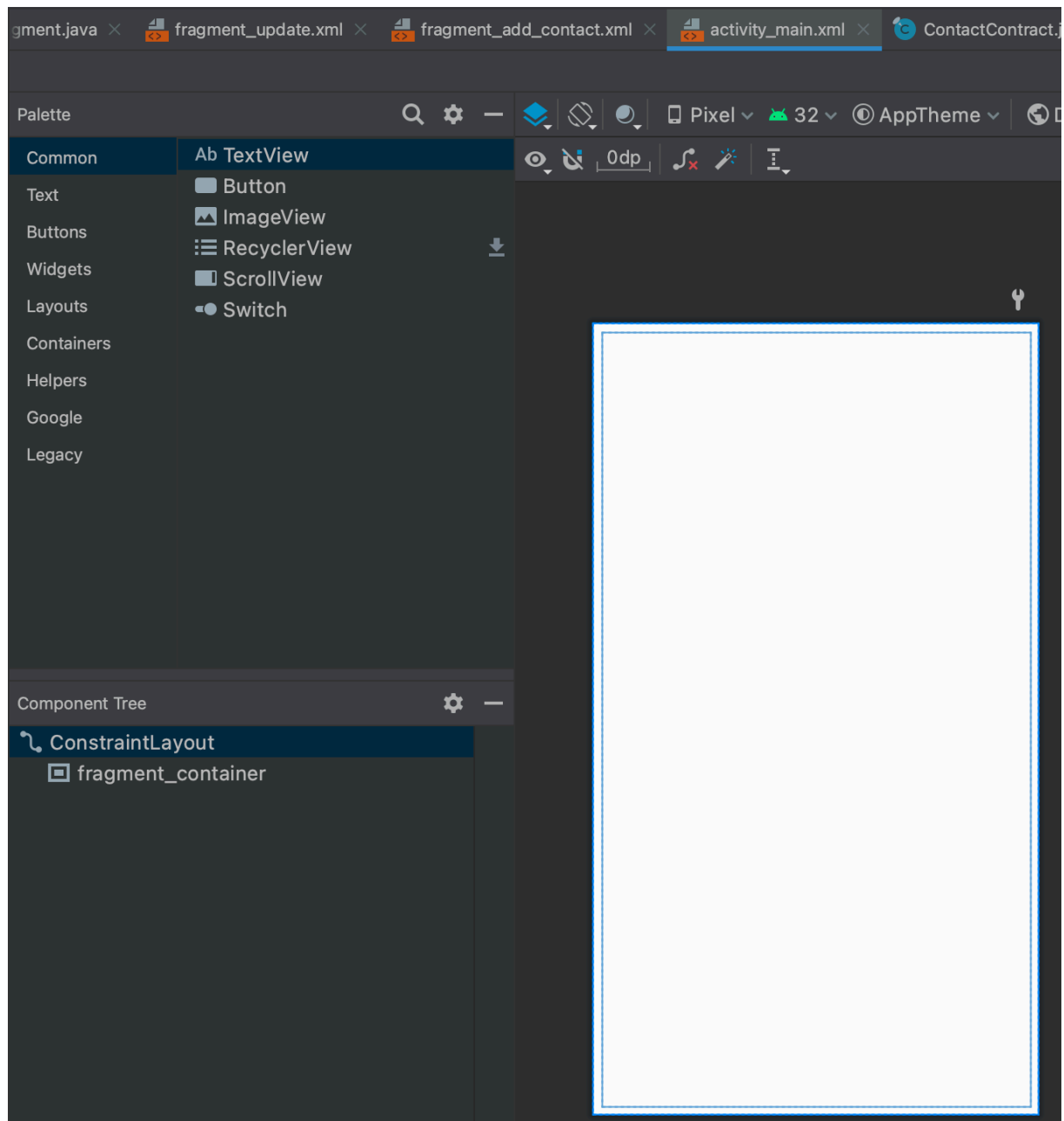


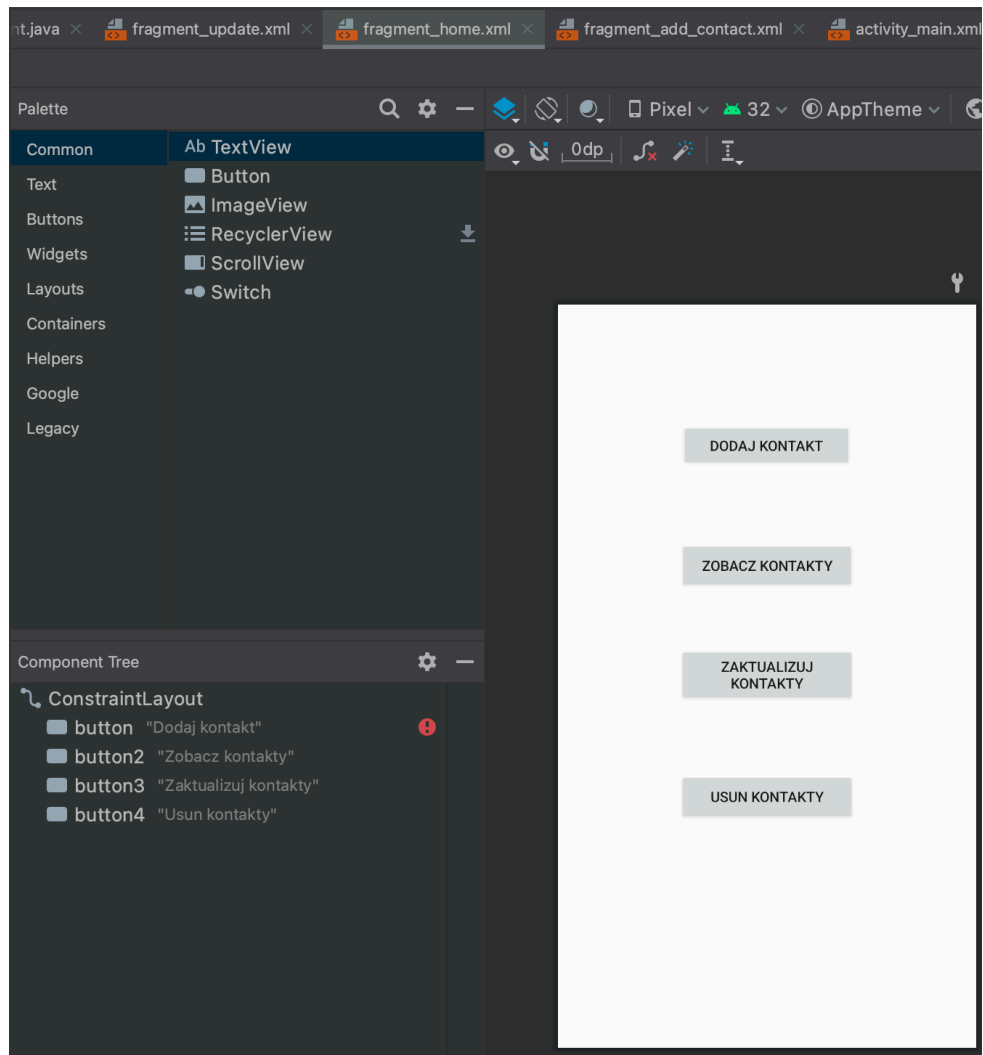
Bazy danych to jedna z wielu możliwości przechowywania danych naszej aplikacji, nawet gdy zostanie ona wyłączona. Lokalne bazy danych nie wymagają połączenia internetowego i stanowią doskonały sposób zapisu dużej ilości informacji. Aplikacje na platformę Android korzystają z SQLite, którego działanie przedstawię w poniższej aplikacji. Aplikacja jest prostą aplikacją wykorzystującą możliwości jakie daje SQLite w Androidzie. (Warto w tym miejscu przypomnieć sobie podstawowe komendy SQL). Aplikacja będzie dawała możliwość zapisywania, wyświetlania, edytowania oraz usuwania „kontaktów”, oprzemy ją na fragmentach – dosłownie w całości, więc `main_activity.xml` będzie tylko i wyłącznie kontenerem na fragmenty:



```
gment.java × fragment_update.xml × fragment_add_contact.xml × activity_main.xml ×

1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity">
9
10
11      <FrameLayout
12          android:id="@+id/fragment_container"
13          android:layout_width="395dp"
14          android:layout_height="715dp"
15          android:layout_marginStart="8dp"
16          android:layout_marginTop="8dp"
17          android:layout_marginEnd="8dp"
18          android:layout_marginBottom="8dp"
19          app:layout_constraintBottom_toBottomOf="parent"
20          app:layout_constraintEnd_toEndOf="parent"
21          app:layout_constraintStart_toStartOf="parent"
22          app:layout_constraintTop_toTopOf="parent">
23
24      </FrameLayout>
25  </android.support.constraint.ConstraintLayout>
```

1. Najpierw stworzymy wszystkie fragmenty i logikę ich uruchamiania. Pierwszym fragmentem będzie HomeFragment, który wyświetla dostępne opcje, powinien wyglądać mniej więcej tak:



Kod XML:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeFragment">

    <Button
        android:id="@+id/button"
        android:layout_width="168dp"
        android:layout_height="44dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="Dodaj kontakt"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
    />
</android.support.constraint.ConstraintLayout>
```

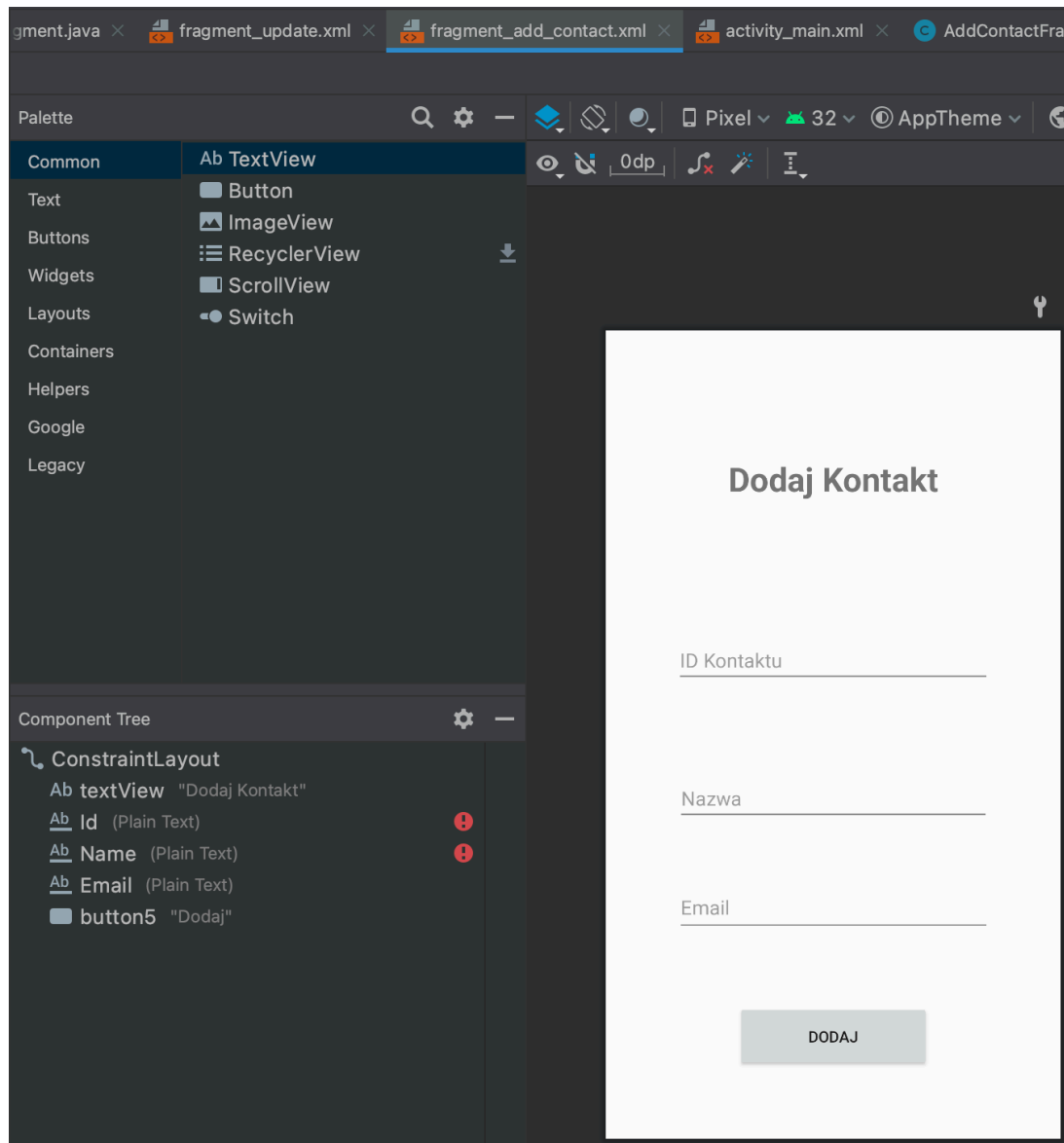
```
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.162" />

<Button
    android:id="@+id/button2"
    android:layout_width="172dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="72dp"
    android:layout_marginEnd="8dp"
    android:text="Zobacz kontakty"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button" />

<Button
    android:id="@+id/button3"
    android:layout_width="173dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="56dp"
    android:layout_marginEnd="8dp"
    android:text="Zaktualizuj kontakty"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button2" />

<Button
    android:id="@+id/button4"
    android:layout_width="173dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:text="Usun kontakty"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button3"
    app:layout_constraintVertical_bias="0.219" />
</android.support.constraint.ConstraintLayout>
```

2. stworzymy teraz fragment: AddContactFragment, będzie on odpowiedzialny za dodawanie nowych kontaktów. Każdy kontakt to trzy pola: ID, nazwa i email. Tak powinien mniej więcej wyglądać:



Kod fragment_add_contact.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AddContactFragment">

    <TextView
        android:id="@+id/textView"
        android:layout_width="283dp"
```

```
android:layout_height="86dp"
android:layout_marginStart="8dp"
android:layout_marginTop="8dp"
android:layout_marginEnd="8dp"
android:layout_marginBottom="8dp"
android:text="Dodaj Kontakt"
android:textSize="30sp"
android:textAlignment="center"
android:textStyle="bold"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.168" />
```

<EditText

```
android:id="@+id/Id"
android:layout_width="285dp"
android:layout_height="wrap_content"
android:layout_marginStart="8dp"
android:layout_marginTop="76dp"
android:layout_marginEnd="8dp"
android:ems="10"
android:hint="ID Kontaktu"
android:inputType="textPersonName"
android:text=""
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/textView" />
```

<EditText

```
android:id="@+id/Name"
android:layout_width="283dp"
android:layout_height="wrap_content"
android:layout_marginStart="8dp"
android:layout_marginTop="80dp"
android:layout_marginEnd="8dp"
android:ems="10"
android:hint="Nazwa"
android:inputType="textPersonName"
android:text=""
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/Id" />
```

<EditText

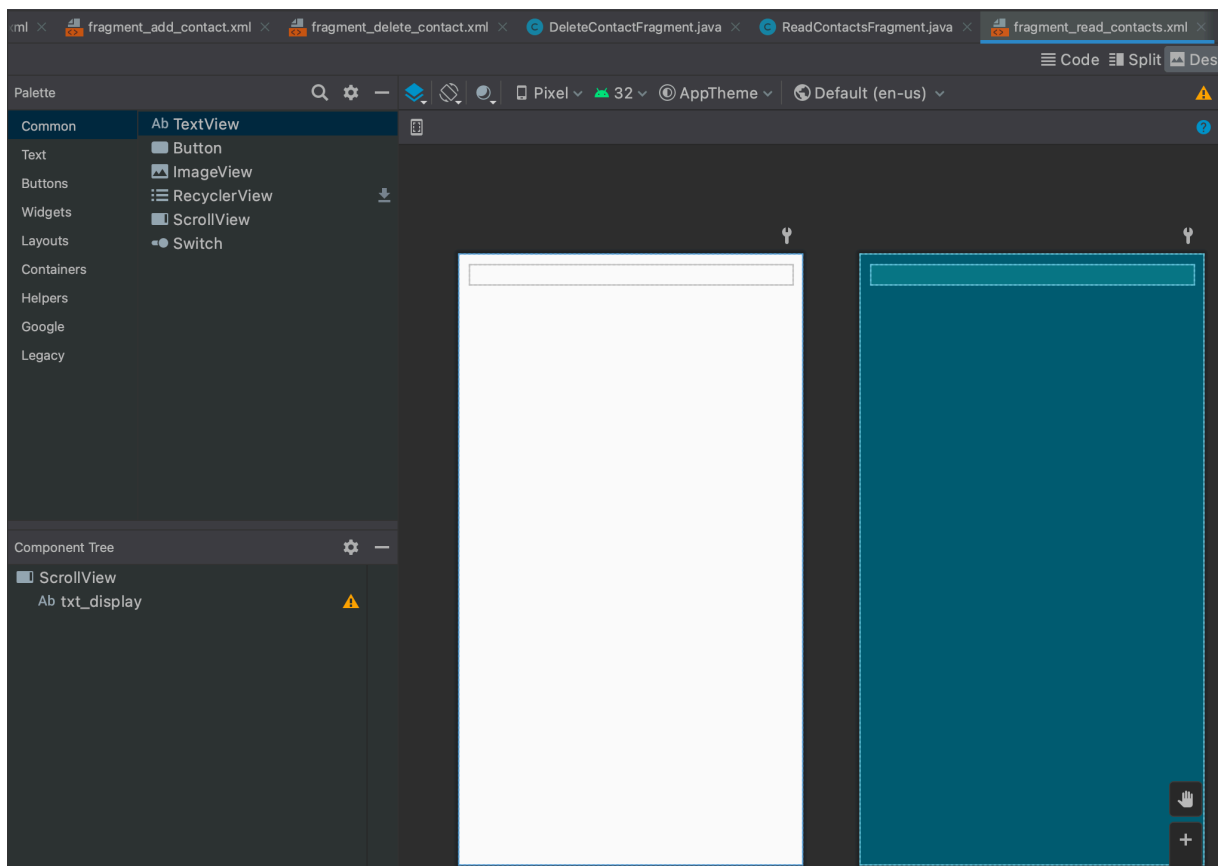
```
android:id="@+id/Email"
android:layout_width="284dp"
android:layout_height="48dp"
android:layout_marginStart="8dp"
android:layout_marginTop="52dp"
android:layout_marginEnd="8dp"
android:ems="10"
android:hint="Email"
android:inputType="textPersonName"
android:text=""
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.504"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/Name" />
```

```

<Button
    android:id="@+id/button5"
    android:layout_width="174dp"
    android:layout_height="59dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:text="Dodaj"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/Email" />
</android.support.constraint.ConstraintLayout>

```

3. Tworzymy kolejny fragment: ReadContactsFragment, który używa komponentu ScrollView i będzie służył do wyświetlania zapisanych kontaktów:



```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ReadContactsFragment">

    <TextView
        android:layout_width="match_parent"

```

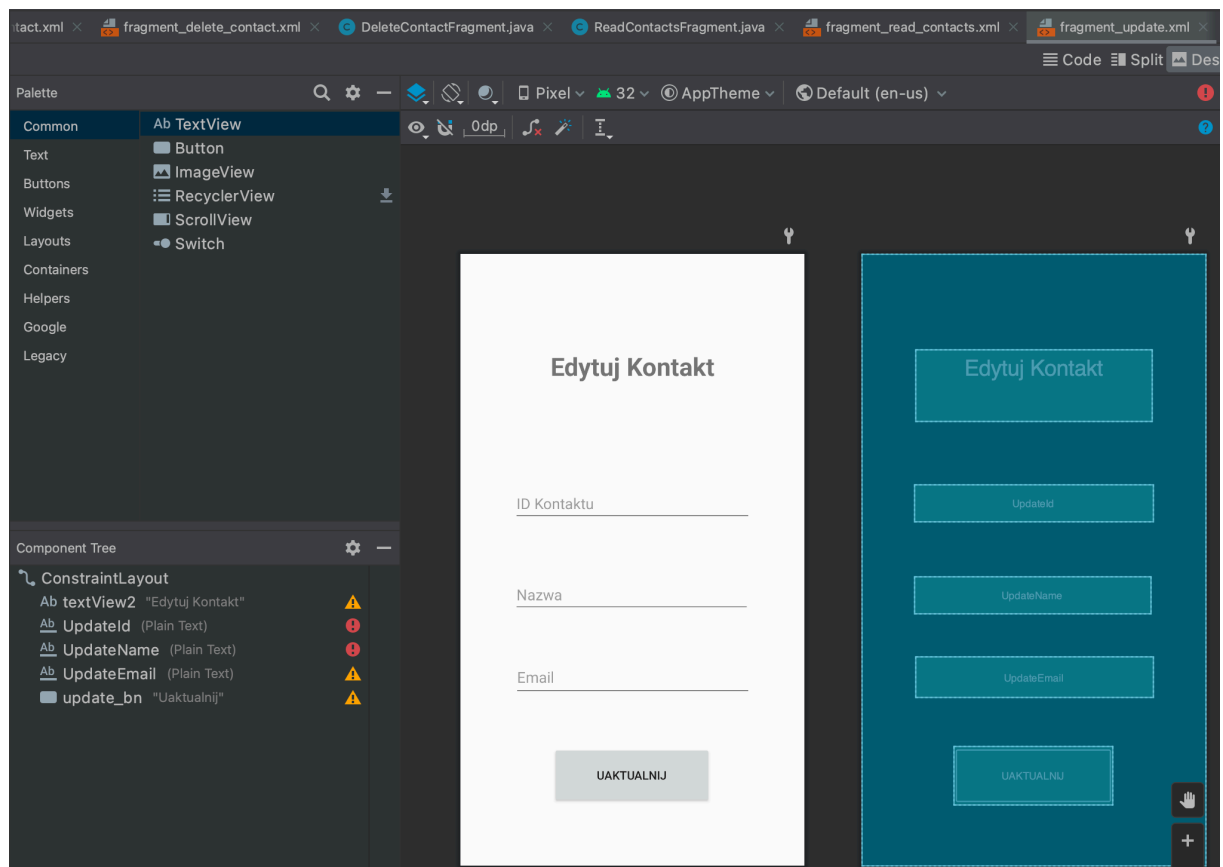
```

        android:layout_height="match_parent"
        android:id="@+id/txt_display"
        android:textStyle="bold"
        android:textSize="18sp"
        android:layout_margin="12dp"
        android:text="" />

</ScrollView>

```

4. Kolejny fragment to UpdateFragment, który będzie służył do edycji kontaktu o podanym ID, jest w zasadzie kalką fragmentu dodającego kontakt:



```

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UpdateFragment">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="283dp"
        android:layout_height="86dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"

```



```

        android:layout_marginBottom="8dp"
        android:text="Edytuj Kontakt"
        android:textSize="30sp"
        android:textAlignment="center"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.168" />

<EditText
    android:id="@+id/UpdateId"
    android:layout_width="285dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="76dp"
    android:layout_marginEnd="8dp"
    android:ems="10"
    android:hint="ID Kontaktu"
    android:inputType="textPersonName"
    android:text=""
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView2" />

<EditText
    android:id="@+id/UpdateName"
    android:layout_width="283dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="64dp"
    android:layout_marginEnd="8dp"
    android:ems="10"
    android:hint="Nazwa"
    android:inputType="textPersonName"
    android:text=""
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.491"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/UpdateId" />

<EditText
    android:id="@+id/UpdateEmail"
    android:layout_width="284dp"
    android:layout_height="48dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="52dp"
    android:layout_marginEnd="8dp"
    android:ems="10"
    android:hint="Email"
    android:inputType="textPersonName"
    android:text=""
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.504"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/UpdateName" />

<Button
    android:id="@+id/update_bn"
    android:layout_width="190dp"

```

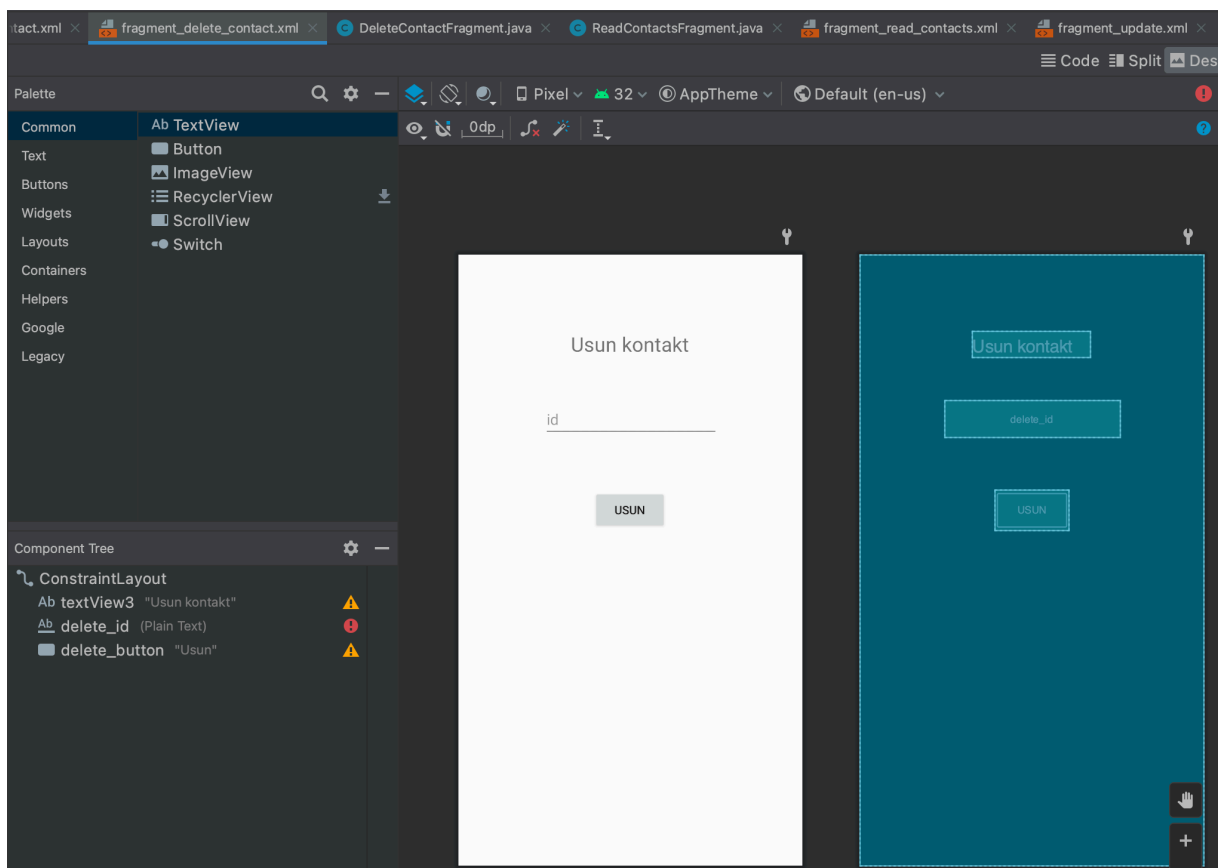
```

        android:layout_height="70dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="Uaktualnij"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.899" />

</android.support.constraint.ConstraintLayout>

```

5. Ostatni fragment, DeleteFragment, który będzie dawał możliwość usunięcia elementu z bazy danych:



```

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DeleteContactFragment">

    <TextView

```

```

        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="Usun kontakt"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.121" />

<EditText
    android:id="@+id/delete_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="98dp"
    android:layout_marginTop="64dp"
    android:layout_marginEnd="100dp"
    android:layout_marginBottom="37dp"
    android:ems="10"
    android:hint="id"
    android:inputType="textPersonName"
    android:text=""
    app:layout_constraintBottom_toTopOf="@+id/delete_button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView3" />

<Button
    android:id="@+id/delete_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="161dp"
    android:layout_marginTop="37dp"
    android:layout_marginEnd="162dp"
    android:layout_marginBottom="414dp"
    android:text="Usun"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/delete_id" />

</android.support.constraint.ConstraintLayout>

```

A tak powinno wyglądać drzewo projektu po dodaniu wszystkich niezbędnych fragmentów: (ContactContract oraz ContactDbHelper jeszcze nie ma, stworzymy je w późniejszych krokach)

SQLiteExample > app > src > main > res > mipmap-hdpi

Project

Android

Resource Manager

app

manifests

java

com.example.sqliteexample

AddContactFragment

ContactContract

ContactDbHelper

DeleteContactFragment

HomeFragment

MainActivity

ReadContactsFragment

UpdateFragment

com.example.sqliteexample (androidTest)

com.example.sqliteexample (test)

java (generated)

res

drawable

layout

activity_main.xml

fragment_add_contact.xml

fragment_delete_contact.xml

fragment_home.xml

fragment_read_contacts.xml

fragment_update.xml

mipmap

values

res (generated)

Structure

Gradle Scripts

6. Kolejny krok to implementacja mechanizmu pozwalającego na wywoływanie odpowiednich fragmentów. Najpierw zaczynamy od zmodyfikowania kodu w HomeFragment:

```
sqliteexample > HomeFragment
HomeFragment.java x
1 package com.example.sqliteexample;
2
3 import ...
11
12 public class HomeFragment extends Fragment implements View.OnClickListener {
13
14     //Musimy zaimplementować we fragmencie interfejs View.OnClickListener!!!
15
16     private Button BnSave, BnView, BnDelete, BnUpdate; //przyciski wyboru
17     OnDbOpListener dbOpListener;
18
19     public HomeFragment() {
20         // Wymaga pustego konstruktora
21     }
22
23     public interface OnDbOpListener{
24         void dbOpPerformed(int method);
25         //interfejsy odpowiedzialne za wybór fragmentu, implementacja w MainActivity
26     }
27
28
29     @Override
30     public View onCreateView(LayoutInflater inflater, ViewGroup container,
31                             Bundle savedInstanceState) {
32
33         View view = inflater.inflate(R.layout.fragment_home, container, attachToRoot: false);
34         BnSave = view.findViewById(R.id.button);
35         //ustawienie guzika do dodawania kontaktów
36         BnSave.setOnClickListener(this);
37         BnView = view.findViewById(R.id.button2);
38         //ustawienie guzika do podglądu kontaktów
39         BnView.setOnClickListener(this);
40         BnUpdate = view.findViewById(R.id.button3);
41         //przycisk edycji kontaktów
42         BnUpdate.setOnClickListener(this);
43         BnDelete = view.findViewById(R.id.button4);
44         //przycisk usuwania kontaktów
45         BnDelete.setOnClickListener(this);
46         return view;
47     }
}
```

Musimy jeszcze przesłonić metodę onClick tak, aby na podstawie naciśniętego przycisku wysłać odpowiednią informację do MainActivity:

```

48
49      @Override
50      public void onClick(View v) {
51          switch (v.getId()){
52              // Jeśli zostanie naciśnięty przycisk dodania kontaktu,
53              // Zostaje ustawiona wartość 0 odczytywana później w MainActivity.
54              // Zastąpiony zostaje wtedy fragment
55              case R.id.button:
56                  dbOpListener.dbOpPerformed( method: 0);
57                  break;
58              case R.id.button2:
59                  dbOpListener.dbOpPerformed( method: 1);
60                  break;
61              case R.id.button3:
62                  dbOpListener.dbOpPerformed( method: 2);
63                  break;
64              case R.id.button4:
65                  dbOpListener.dbOpPerformed( method: 3);
66                  break;
67          }
68      }
69
70      public void onAttach(Context context){
71
72          super.onAttach(context);
73          Activity activity = (Activity) context;
74          try{
75              dbOpListener = (OnDbOpListener) activity;
76          }
77          catch (ClassCastException e){
78              throw new ClassCastException(activity.toString()
79                  + " trzeba zaimplementować interfejs");
80          }
81
82      }
83  }

```

W MainActivity implementujemy kontener na fragmenty modyfikując metodę onCreate: (Pamiętajmy o tym, by zaimplementować interfejs z HomeFragment)

sqliteexample > MainActivity

app Pixel 5 API 27

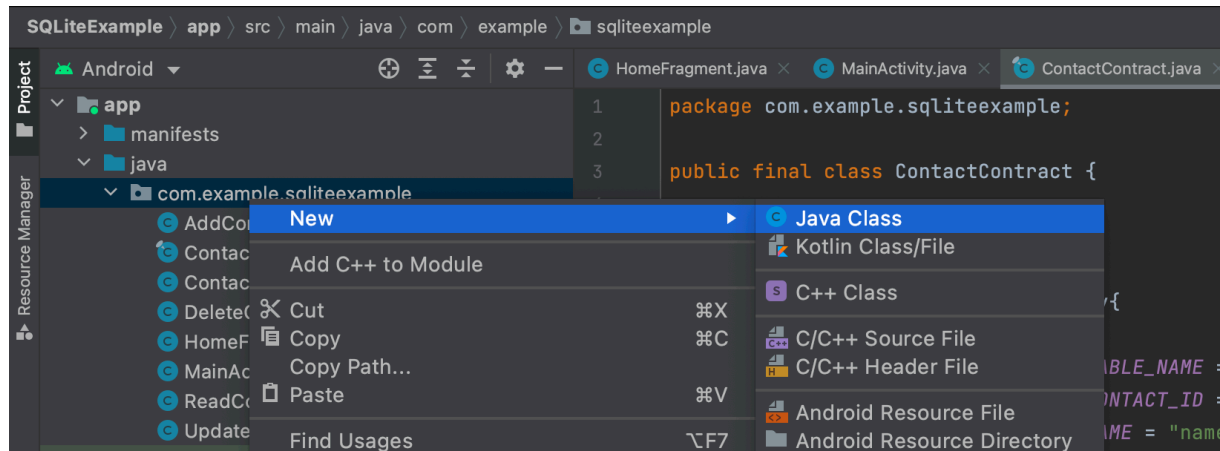
HomeFragment.java x MainActivity.java x

```

1 package com.example.sqliteexample;
2
3 import ...
4
5
6 public class MainActivity extends AppCompatActivity implements HomeFragment.OnDbOpListener {
7
8     //Pamiętajmy, że implementujemy interfejs z HomeFragment!!!!
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13
14        if(findViewById(R.id.fragment_container)!=null){
15            if(savedInstanceState!=null){
16                return;
17            }
18
19            HomeFragment homeFragment = new HomeFragment();
20            getSupportFragmentManager().beginTransaction()
21                .add(R.id.fragment_container, homeFragment).commit();
22        }
23
24    }
25
26    @Override
27    public void dbOpPerformed(int method) {
28        switch (method){
29            //ODBIERANIE INFORMACJI O AKTUALNIE WYŚWIETLANYM FRAGMENTCIE
30            case 0:
31                getSupportFragmentManager().beginTransaction()
32                    .replace(R.id.fragment_container,
33                        new AddContactFragment()).addToBackStack(null).commit();
34                break;
35            case 1:
36                getSupportFragmentManager().beginTransaction()
37                    .replace(R.id.fragment_container,
38                        new ReadContactsFragment()).addToBackStack(null).commit();
39                break;
40            case 2:
41                getSupportFragmentManager().beginTransaction()
42                    .replace(R.id.fragment_container,
43                        new UpdateFragment()).addToBackStack(null).commit();
44                break;
45            case 3:
46                getSupportFragmentManager().beginTransaction()
47                    .replace(R.id.fragment_container,
48                        new DeleteContactFragment()).addToBackStack(null).commit();
49                break;
50        }
51    }
52 }
53

```

7. Tworzymy teraz klasę Javy, która będzie nam służyła za szablon obiektu dla wpisywanych danych:



8. Kolejny krok, to stworzenie klasy, która będzie naszą klasą z metodami obsługującymi bazę danych. Większość to po prostu komendy SQL:

```
sqliteexample > ContactDbHelper
ContactContract.java × ContactDbHelper.java ×
1 package com.example.sqliteexample;
2
3 import ...
9
10 // Ta klasa to tak zwany Helper, tworzymy tu wszystkie metody
11 // odpowiedzialne za naszą bazę danych.
12 public class ContactDbHelper extends SQLiteOpenHelper{
13
14     public static final String DATABASE_NAME = "contact_db";
15     // Definiujemy nazwę naszej bazy danych
16     public static final int DATABASE_VERSION = 1;
17
18     //połączenie tworzenia bazy danych
19     public static final String CREATE_TABLE = "create table "
20         + ContactContract.ContractEntry.TABLE_NAME+"("
21         + ContactContract.ContractEntry.CONTACT_ID+" number,"
22         + ContactContract.ContractEntry.NAME+" text,"
23         + ContactContract.ContractEntry.EMAIL+" text)";
24
25     public static final String DROP_TABLE = "drop table if exists "
26         + ContactContract.ContractEntry.TABLE_NAME;
27
28     public ContactDbHelper(Context context){
29         super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);
30     }
31
32     @Override
33     public void onCreate(SQLiteDatabase db) {
34         db.execSQL(CREATE_TABLE);
35     }
36
37     @Override
38     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
39         db.execSQL(DROP_TABLE);
40         onCreate(db);
41     }
42
43     //metoda dodająca kontakt do bazy danych
44     // używa atrybutów z klasy ContactContract
45     public void addContact(int id, String name, String email,
46         SQLiteDatabase database){
47         ContentValues contactValues = new ContentValues();
48         contactValues.put(ContactContract.ContractEntry.CONTACT_ID, id);
49         contactValues.put(ContactContract.ContractEntry.NAME, name);
50         contactValues.put(ContactContract.ContractEntry.EMAIL, email);
51
52         database.insert(ContactContract.ContractEntry.TABLE_NAME,
53             nullColumnHack: null, contactValues);
54     }
55 }
```

9. Mamy wszystko. Teraz implementacja fragmentów. Najpierw dodawanie kontaktu do bazy:

```
sqliteexample > AddContactFragment
AddContactFragment.java x
1 package com.example.sqliteexample;
2
3
4 import ...
13
14 public class AddContactFragment extends Fragment {
15
16     private Button bnSave;
17     EditText Id, Name, Email;
18
19     public AddContactFragment() {
20     }
21
22     @Override
23     public View onCreateView(LayoutInflater inflater, ViewGroup container,
24                             Bundle savedInstanceState) {
25         View view = inflater.inflate(R.layout.fragment_add_contact,
26                                     container, attachToRoot: false);
27
28         bnSave = view.findViewById(R.id.button5);
29         Id = view.findViewById(R.id.Id);
30         Name = view.findViewById(R.id.Name);
31         Email = view.findViewById(R.id.Email);
32
33         bnSave.setOnClickListener(new View.OnClickListener() {
34             @Override
35             public void onClick(View v) {
36                 //pobieranie elementów bazy danych
37                 String id = Id.getText().toString();
38                 String name = Name.getText().toString();
39                 String email = Email.getText().toString();
40
41                 ContactDbHelper contactDbHelper = new ContactDbHelper(getActivity());
42                 SQLiteDatabase database = contactDbHelper.getWritableDatabase();
43                 int int_id = Integer.parseInt(id);
44                 contactDbHelper.addContact(int_id, name, email, database);
45                 contactDbHelper.close();
46                 Id.setText("");
47                 Name.setText("");
48                 Email.setText("");
49                 Toast.makeText(getActivity(), text: "Dodano",
50                             Toast.LENGTH_SHORT).show();
51             }
52         });
53
54         return view;
55     }
56
57 }
58
```

Dodawanie powinno działać, jeszcze nie możemy podejrzeć efektu, ale jeśli zobaczymy wiadomość Toast, najpewniej wszystko działa poprawnie.

10. Żeby wyświetlić dodane kontakty, musimy dodać metodę do DbHelpera:

```
57 @ public Cursor readCotacts(SQLiteDatabase database){
58
59     String[] projections = {ContactContract.ContractEntry.CONTACT_ID,
60         ContactContract.ContractEntry.NAME, ContactContract.ContractEntry.EMAIL};
61
62
63     Cursor cursor = database.query(ContactContract.ContractEntry.TABLE_NAME, projections,
64         selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
65
66     return cursor;
67
68 }
```

Teraz fragment ReadContactsFragment wyświetlający wszystkie wpisy z bazy:

```
sqliteexample > ReadContactsFragment
AddContactFragment.java x fragment_read_contacts.xml x ReadContactsFragment.java x
1 package com.example.sqliteexample;
2
3 import ...
11
12 public class ReadContactsFragment extends Fragment {
13
14     private TextView Txt_Display;
15
16     public ReadContactsFragment() {
17     }
18
19     @Override
20     @ public View onCreateView(LayoutInflater inflater, ViewGroup container,
21         Bundle savedInstanceState) {
22
23         View view = inflater.inflate(R.layout.fragment_read_contacts,
24             container, attachToRoot: false);
25
26         Txt_Display = view.findViewById(R.id.txt_display);
27         readContacts();
28         return view;
29     }
```

```

31     private void readContacts(){
32         ContactDbHelper contactDbHelper = new ContactDbHelper(getActivity());
33         SQLiteDatabase database = contactDbHelper.getReadableDatabase();
34
35
36         Cursor cursor = contactDbHelper.readContacts(database);
37         String info = "";
38
39         // Za pomocą petli pobieramy wszystkie wpisy z tabeli
40         while (cursor.moveToNext()){
41             String id = Integer.toString(cursor.getInt(cursor
42                 .getColumnIndex(ContactContract.ContractEntry.CONTACT_ID)));
43             String name = cursor.getString(cursor
44                 .getColumnIndex(ContactContract.ContractEntry.NAME));
45             String email = cursor.getString(cursor
46                 .getColumnIndex(ContactContract.ContractEntry.EMAIL));
47
48             info = info + "\n\n" + "Id: " + id + "\nName : " + name + "\nEmail : " + email;
49         }
50
51         Txt_Display.setText(info);
52         contactDbHelper.close();
53     }
54 }
55
56 }

```

11. Tak samo by edytować kontakt, dodajemy odpowiednią metodę do ContactDbHelper:

```

70 @ public void updateContact(int id, String name, String email, SQLiteDatabase database){
71     ContentValues contentValues = new ContentValues();
72     contentValues.put(ContactContract.ContractEntry.NAME, name);
73     contentValues.put(ContactContract.ContractEntry.EMAIL, email);
74
75     String selection = ContactContract.ContractEntry.CONTACT_ID + " = " + id;
76
77     database.update(ContactContract.ContractEntry
78         .TABLE_NAME, contentValues, selection, whereArgs: null);
79 }

```

Następnie modyfikujemy kod fragmentu UpdateFragment

```

sqliteexample > UpdateFragment / updateContact
AddContactFragment.java x fragment_read_contacts.xml x ReadContactsFragment.java x UpdateFragment.java x ContactDbHelper
1 package com.example.sqliteexample;
2
3 import ...
12
13 public class UpdateFragment extends Fragment {
14
15     private EditText Update_id, Update_name, Update_email;
16     private Button Update_bn;
17
18     public UpdateFragment() {
19         // Required empty public constructor
20     }
21
22     @Override
23     @
24     public View onCreateView(LayoutInflater inflater, ViewGroup container,
25                             Bundle savedInstanceState) {
26
27         View view = inflater.inflate(R.layout.fragment_update, container, attachToRoot: false);
28
29         Update_id = view.findViewById(R.id.UpdateId);
30         Update_name = view.findViewById(R.id.UpdateName);
31         Update_email = view.findViewById(R.id.UpdateEmail);
32
33         Update_bn = view.findViewById(R.id.update_bn);
34
35         Update_bn.setOnClickListener(new View.OnClickListener() {
36             @Override
37             public void onClick(View v) {
38                 updateContact();
39             }
40         });
41         return view;
42     }
43
44     private void updateContact(){
45
46         int id = Integer.parseInt(Update_id.getText().toString());
47         String name = Update_name.getText().toString();
48         String email = Update_email.getText().toString();
49
50         ContactDbHelper contactDbHelper = new ContactDbHelper(getActivity());
51         SQLiteDatabase database = contactDbHelper.getWritableDatabase();
52
53         contactDbHelper.updateContact(id, name, email, database);
54
55         contactDbHelper.close();
56
57         Toast.makeText(getActivity(), text: "Kontakt zaktualizowany", Toast.LENGTH_LONG).show();
58
59         Update_id.setText("");
60         Update_name.setText("");
61         Update_email.setText("");
62     }
63 }
64

```

12. Na koniec usuwanie, dodajemy metodę do ContactDbHelper:

```
81 @ public void deleteContact(int id, SQLiteDatabase database){
82     String selection = ContactContract.ContractEntry.CONTACT_ID+" = "+id;
83     database.delete(ContactContract.ContractEntry.TABLE_NAME, selection, whereArgs: null);
84
85 }
86
87 }
```

Oraz implementujemy usuwanie w DeleteFragment:

```
sqliteexample > DeleteContactFragment
DeleteContactFragment.java x fragment_read_contacts.xml x ReadContactsFragment.java x UpdateFragment.java x ContactDbHelper.java x
1 package com.example.sqliteexample;
2
3
4 import ...
5
13
14
15 public class DeleteContactFragment extends Fragment {
16     private EditText Id;
17     private Button Button_Delete;
18
19     public DeleteContactFragment() {
20     }
21
22     @Override
23     @ public View onCreateView(LayoutInflater inflater, ViewGroup container,
24                             Bundle savedInstanceState) {
25
26         View view = inflater.inflate(R.layout.fragment_delete_contact,
27                                     container, attachToRoot: false);
28
29         Id = view.findViewById(R.id.delete_id);
30         Button_Delete = view.findViewById(R.id.delete_button);
31
32         Button_Delete.setOnClickListener(new View.OnClickListener() {
33             @Override
34             public void onClick(View v) {
35                 deleteContact();
36             }
37         });
38
39         return view;
40     }
```

```
41
42     private void deleteContact(){
43
44         ContactDbHelper contactDbHelper = new ContactDbHelper(getActivity());
45         SQLiteDatabase database = contactDbHelper.getWritableDatabase();
46
47         int id = Integer.parseInt(Id.getText().toString());
48         contactDbHelper.deleteContact(id, database);
49         contactDbHelper.close();
50         Id.setText("");
51         Toast.makeText(getActivity(), text: "Usunieto", Toast.LENGTH_SHORT).show();
52
53     }
54
55 }
56
57
```