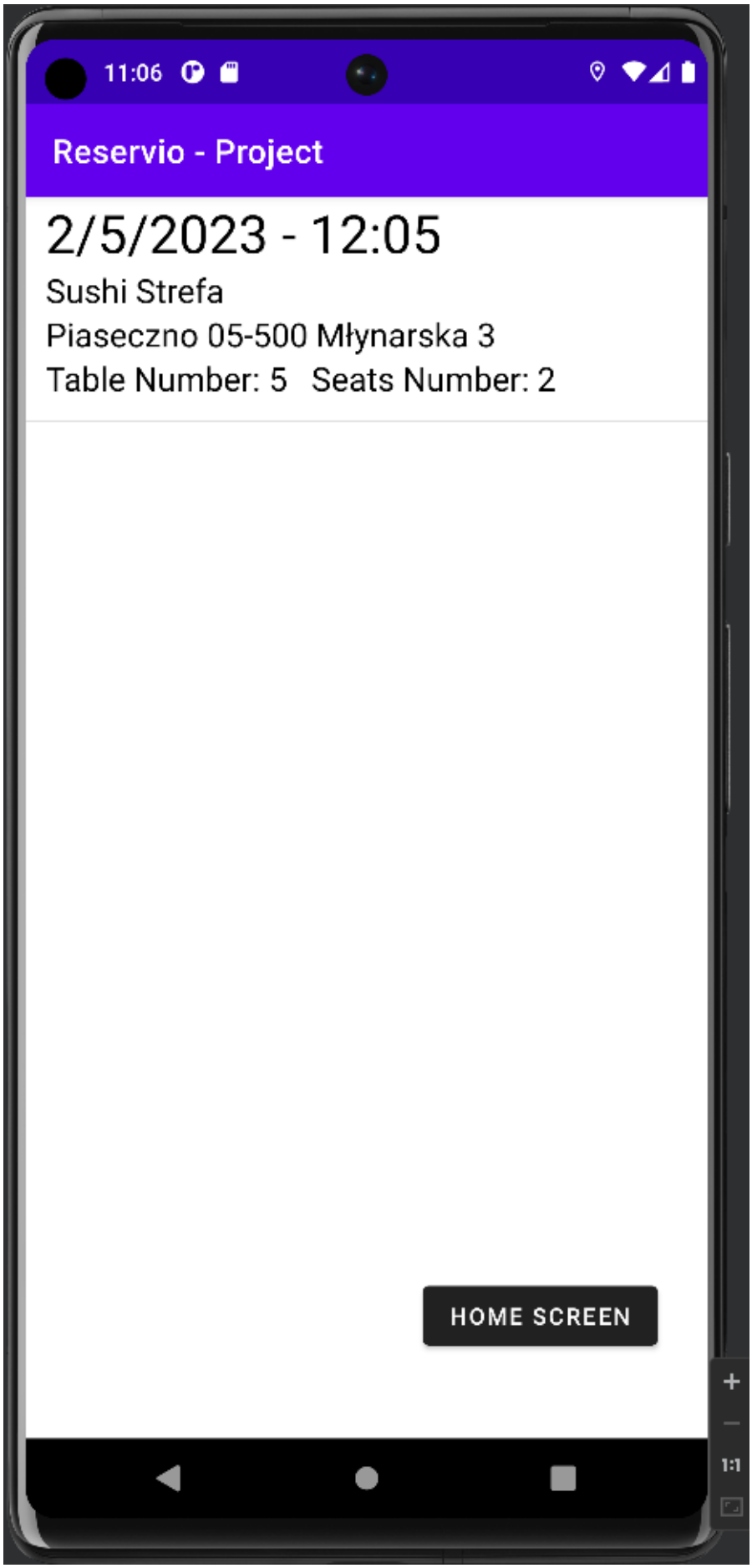

11:05
Reservio - Project
Sushi Strefa
SELECT DATE
SELECT HOUR
Which table number do you want ?
25
1
2
For how many people do you want to book a table ?
10
1
2
BOOK TABLE

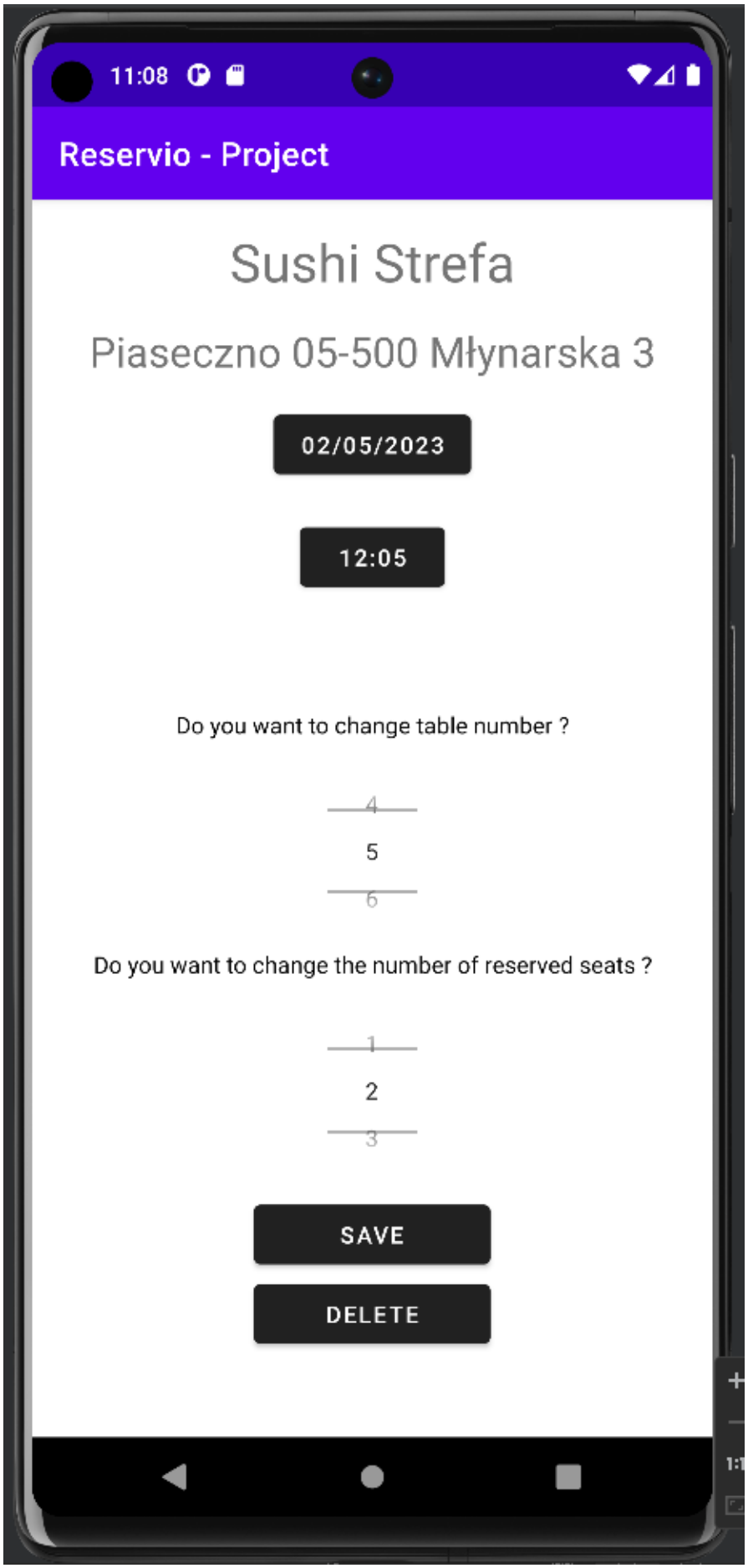

11:06
Reservio - Project
Sushi Strefa
2/5/2023
12:05
Which table number do you want ?
4
5
6
For how many people do you want to book a table ?
1
2
3
BOOK TABLE

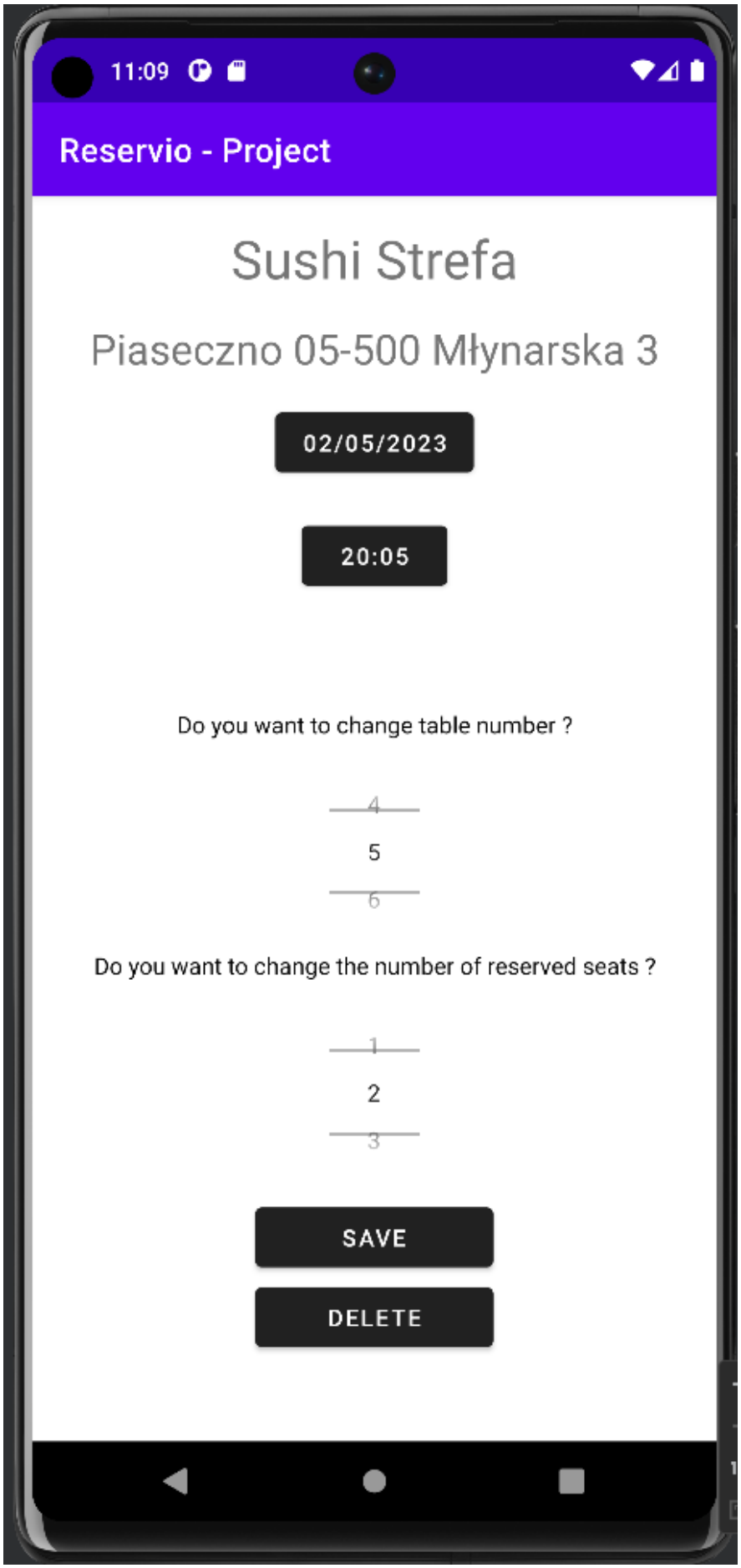

11:06
Reservio - Project
2/5/2023 - 12:05
Sushi Strefa
Piaseczno 05-500 Młynarska 3
Table Number: 5 Seats Number: 2
HOME SCREEN
1:1

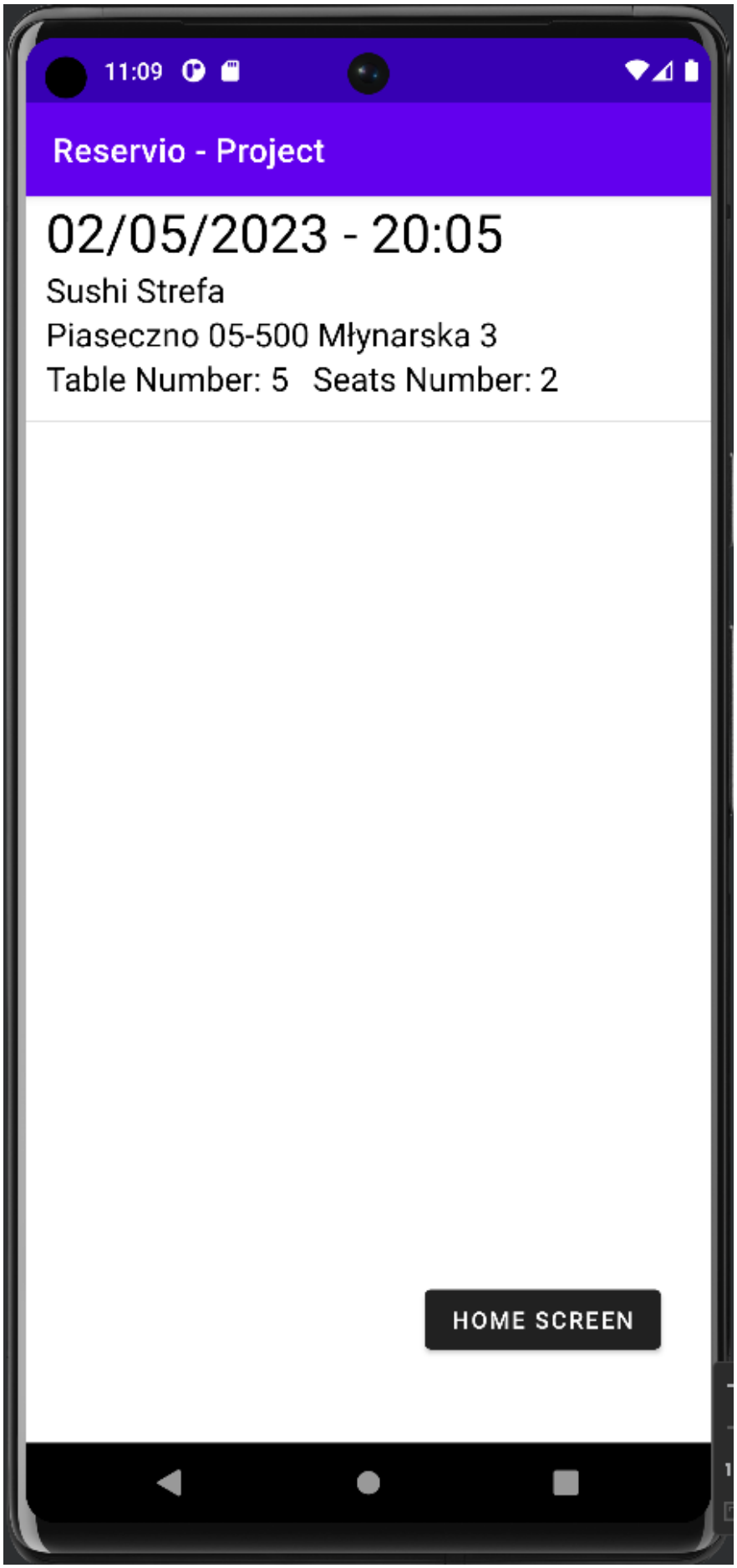

11:07
Reservio - Project
30/4/2023 - 16:15
Pizzeria La Favola
Table Number: 25 Seats Number: 5
2/5/2023 - 12:05
Sushi Strefa
Piaseczno 05-500 Młynarska 3
Table Number: 5 Seats Number: 2
HOME SCREEN


11:08
Reservio - Project
Pizzeria La Favola
30/04/2023
16:15
Do you want to change table number ?
24
25
1
Do you want to change the number of reserved seats ?
4
5
6
SAVE
DELETE


11:06
Reservio - Project
2/5/2023 - 12:05
Sushi Strefa
Piaseczno 05-500 Młynarska 3
Table Number: 5 Seats Number: 2
HOME SCREEN
1:1


11:08
Reservio - Project
Sushi Strefa
Piaseczno 05-500 Młynarska 3
02/05/2023
12:05
Do you want to change table number ?
4
5
6
Do you want to change the number of reserved seats ?
1
2
3
SAVE
DELETE


11:09
Reservio - Project
Sushi Strefa
Piaseczno 05-500 Młynarska 3
02/05/2023
20:05
Do you want to change table number ?
4
5
6
Do you want to change the number of reserved seats ?
1
2
3
SAVE
DELETE


11:09
Reservio - Project
02/05/2023 - 20:05
Sushi Strefa
Piaseczno 05-500 Młynarska 3
Table Number: 5 Seats Number: 2
HOME SCREEN

Wykorzystane biblioteki i sposób ich zaimportowania do Projektu:

- AndroidX AppCompat - biblioteka, która dostarcza zgodność z wersją platformy Android do starszych wersji Androida
- Material Design - biblioteka, która dostarcza komponenty UI w stylu Material Design firmy Google
- ConstraintLayout - biblioteka, która umożliwia tworzenie elastycznych układów interfejsu użytkownika
- JUnit - biblioteka, która umożliwia pisanie testów jednostkowych w języku Java
- Espresso - biblioteka, która umożliwia pisanie testów funkcjonalnych interfejsu użytkownika w języku Java
- OsmDroid - biblioteka, która umożliwia wykorzystanie map OpenStreetMap w aplikacji
- OkHttp - biblioteka, która umożliwia wykonywanie zapytań sieciowych http
- Dagger - biblioteka, która umożliwia wstrzykiwanie zależności w języku Java

Wszystkie biblioteki zostały dodane do pliku build.gradle jako dependencies i zaimportowane do Projektu. Inne biblioteki które mogły być użyte w projekcie zostały zaimportowane poprzez zwykle wywołanie import w klasach Java.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.google.android.gms:play-services-maps:18.1.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    implementation 'org.osmdroid:osmdroid-android:6.1.2'
    implementation 'com.squareup.okhttp3:okhttp:4.9.1'
    implementation 'com.google.dagger:dagger:2.28.3'
    annotationProcessor 'com.google.dagger:dagger-compiler:2.28.3'
}
```

Komponenty/Strony na których się wzorowano w implementacji:

- https://www.youtube.com/watch?v=fis26HvvDII&ab_channel=freeCodeCamp.org
- https://www.youtube.com/watch?v=LMk4ss8XxWc&ab_channel=AMJayshri
- https://developer.android.com/studio/write/resource-manager
- https://www.geeksforgeeks.org/how-to-change-text-color-of-toolbar-title-in-an-android-app/
- https://stackoverflow.com/questions/64715253/cant-change-buttons-background-color-in-xml-android-studio
- https://www.youtube.com/watch?v=RKXnIrFmmYA&ab_channel=CodingWithMitch
- https://www.youtube.com/watch?v=V0AETAjxqLI&ab_channel=John%27sAndroidStudioTutorials
- https://stackoverflow.com/questions/34822485/how-to-set-visibility-of-textview
- https://stackoverflow.com/questions/5308200/clear-text-in-edittext-when-entered

Dzięki tym stronom i wybranym filmom udało mi się obeznać z Android Studio i tym jak zaimplementować niektóre funkcje obsługujące pliki xml oraz jak zmieniać parametry elementów tych plików.

- https://help.openstreetmap.org/questions/63330/how-to-retrieve-only-restaurants-and-food-stores-in-an-area
- https://www.youtube.com/watch?v=rmIhGPy8rSY&ab_channel=TheArtOfCoding
- https://stackoverflow.com/questions/7148559/get-a-list-of-places-streets-etc-from-openstreetmap
- https://stackoverflow.com/questions/37661811/empty-open-street-map-with-osmdroid-android-studio
- https://gis.stackexchange.com/questions/342367/adding-label-to-openstreetmap
- https://help.openstreetmap.org/questions/7019/how-to-put-a-pin-at-the-map
- https://gis.stackexchange.com/questions/34817/openlayers-display-a-markers-popup-using-a-button-out-of-the-map
- https://sanaebadi97.medium.com/learn-how-to-work-with-osm-map-in-android-app-ac42f933cbd3
- https://help.openstreetmap.org/questions/76673/how-to-add-text-labels-to-point-markers-in-openstreetmap
- https://stackoverflow.com/questions/56111207/display-marker-with-labels-in-osm
- https://gis.stackexchange.com/questions/376393/labels-under-markers
- https://dot-net-web-developer-bristol.blogspot.com/2014/02/markers-popups-and-labels-with-leaflet.html
- https://stackoverflow.com/questions/23108709/show-marker-details-with-image-onclick-marker-openstreetmap
- https://leafletjs.com/examples/layers-control/
- https://gis.stackexchange.com/questions/34817/openlayers-display-a-markers-popup-using-a-button-out-of-the-map
- https://gis.stackexchange.com/questions/447287/attach-click-function-to-the-button-inside-a-popup-in-leaflet
- https://medium.com/@nargessmi87/how-to-customize-the-openstreetmap-marker-icon-and-binding-popup-ab2254bddec2

Dzięki tym stronom udało mi się utworzyć scene z OpenStreetMap oraz pobrać przy użyciu OpenStreetMap API nazwy i dane dotyczące restauracji. Dodatkowo dzięki temu udało mi się stworzyć markery na mapie i popupy zgodnie ze swoją wizją.

- https://www.youtube.com/watch?v=c6c1giRekB4&ab_channel=CodeWithCal
- https://www.youtube.com/watch?v=E1LSY3g-CtY&ab_channel=AndroidCoding
- https://www.youtube.com/watch?v=dWq5CJDBDVE&ab_channel=PracticalCoding
- https://developer.android.com/reference/android/widget/NumberPicker
- https://www.youtube.com/watch?v=5dkrJ6jaK6s&ab_channel=CodingwithDev
- https://tutorialwing.com/android-numberpicker-tutorial-with-example/
- https://www.youtube.com/watch?v=gSkbeIGZhLM&ab_channel=CodeWithCal
- https://medium.com/@sc71/android-numberpickers-3ef535c45487

Dzięki tym stronom oraz filmom udało mi się wizualnie przedstawić tworzenie swoich rezerwacji stolików i w kodzie utworzyć odpowiednie funkcje które by przesyłały wybrane wartości przez użytkownika.

- https://stackoverflow.com/questions/10086053/comparing-hours-in-java
- https://www.tutorialspoint.com/how-to-compare-two-dates-in-java
- https://stackoverflow.com/questions/2592501/how-to-compare-dates-in-java

Dzięki tym stronom udało mi się rozwiązać problem porównywania różnych typów dat.

- https://www.geeksforgeeks.org/how-to-create-and-add-data-to-sqlite-database-in-android/
- https://www.youtube.com/watch?v=PCFcGpE2Fm8&ab_channel=SandipBhattacharya
- https://www.youtube.com/watch?v=PgWM9ACLIu4&ab_channel=SandipBhattacharya
- https://www.youtube.com/watch?v=VQEmj7Zos6Y&ab_channel=SandipBhattacharya
- https://www.youtube.com/watch?v=ow0Fj0xhjR0&list=PLGY_UftsHsIZDnXQk4BLUb-voVO4hQMmm&ab_channel=SandipBhattacharya
- https://www.youtube.com/watch?v=Ibe6JRvjnY4&list=PLGY_UftsHsIZDnXQk4BLUb-voVO4hQMmm&index=5&ab_channel=SandipBhattacharya
- https://www.geeksforgeeks.org/how-to-update-data-to-sqlite-database-in-android/
- https://stackoverflow.com/questions/8434819/android-sqlite-auto-increment
- https://www.sqlitetutorial.net/sqlite-autoincrement/
- https://www.geeksforgeeks.org/how-to-delete-data-in-sqlite-database-in-android/
- https://learnsql.com/cookbook/how-to-order-by-date-in-sqlite/
- https://stackoverflow.com/questions/29649004/sqlite-sorting-by-date-column
- https://www.b4x.com/android/forum/threads/sqlite-order-by-date.113350/
- https://stackoverflow.com/questions/32051477/android-studio-button-positioning

Dzięki tym stronom i fimom udało mi się nauczyć i wykorzystać bazę danych SQLite w swojej aplikacji i obsługuje wszystkie funkcję bazy danych CRUD.

- https://developer.android.com/training/dependency-injection/dagger-android

Przy pomocy tej strony udało mi się zastosować Dependency Injection w swoim projekcie

- https://www.youtube.com/watch?v=zZoboXqsCNw&ab_channel=Randomcode
- https://www.digitalocean.com/community/tutorials/java-httpurlconnection-example-java-http-request-get-post
- https://rapidapi.com/blog/how-to-use-an-api-with-java/
- https://docs.oracle.com/javaee/7/api/javax/json/JsonObject.html
- https://www.baeldung.com/java-org-json

Strony te pokazały mi jak połączyć się z API i jak pobrać dane JSON z takiego API i wybrać interesujące mnie dane

- https://stackoverflow.com/questions/1513485/how-do-i-get-the-current-gps-location-programmatically-in-android
- https://stackoverflow.com/questions/47528006/how-to-set-the-location-manually-in-android-studio-emulator

Dzięki tym stronom udało mi się zastosować w swojej aplikacji wykorzystanie sensora urządzenia mobilnego a dokładnie GPS i jak zmieniać położenie w emulatorze.

Score Card:

- ✓ Baza danych – CRUD

W aplikacji została wykorzystana baza danych SQLite i zaimplementowane zostały metody takie jak Create, Add, Read, Update oraz Delete. Baza ta służy do przetrzymywania rezerwacji utworzonych przez użytkownika. Dokładny wygląd bazy danych i jej metod:

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {

    String createTableQuery = "CREATE TABLE " + TABLE_NAME + " ("
            + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "
            + COLUMN_DATE + " TEXT, "
            + COLUMN_TIME + " TEXT, "
            + COLUMN_RESTAURANT_NAME + " TEXT, "
            + COLUMN_ADDRESS + " TEXT, "
            + COLUMN_TABLE_NUM + " INTEGER, "
            + COLUMN_SEATS_NUM + " INTEGER, "
            + COLUMN_DATE_NUM + " INTEGER, "
            + COLUMN_TIME_NUM + " INTEGER)";
    sqLiteDatabase.execSQL(createTableQuery);
}
public long addReservation(Reservation reservation)
{
    SQLiteDatabase database = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_DATE, reservation.getDate());
    values.put(COLUMN_TIME, reservation.getTime());
    values.put(COLUMN_RESTAURANT_NAME, reservation.getRestaurantName());
    values.put(COLUMN_ADDRESS, reservation.getAddress());
    values.put(COLUMN_TABLE_NUM, reservation.getTableNum());
    values.put(COLUMN_SEATS_NUM, reservation.getSeatsNum());
    values.put(COLUMN_DATE_NUM, reservation.getDateNum());
    values.put(COLUMN_TIME_NUM, reservation.getTimeNum());
    long id = database.insert(TABLE_NAME, null, values);
    database.close();
    return id;
}

public SimpleCursorAdapter getAllReservations()
{
    SQLiteDatabase database = this.getReadableDatabase();
    String columns[] = {COLUMN_ID, COLUMN_DATE + " || ' - ' || " +
COLUMN_TIME + " AS " + "reservation_datetime", COLUMN_RESTAURANT_NAME,
COLUMN_ADDRESS, COLUMN_TABLE_NUM, COLUMN_SEATS_NUM, COLUMN_DATE_NUM,
COLUMN_TIME};
    Cursor cursor = database.query(TABLE_NAME, columns, null, null,null,
null, COLUMN_DATE_NUM + " ASC, " + COLUMN_TIME + " ASC");
    String[] fromFieldNames = new String[]{
            COLUMN_ID, "reservation_datetime", COLUMN_RESTAURANT_NAME,
COLUMN_ADDRESS, COLUMN_TABLE_NUM, COLUMN_SEATS_NUM, COLUMN_DATE_NUM,
COLUMN_TIME
    };
    int[] toViewIDs = new int[]{R.id.item_id, R.id.ReservationDate,
R.id.RestaurantName, R.id.RestaurantAdress, R.id.TNum, R.id.SNum};
    SimpleCursorAdapter reservationAdapter = new SimpleCursorAdapter(
            context,
            R.layout.single_item,
            cursor,
```

```
            fromFieldNames,
            toViewIDs
    );
    return reservationAdapter;
}

public long updateReservation(long id, Reservation reservation)
{
    SQLiteDatabase database = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_DATE, reservation.getDate());
    values.put(COLUMN_TIME, reservation.getTime());
    values.put(COLUMN_RESTAURANT_NAME, reservation.getRestaurantName());
    values.put(COLUMN_ADDRESS, reservation.getAddress());
    values.put(COLUMN_TABLE_NUM, reservation.getTableNum());
    values.put(COLUMN_SEATS_NUM, reservation.getSeatsNum());
    values.put(COLUMN_DATE_NUM, reservation.getDateNum());
    values.put(COLUMN_TIME_NUM, reservation.getTimeNum());
    String whereArgs[] = {""+id};
    long count = database.update(TABLE_NAME, values, COLUMN_ID + "=?",
whereArgs);
    database.close();
    return count;
}

public long deleteItem(long id)
{
    SQLiteDatabase database = this.getWritableDatabase();
    String whereArgs[] = {""+id};
    long count = database.delete(TABLE_NAME,COLUMN_ID + "=?", whereArgs);
    database.close();
    return count;
}
```

- ✓ Integracja z API (wyświetlanie danych z zewnętrznego API)

Aplikacja korzysta z OpenStreetMap API w celu pobrania z niej danych dotyczących restauracji takich jak współrzędne, nazwa restauracji czy jej adres. Dodatkowo API umożliwia mi skorzystanie z mapy i dodawania do niej pinesek które można edytować. Dokładny wygląd kodu:

```
@Override
public void RestaurantsAPISearch(double latitude, double longitude, int
radius, MapView mapView,
                                 android.graphics.drawable.Drawable
drawable)
{
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            OkHttpClient httpClient = new OkHttpClient();
            Request request = new Request.Builder()
                    .url("https://www.overpass-
api.de/api/interpreter?data=[out:json];node(around:" + radius + "," +
latitude + "," + longitude + ")[\"amenity\"=\"restaurant\"];out;")
                    .build();
            try (Response response = httpClient.newCall(request).execute())
{
                if (response.isSuccessful()) {
                    String responseBody = response.body().string();
                    JSONObject jsonResponse = new JSONObject(responseBody);
```

```
                    JSONArray elements =
jsonResponse.getJSONArray("elements");
                    for (int i = 0; i < elements.length(); i++) {
                        JSONObject element = elements.getJSONObject(i);
                        String name =
element.getJSONObject("tags").optString("name");
                        double lat = element.optDouble("lat");
                        double lon = element.optDouble("lon");
                        String cuisine =
element.getJSONObject("tags").optString("cuisine");
                        String adress =
element.getJSONObject("tags").optString("addr:city") + " " +

element.getJSONObject("tags").optString("addr:postcode") + " " +

element.getJSONObject("tags").optString("addr:street") + " " +

element.getJSONObject("tags").optString("addr:housenumber");
                        GeoPoint restaurantLocation = new GeoPoint(lat,
lon);
                        Marker restaurantMarker = new Marker(mapView);
                        restaurantMarker.setPosition(restaurantLocation);
                        restaurantMarker.setAnchor(Marker.ANCHOR_CENTER,
Marker.ANCHOR_BOTTOM);
                        restaurantMarker.setIcon(drawable);
                        CustomMarkerInfoWindow infoWindow = new
CustomMarkerInfoWindow(R.layout.custom_pop_up, mapView, name, cuisine,
adress);
                        restaurantMarker.setInfoWindow(infoWindow);
                        mapView.getOverlays().add(restaurantMarker);
                    }
                } else {
                    System.out.println("Error: " + response.code() + " " +
response.message());
                }
            } catch (IOException e) {
                e.printStackTrace();
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
        }
    });
    thread.start();
}
```

- ✓ Wykorzystanie jednego z sensorów urządzenia mobilnego – w tym przypadku GPS

Aplikacja ta korzysta z sensora GPS i w zależności od lokacji użytkownika i podanego przez niego zasięgu program szuka restauracji znajdujących się w tym zasięgu od niego. Dokładny kod wygląda następująco:

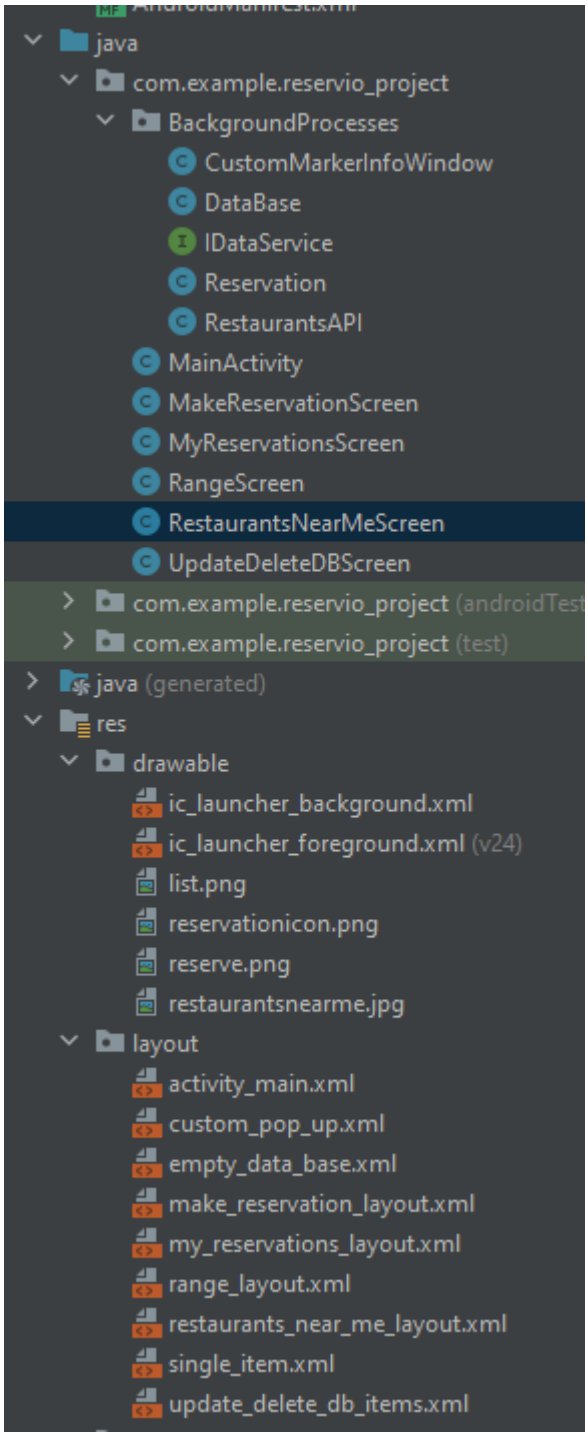
```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
/>
```

```
locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

if (ActivityCompat.checkSelfPermission(this,
```

```
permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(this, new
String[]{permission.ACCESS_FINE_LOCATION}, 1);
} else {
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
0, new LocationListener() {
        @SuppressLint("UseCompatLoadingForDrawables")
        @Override
        public void onLocationChanged(Location location) {
            latitude = location.getLatitude();
            longitude = location.getLongitude();
            GeoPoint currentLocation = new GeoPoint(latitude, longitude);
            if(x == 0)
            {
                mapController.setCenter(currentLocation);
            }
            Marker marker = new Marker(mapView);
            marker.setPosition(currentLocation);
            marker.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM);

marker.setIcon(getResources().getDrawable(org.osmdroid.library.R.drawable.p
erson));
            marker.setTitle("That's you !");
            mapView.getOverlays().add(marker);
            android.graphics.drawable.Drawable restauranticon =
getResources().getDrawable(org.osmdroid.library.R.drawable.marker_default);
            dataService = new RestaurantsAPI();
            dataService.RestaurantsAPISearch(latitude, longitude, range,
mapView, restauranticon);
            mapView.invalidate();
            x = 1;
        }
    });
}
```

- ✓ Zastosowanie odpowiednich wzorców projektowych i architektonicznych – w tym przypadku MVC

Cały projekt został utworzony na bazie wzorca MVC – posiada odpowiednie klasy, kontrolery oraz view w postaci plików xml. Dokładna struktura plików jest następująca:



- ✓ Dodatkowo punktowane wykorzystanie bibliotek DI (dependency injection)

Do dependency injection została użyta biblioteka dagger i stosuję ją w momencie przesyłania danych z API do klasy obsługującej mapę. Dokładny kod obsługujący tę działanie jest następujący:

```
public interface IDataService {
    void RestaurantsAPISearch(double latitude, double longitude, int
radius, MapView mapView,

android.graphics.drawable.Drawable drawable);
}
```

```
@Inject
IDataService dataService;
```

```
dataService = new RestaurantsAPI();
dataService.RestaurantsAPISearch(latitude, longitude, range, mapView,
restauranticon);
```

```
public class RestaurantsAPI implements IDataService {
```