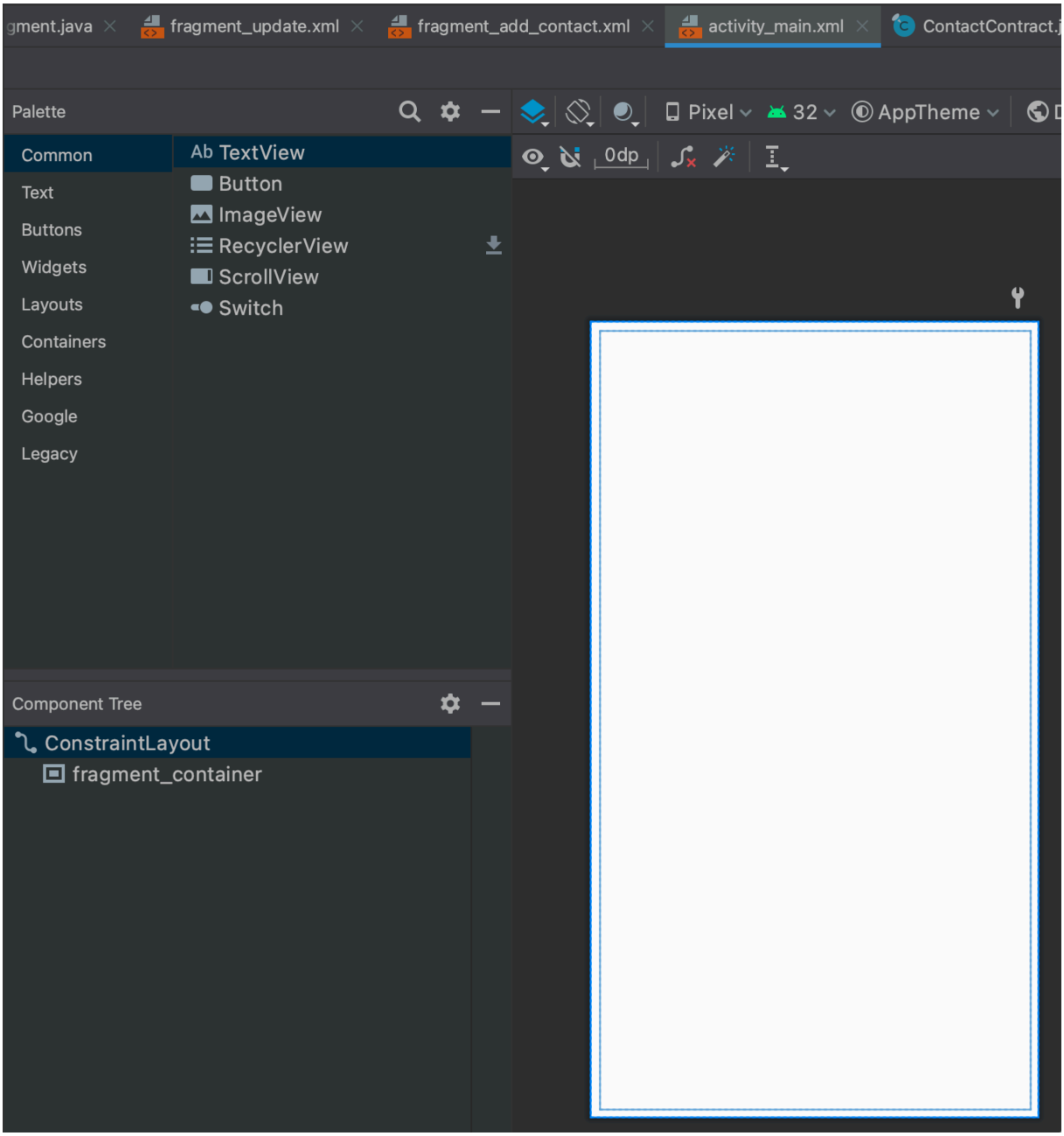
Bazy danych to jedna z wielu możliwości przechowywania danych naszej aplikacji, nawet gdy zostanie ona wyłączona. Lokalne bazy danych nie wymagają połączenia internetowego i stanowią doskonały sposób zapisu dużej ilości informacji. Aplikacje na platformę Android korzystają z SQLite, którego działanie 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:

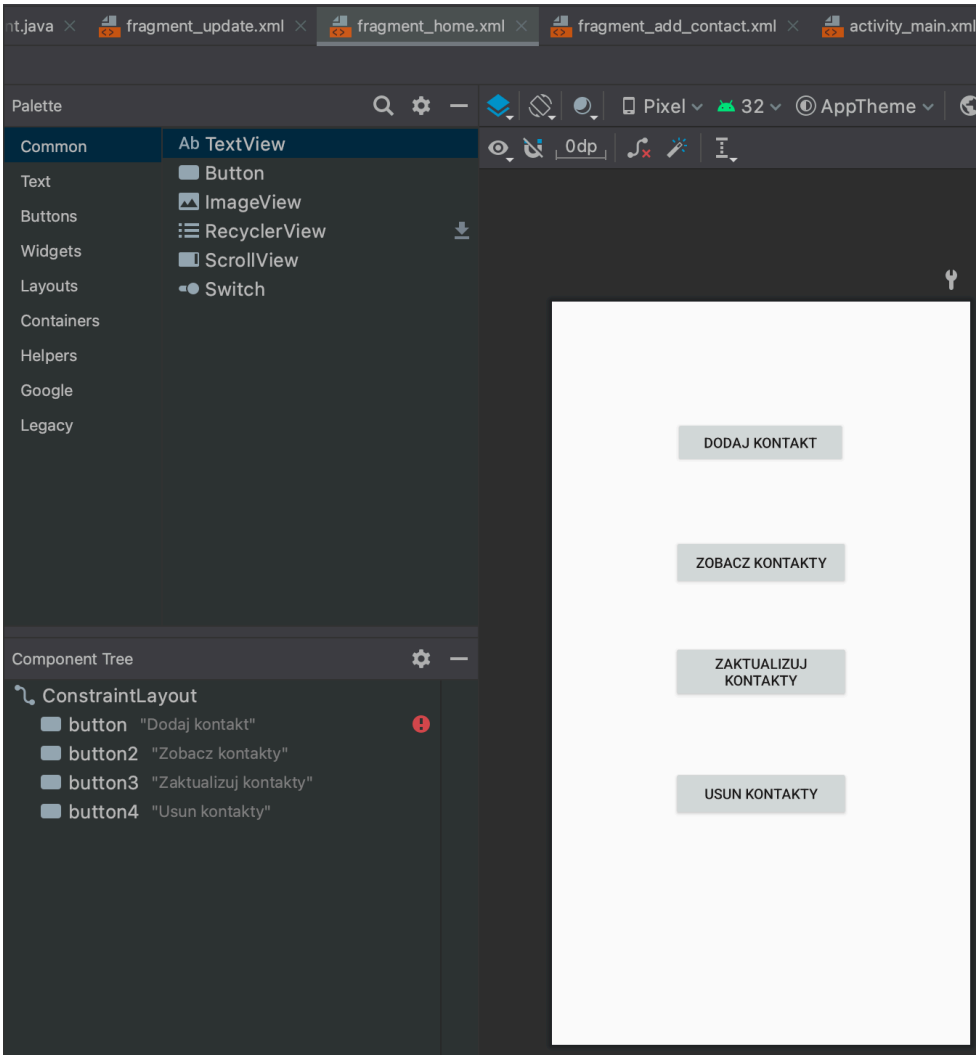


gment.java | fragment_update.xml | fragment_add_contact.xml | activity_main.xml

```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=".MainActivity">


    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="395dp"
        android:layout_height="715dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

    </FrameLayout>
</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:

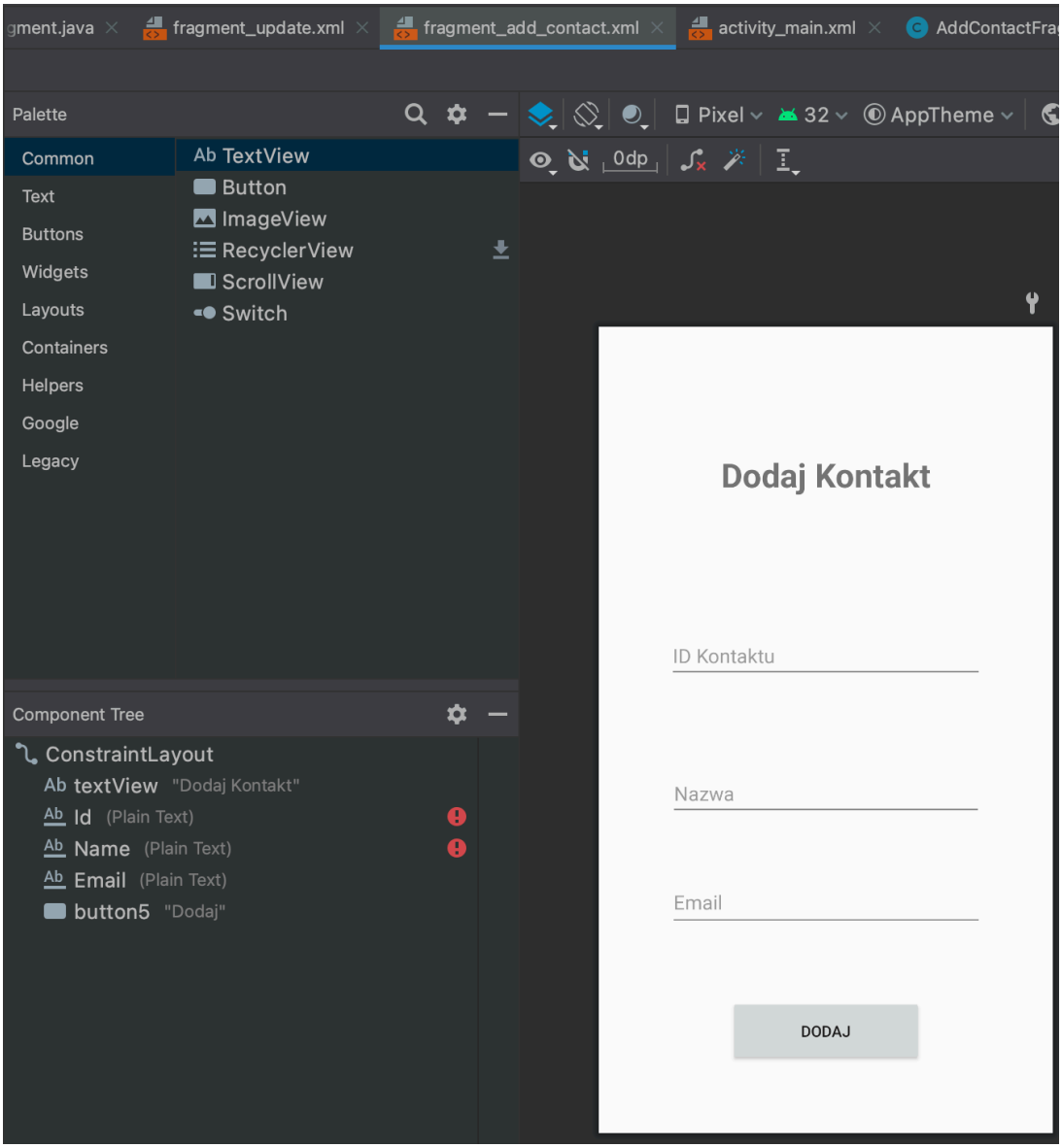


Kod XML:

```
<?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"
```

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



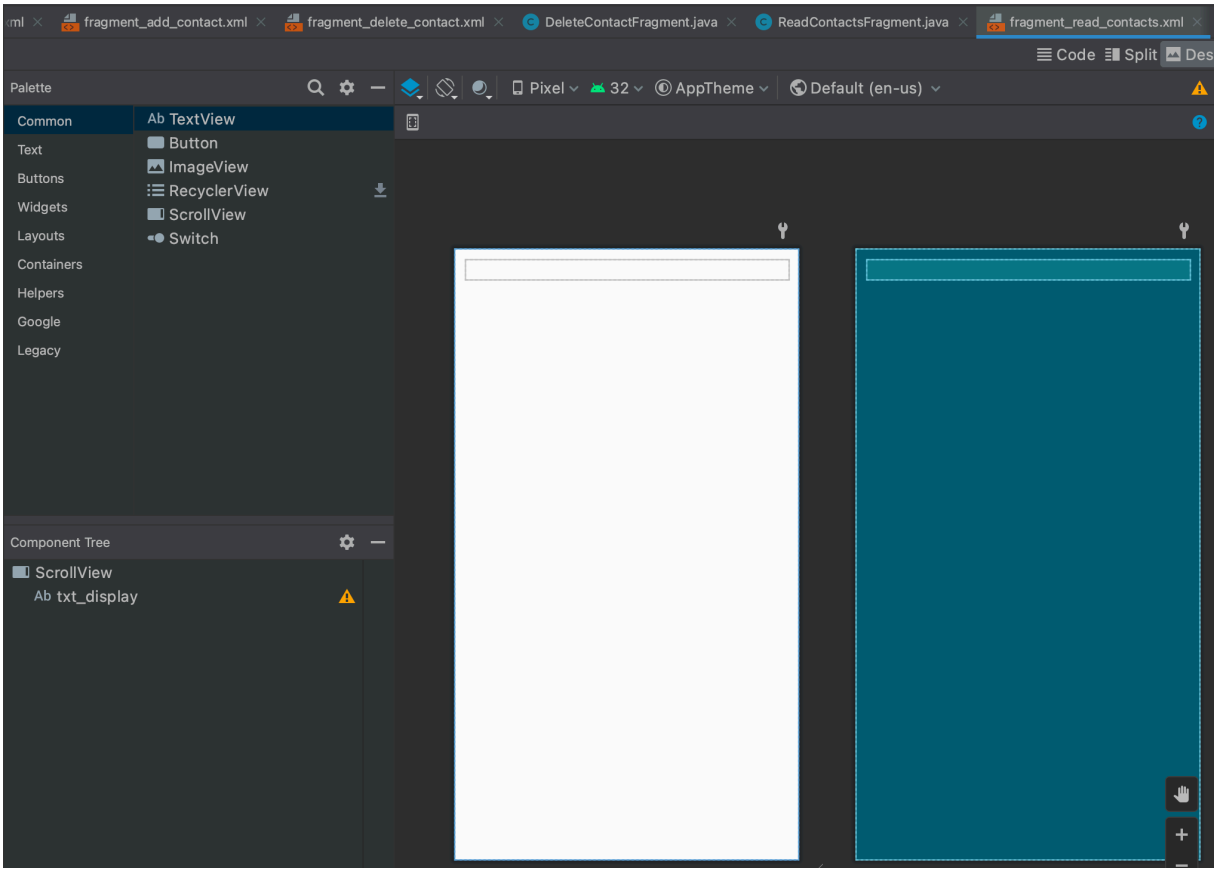
Kod fragment_add_contact.xml:

```
<?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>
```

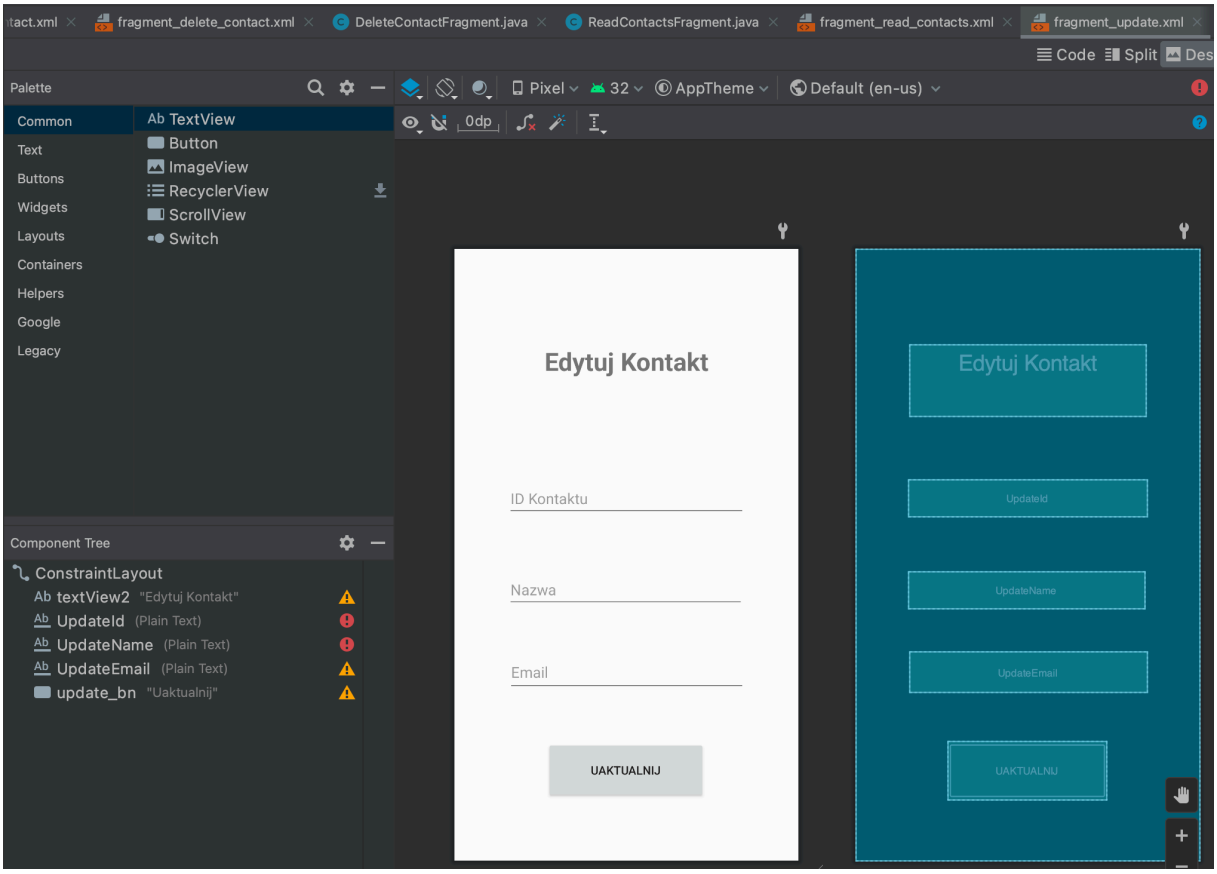
3. Tworzymy kolejny fragment: ReadContactsFragment, który używa komponentu ScrollView i będzie służył do wyświetlania zapisanych kontaktów:




```
<?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:

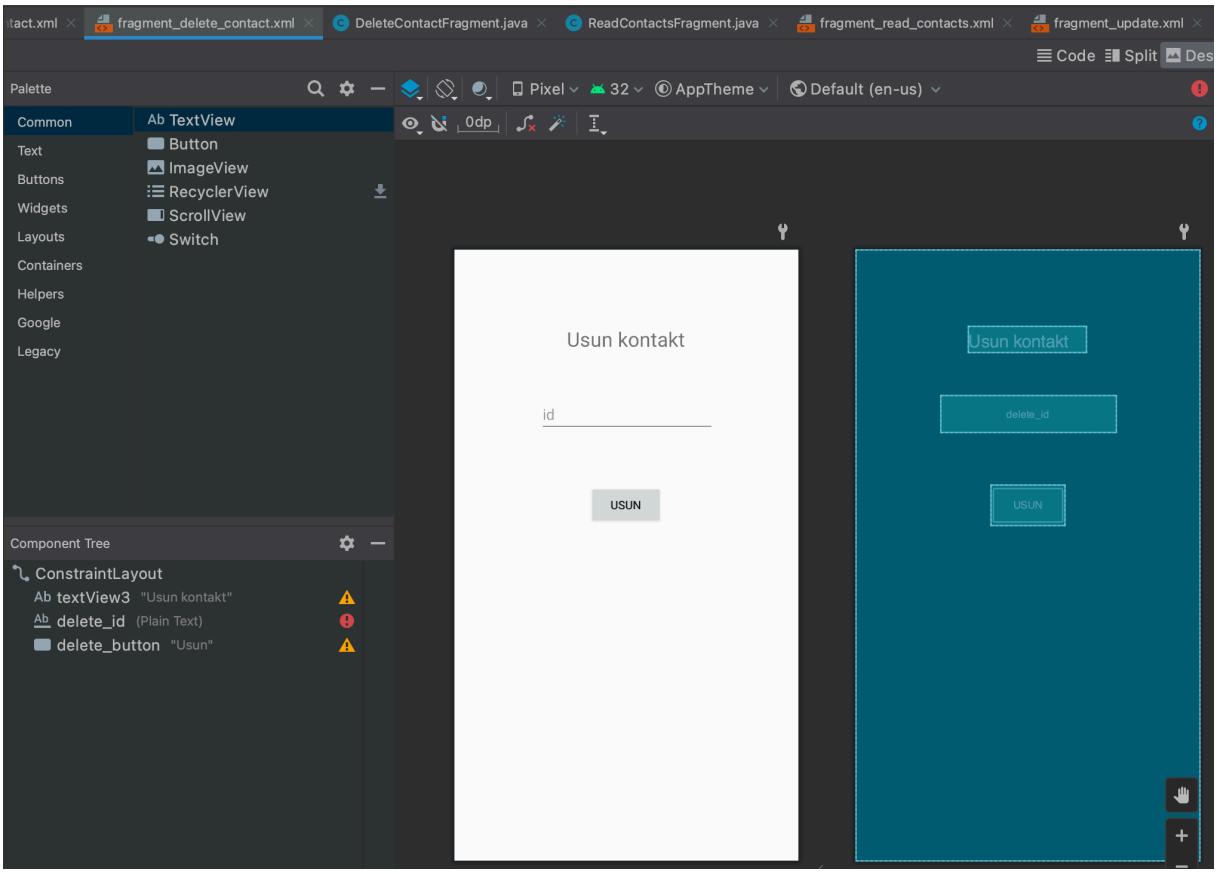



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

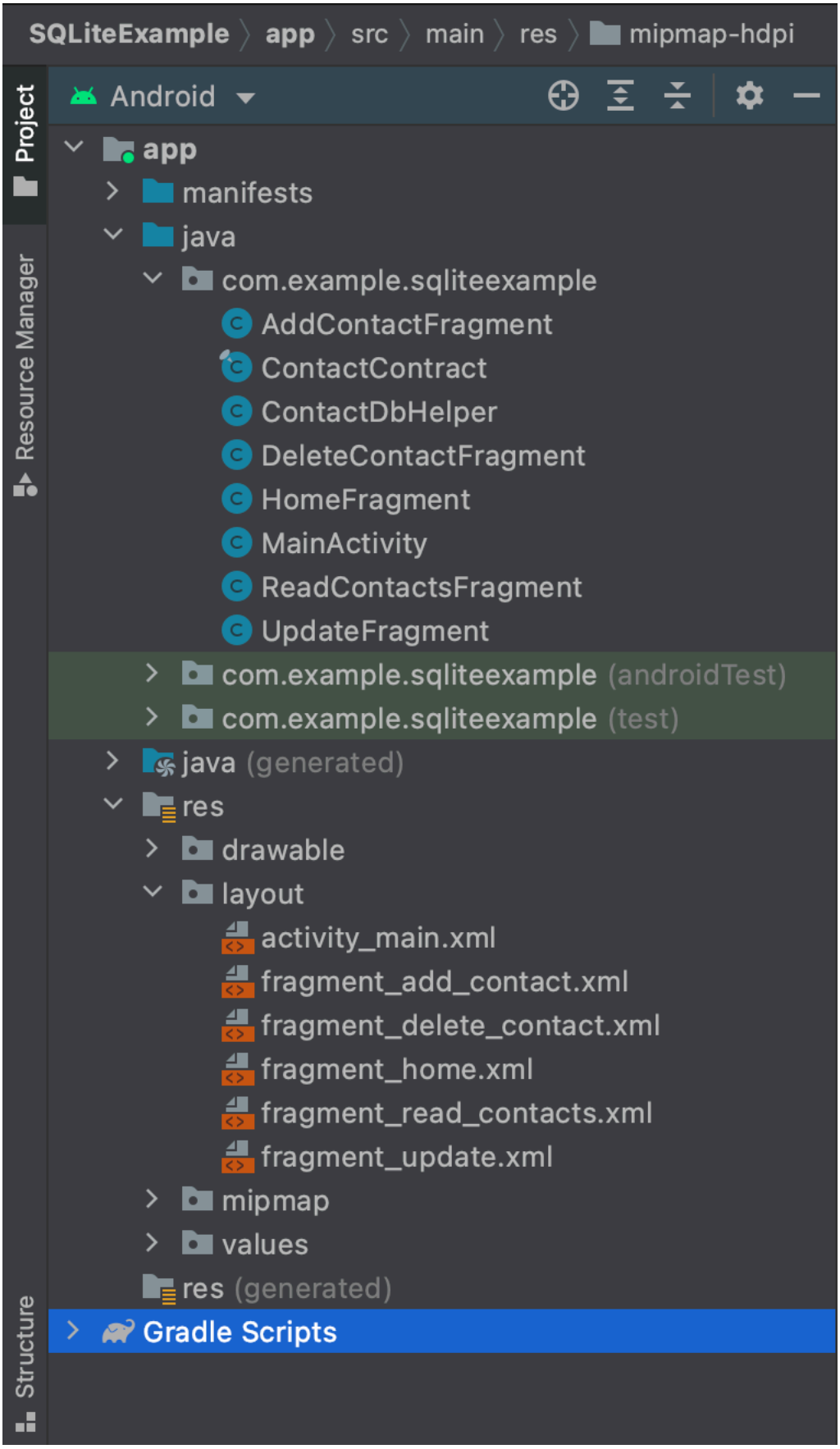
5. Ostatni fragment, DeleteFragment, który będzie dawał możliwość usunięcia elementu z bazy danych:



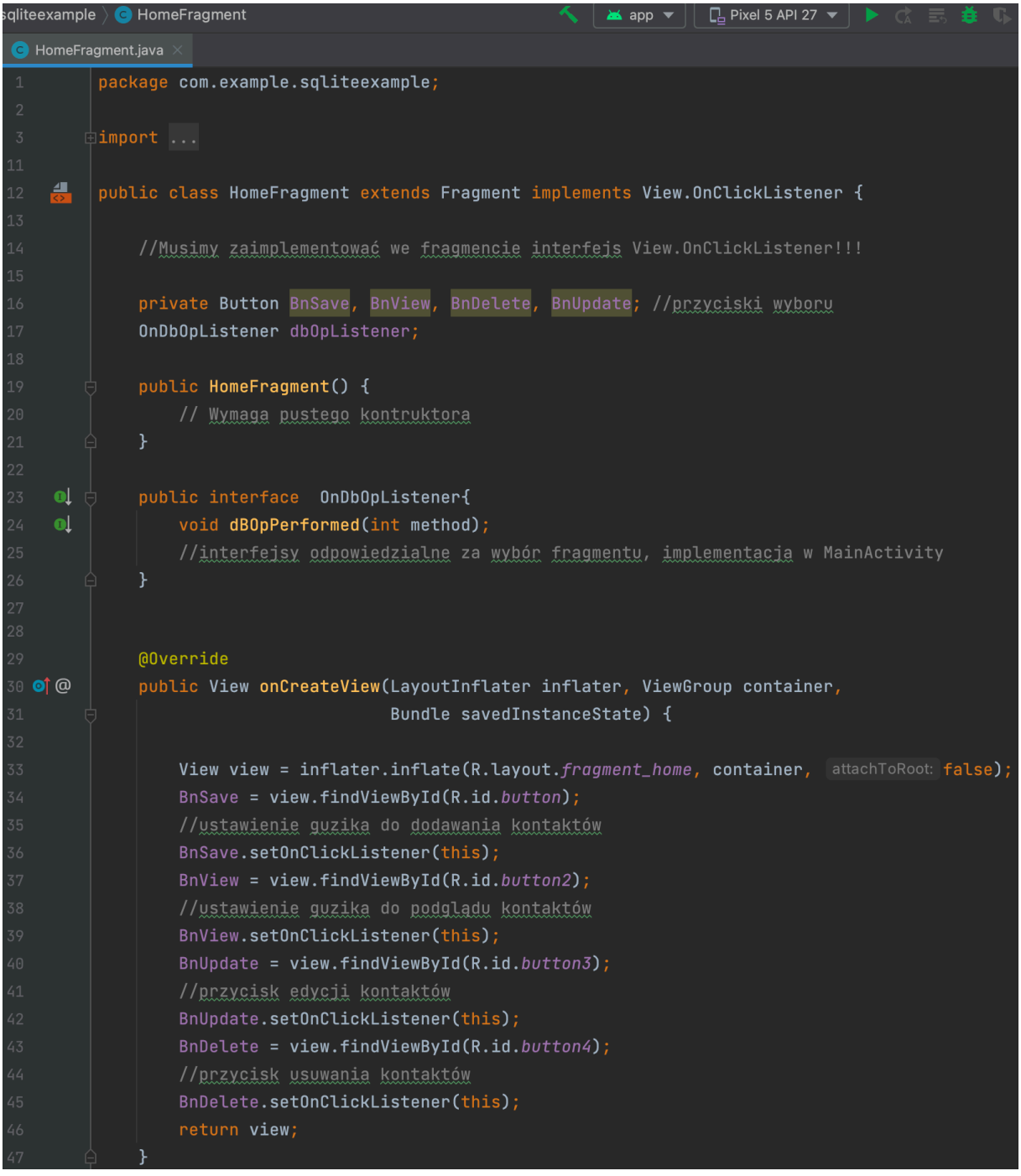

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    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: (ContanctContract oraz ContactDbHelper jeszcze nie ma, stworzymy je w późniejszych krokach)


SQLiteExample
app
src
main
res
mipmap-hdpi
Project
Resource Manager
Structure
Android
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)
Gradle Scripts

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




```java
package com.example.sqliteexample;

import ...

public class HomeFragment extends Fragment implements View.OnClickListener {

    //Musimy zaimplementować we fragmencie interfejs View.OnClickListener!!!

    private Button BnSave, BnView, BnDelete, BnUpdate; //przyciski wyboru
    OnDbOpListener dbOpListener;

    public HomeFragment() {
        // Wymaga pustego kontruktora
    }

    public interface  OnDbOpListener{
        void dBOpPerformed(int method);
        //interfejsy odpowiedzialne za wybór fragmentu, implementacja w MainActivity
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_home, container,  attachToRoot: false);
        BnSave = view.findViewById(R.id.button);
        //ustawienie guzika do dodawania kontaktów
        BnSave.setOnClickListener(this);
        BnView = view.findViewById(R.id.button2);
        //ustawienie guzika do podgladu kontaktów
        BnView.setOnClickListener(this);
        BnUpdate = view.findViewById(R.id.button3);
        //przycisk edycji kontaktów
        BnUpdate.setOnClickListener(this);
        BnDelete = view.findViewById(R.id.button4);
        //przycisk usuwania kontaktów
        BnDelete.setOnClickListener(this);
        return view;
    }
```

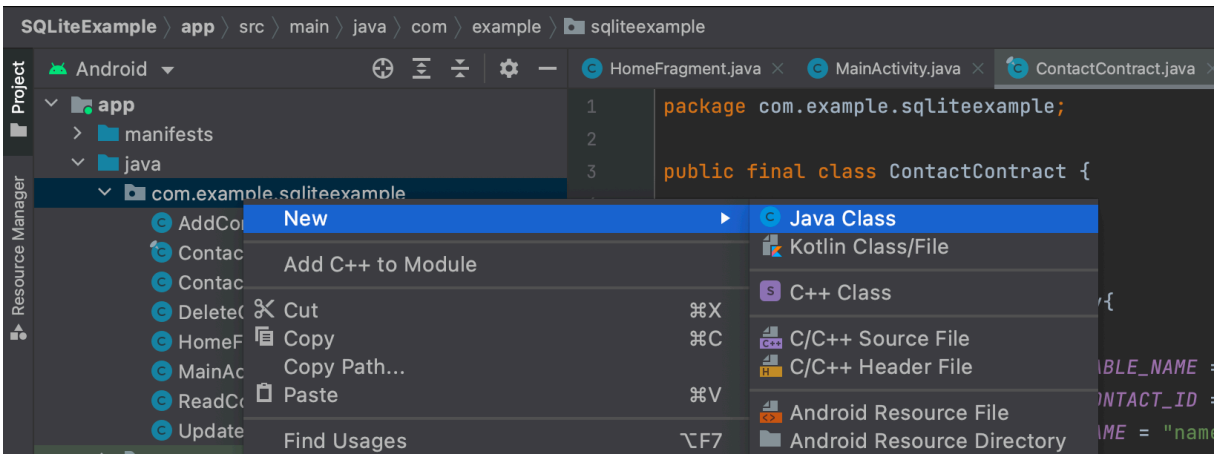
Musimy jeszcze przesłonić metodę onClick tak, aby na podstawie naciśniętego przycisku wysyłać odpowiednią informację do MainActivity:

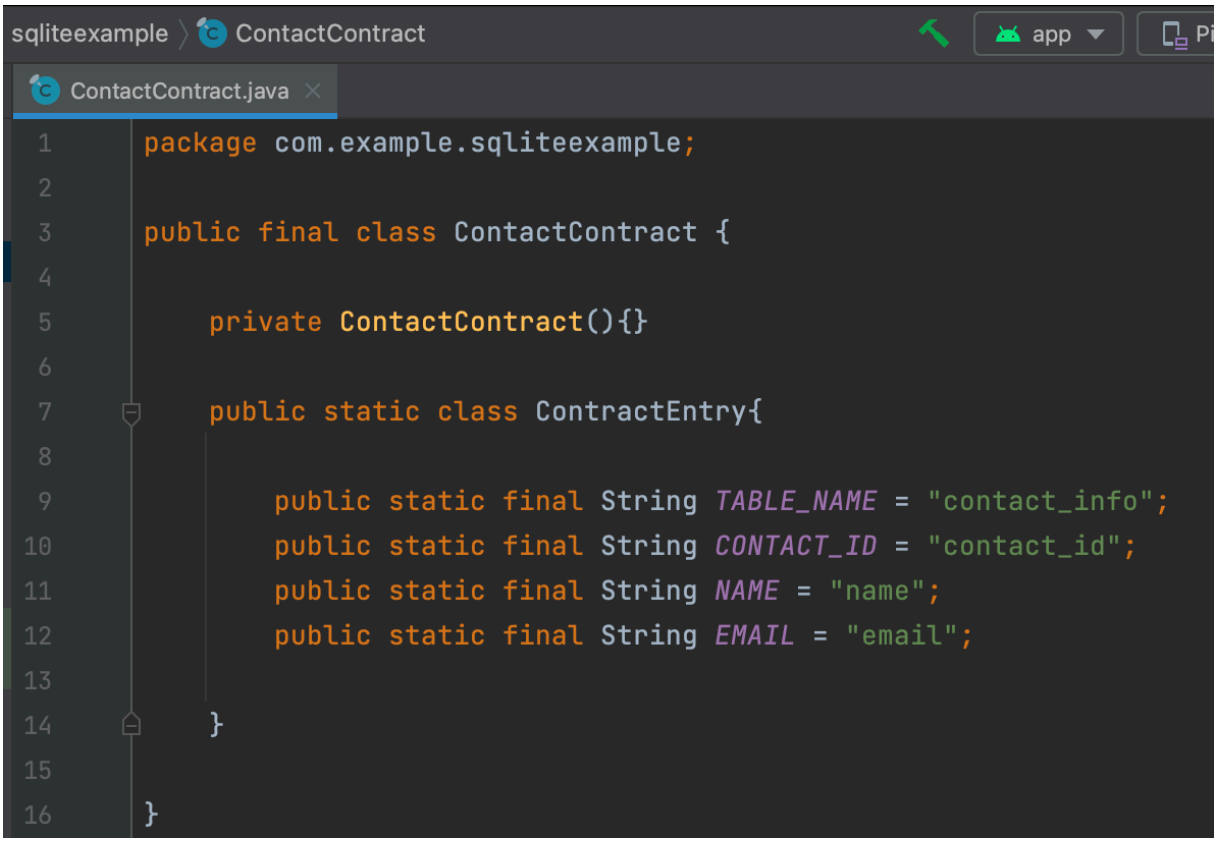
```
48
49      @Override
50  @   public void onClick(View v) {
51          switch (v.getId()){
52              // Jeśli zostanie naciśnięty przycisk dodania kontaktu,
53              // Zostaje ustawiona wartośc 0 odczytywana później w MainActivity.
54              // Zastąpiony zostaje wtedy fragment
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)

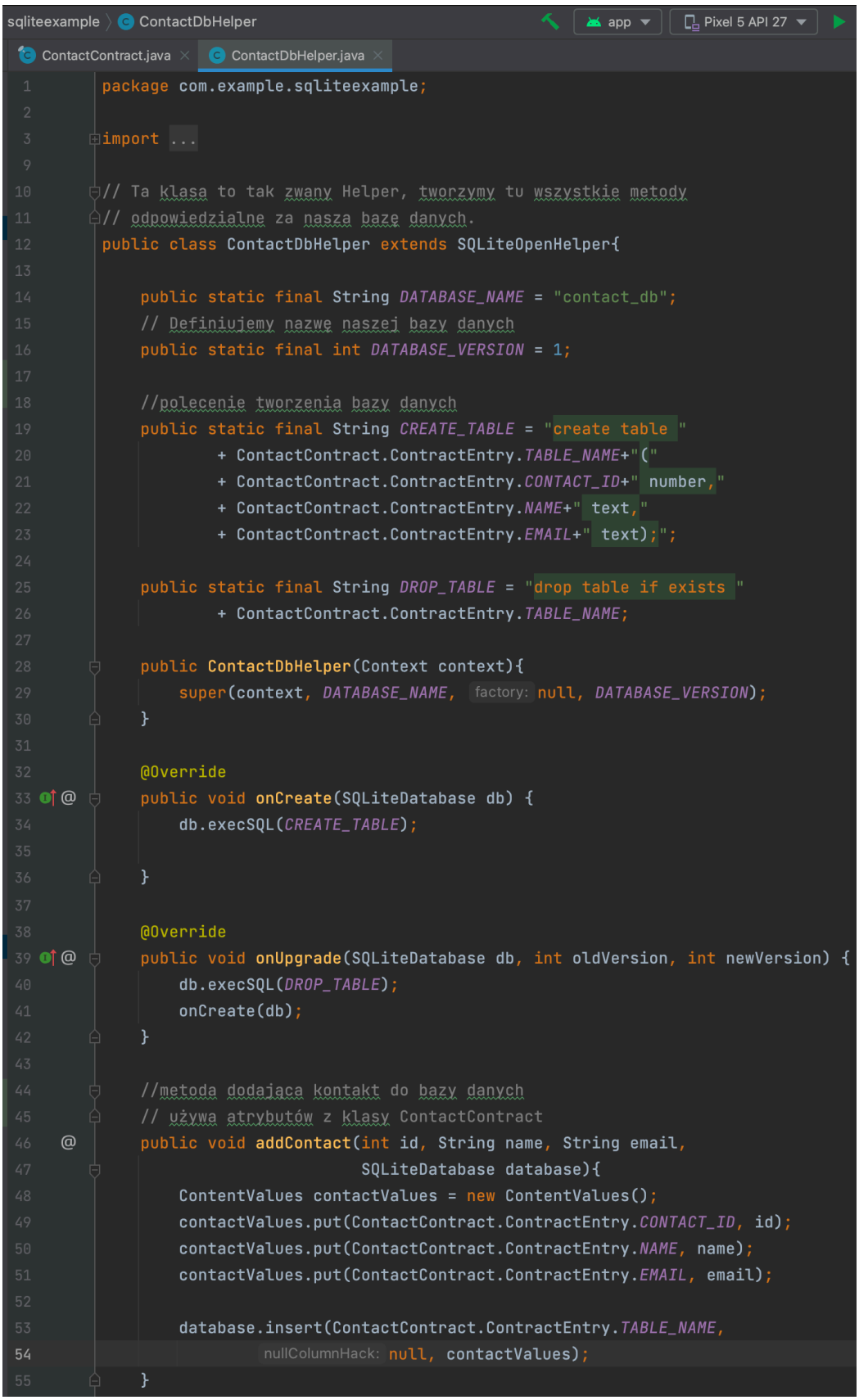

sqliteexample
MainActivity
app
Pixel 5 API 27
HomeFragment.java
MainActivity.java
package com.example.sqliteexample;
import ...
public class MainActivity extends AppCompatActivity implements HomeFragment.OnDbOpListener {
//Pamiętajmy, że implementujemy interfejs z HomeFragment!!!!
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
if(findViewById(R.id.fragment_container)!=null){
if(savedInstanceState!=null){
return;
}
HomeFragment homeFragment = new HomeFragment();
getSupportFragmentManager().beginTransaction()
.add(R.id.fragment_container, homeFragment).commit();
}
}
@Override
public void dBOpPerformed(int method) {
switch (method){
//ODBIERANIE INFORMACJI O AKTUALNIE WYSWIETLANYM FRAGMENCIE
case 0:
getSupportFragmentManager().beginTransaction()
.replace(R.id.fragment_container,
new AddContactFragment()).addToBackStack(null).commit();
break;
case 1:
getSupportFragmentManager().beginTransaction()
.replace(R.id.fragment_container,
new ReadContactsFragment()).addToBackStack(null).commit();
break;
case 2:
getSupportFragmentManager().beginTransaction()
.replace(R.id.fragment_container,
new UpdateFragment()).addToBackStack(null).commit();
break;
case 3:
getSupportFragmentManager().beginTransaction()
.replace(R.id.fragment_container,
new DeleteContactFragment()).addToBackStack(null).commit();
break;
}
}
}

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

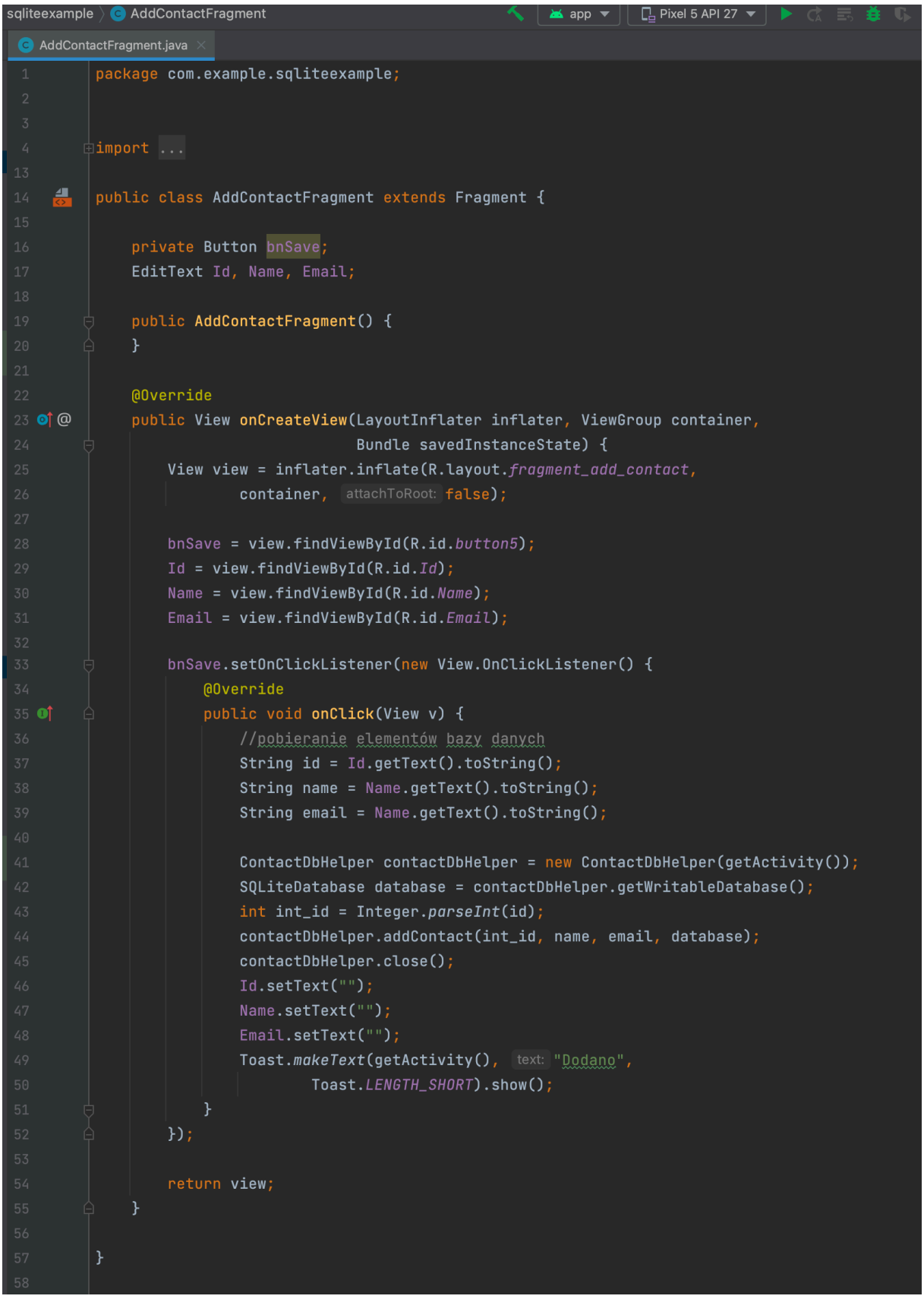




8. Kolejny krok, to stworzonie klasy, która będzie naszą klasą z metodami obsługującymi bazę danych. Większość to po prostu komendy SQL:



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



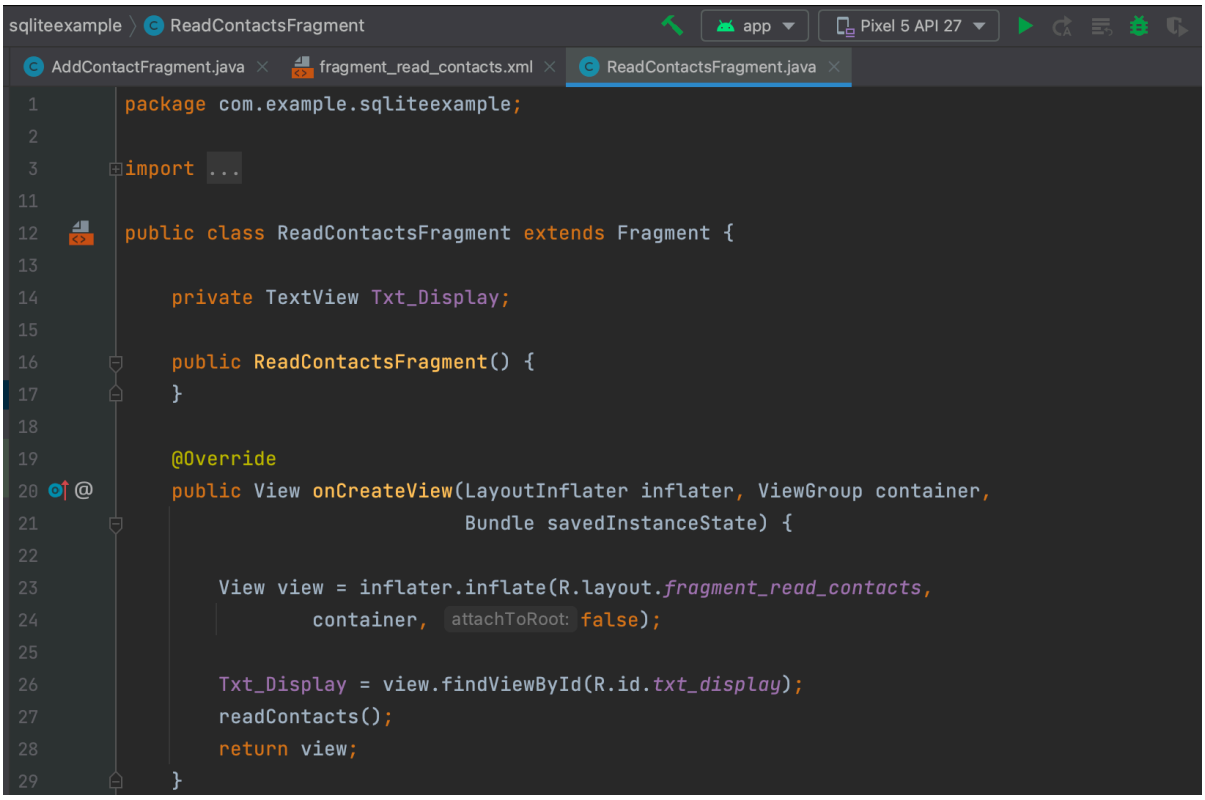

```java
package com.example.sqliteexample;


import ...

public class AddContactFragment extends Fragment {

    private Button bnSave;
    EditText Id, Name, Email;

    public AddContactFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_add_contact,
                container, attachToRoot: false);

        bnSave = view.findViewById(R.id.button5);
        Id = view.findViewById(R.id.Id);
        Name = view.findViewById(R.id.Name);
        Email = view.findViewById(R.id.Email);

        bnSave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //pobieranie elementów bazy danych
                String id = Id.getText().toString();
                String name = Name.getText().toString();
                String email = Name.getText().toString();

                ContactDbHelper contactDbHelper = new ContactDbHelper(getActivity());
                SQLiteDatabase database = contactDbHelper.getWritableDatabase();
                int int_id = Integer.parseInt(id);
                contactDbHelper.addContact(int_id, name, email, database);
                contactDbHelper.close();
                Id.setText("");
                Name.setText("");
                Email.setText("");
                Toast.makeText(getActivity(), text: "Dodano",
                        Toast.LENGTH_SHORT).show();
            }
        });

        return view;
    }

}
```

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:

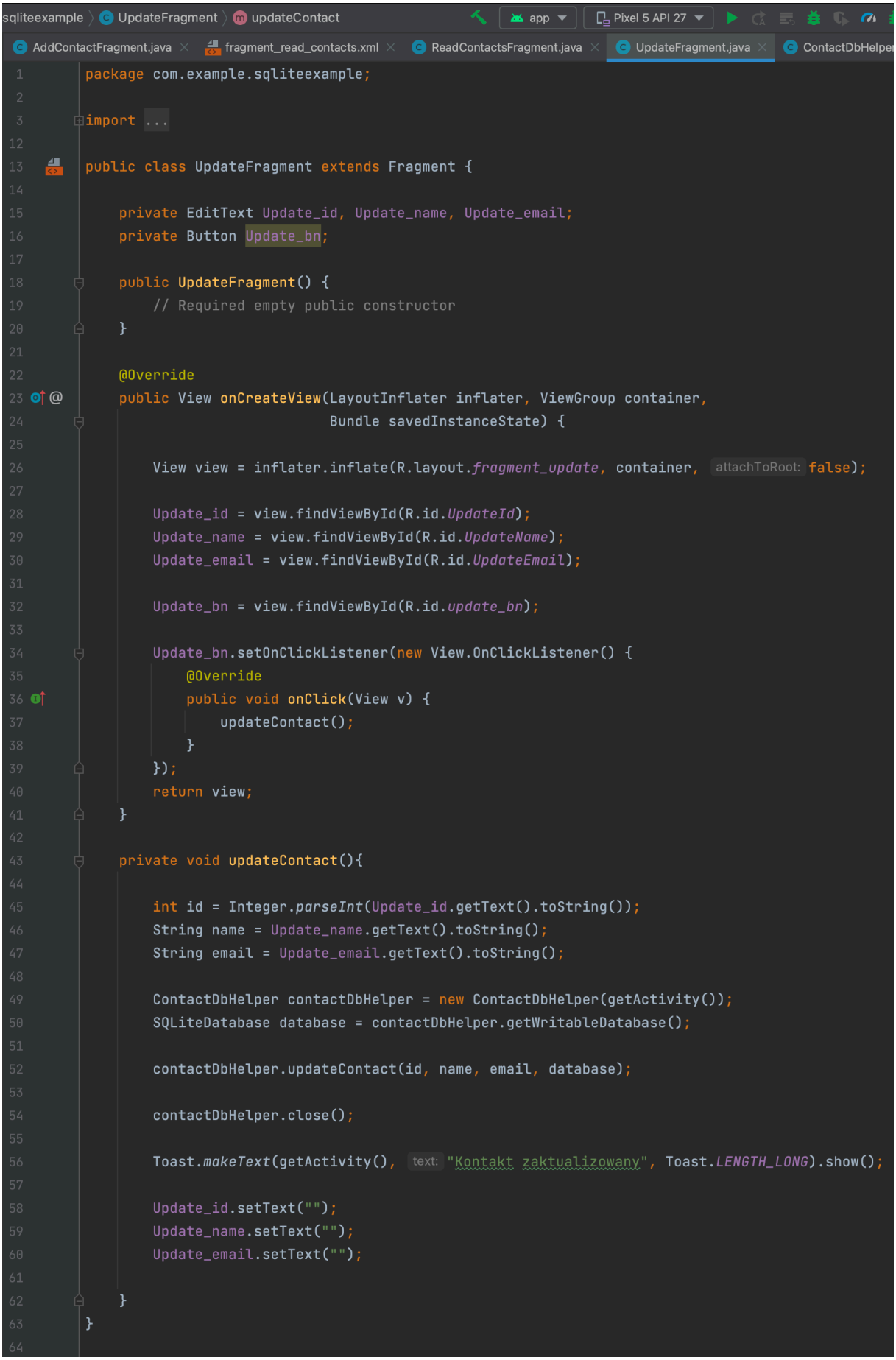


```
    private void readContacts(){
        ContactDbHelper contactDbHelper = new ContactDbHelper(getActivity());
        SQLiteDatabase database = contactDbHelper.getReadableDatabase();


        Cursor cursor = contactDbHelper.readCotacts(database);
        String info = "";

        // Za pomocą petli pobieramy wszystkie wpisy z tabeli
        while (cursor.moveToNext()){
            String id = Integer.toString(cursor.getInt(cursor
                    .getColumnIndex(ContactContract.ContractEntry.CONTACT_ID)));
            String name = cursor.getString(cursor
                    .getColumnIndex(ContactContract.ContractEntry.NAME));
            String email = cursor.getString(cursor
                    .getColumnIndex(ContactContract.ContractEntry.EMAIL));

            info = info +"\n\n"+"Id: "+id+"\nName : "+name+"\nEmail : "+email;
        }

        Txt_Display.setText(info);
        contactDbHelper.close();

    }

}
```

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

```
    @   public void updateContact(int id, String name, String email, SQLiteDatabase database){
            ContentValues contentValues = new ContentValues();
            contentValues.put(ContactContract.ContractEntry.NAME, name);
            contentValues.put(ContactContract.ContractEntry.EMAIL, email);

            String selection = ContactContract.ContractEntry.CONTACT_ID+ " = " +id;

            database.update(ContactContract.ContractEntry
                    .TABLE_NAME, contentValues, selection,  whereArgs: null);
        }
```

Następnie modyfikujemy kod fragmentu UpdateFragment


sqliteexample
UpdateFragment
updateContact
app
Pixel 5 API 27
AddContactFragment.java
fragment_read_contacts.xml
ReadContactsFragment.java
UpdateFragment.java
ContactDbHelper
package com.example.sqliteexample;
import ...
public class UpdateFragment extends Fragment {
private EditText Update_id, Update_name, Update_email;
private Button Update_bn;
public UpdateFragment() {
// Required empty public constructor
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
View view = inflater.inflate(R.layout.fragment_update, container, attachToRoot: false);
Update_id = view.findViewById(R.id.UpdateId);
Update_name = view.findViewById(R.id.UpdateName);
Update_email = view.findViewById(R.id.UpdateEmail);
Update_bn = view.findViewById(R.id.update_bn);
Update_bn.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
updateContact();
}
});
return view;
}
private void updateContact(){
int id = Integer.parseInt(Update_id.getText().toString());
String name = Update_name.getText().toString();
String email = Update_email.getText().toString();
ContactDbHelper contactDbHelper = new ContactDbHelper(getActivity());
SQLiteDatabase database = contactDbHelper.getWritableDatabase();
contactDbHelper.updateContact(id, name, email, database);
contactDbHelper.close();
Toast.makeText(getActivity(), text: "Kontakt zaktualizowany", Toast.LENGTH_LONG).show();
Update_id.setText("");
Update_name.setText("");
Update_email.setText("");
}
}

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                    app    Pixel 5 API 27
ctFragment.java   fragment_read_contacts.xml   ReadContactsFragment.java   UpdateFragment.java   ContactDbHelper.java
1   package com.example.sqliteexample;
2
3
4   import ...
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
```