Sprawozdanie z realizacji projektu aplikacji mobilnej

SimpleNote

W ramach przedmiotu Programowanie Urządzeń Mobilnych

Szymon Grabarkiewicz

Indeks: **38558**

Prowadzący: **dr Aleksander Klosow**

Legnica

5 stycznia 2021

Spis treści

Rozdział 1 – Opis aplikacji

Rozdział 2 – Przegląd technologii i narzędzi użytych w projekcie

Rozdział 3 – Interfejs aplikacji

Rozdział 4 – Fragmenty kodu

Rozdział 5 – Link do aplikacji

Rozdział 1 - Opis aplikacji

SimpleNote jest aplikacją umożliwiającą tworzenie i zapisywanie swoich własnych notatek. Aplikacja została stworzona na system Android za pomocą narzędzia Android Studio.

Aplikacja umożliwia tworzenie własnych notatek, które mogą być później edytowane i usuwane. Po dodaniu notatki widoczna jest data dodania. Notatki są wyświetlane w kolejności dodania.

Notatki są zapisywane w bazie danych na telefonie za pomocą SQLite, dzięki czemu nawet po wyłączeniu aplikacji będą dostępne.

Rozdział 2 – przegląd technologii i narzędzi użytych w projekcie

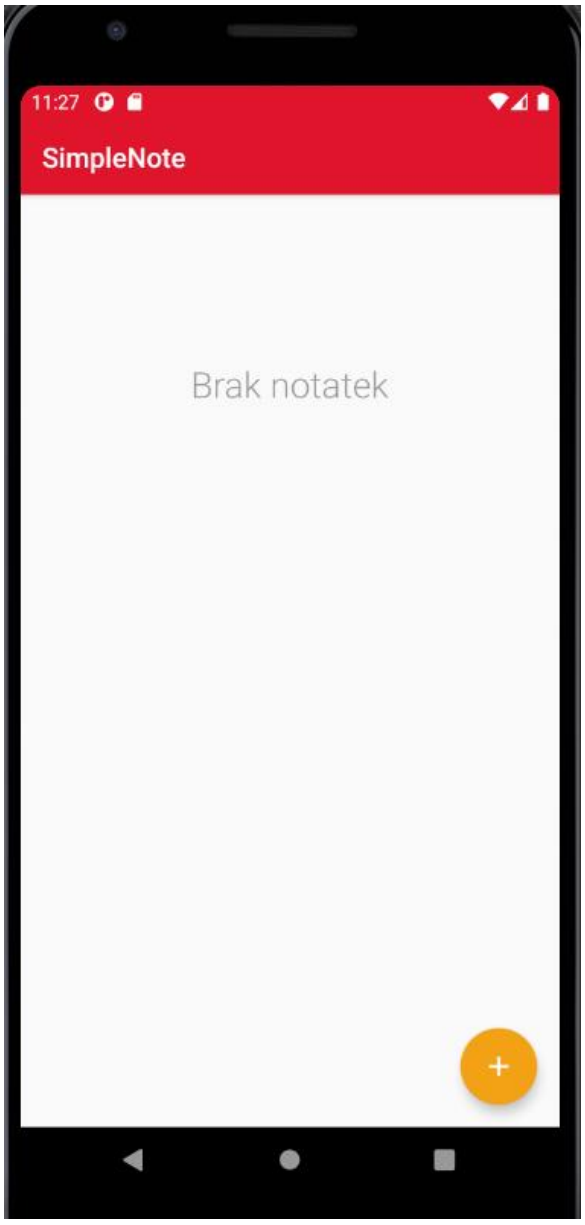
W projekcie zostały wykorzystane następujące technologie:

1. **Java** - współbieżny, oparty na klasach, obiektowy język programowania ogólnego zastosowania. W przeciwieństwie do wieloparadygmatowego języka C++, Java jest silnie ukierunkowana na programowanie obiektowe. Od maja 2019 firma Google rekomenduje, aby wszystkie najnowsze aplikacje na Androida pisać w języku Kotlin, jednak Java nadal jest wspierana.
2. **Android Studio** - to oficjalne zintegrowane środowisko programistyczne dla systemu operacyjnego Android firmy Google, zbudowane na oprogramowaniu JetBrains IntelliJ IDEA i zaprojektowane specjalnie na potrzeby rozwoju Androida.

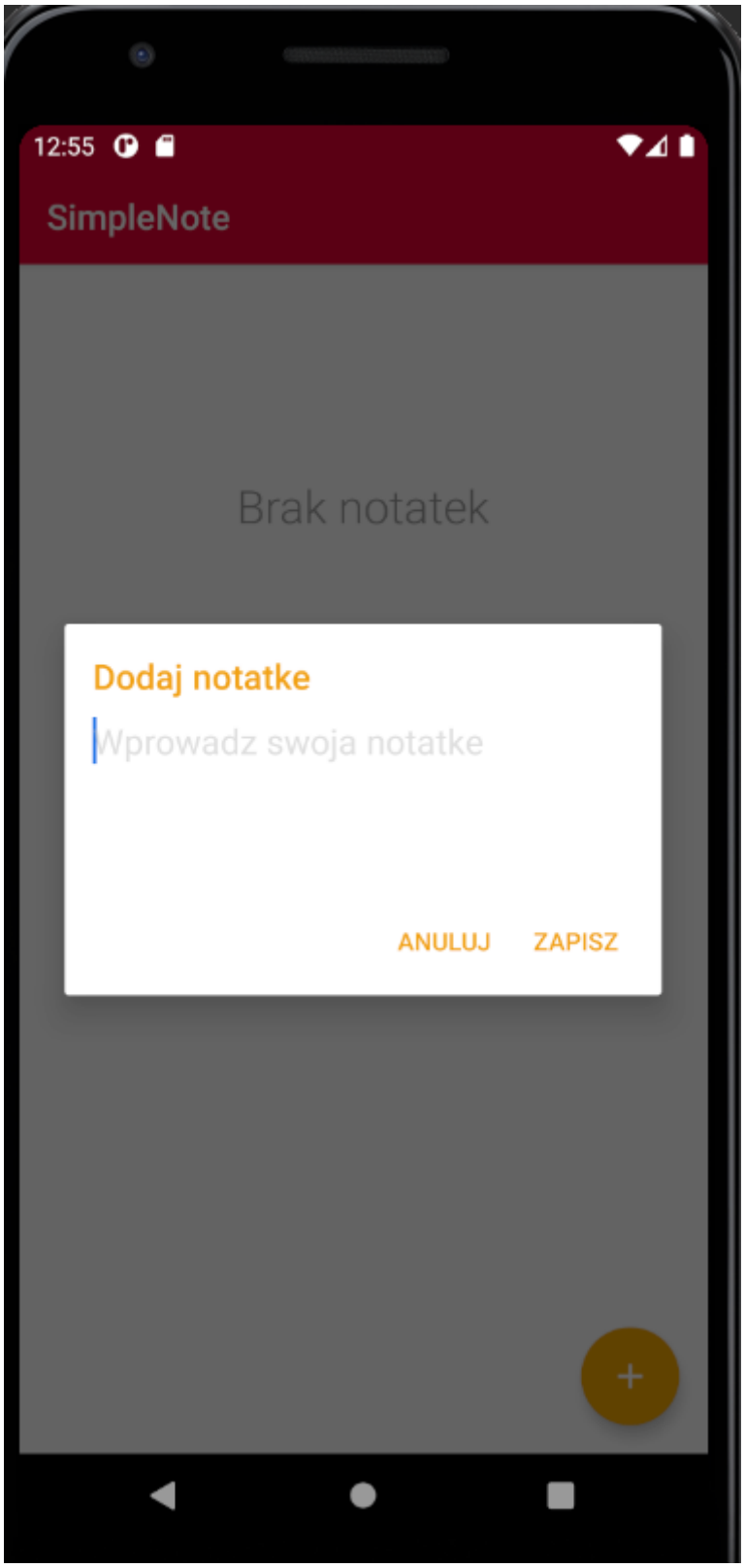
3. **Baza danych SQLite** – System Android aktualnie dostarcza nam kilka mechanizmów związanych z przechowywaniem danych, z których każdy ma inne zastosowanie. W projekcie zostało wykorzystane narzędzie SQLite wykorzystywane do przechowywania dużej ilości uporządkowanych danych, które ze względu na swoja ilość wymagają wysokiej wydajności dostępu.

Rozdział 3 – Interfejs aplikacji

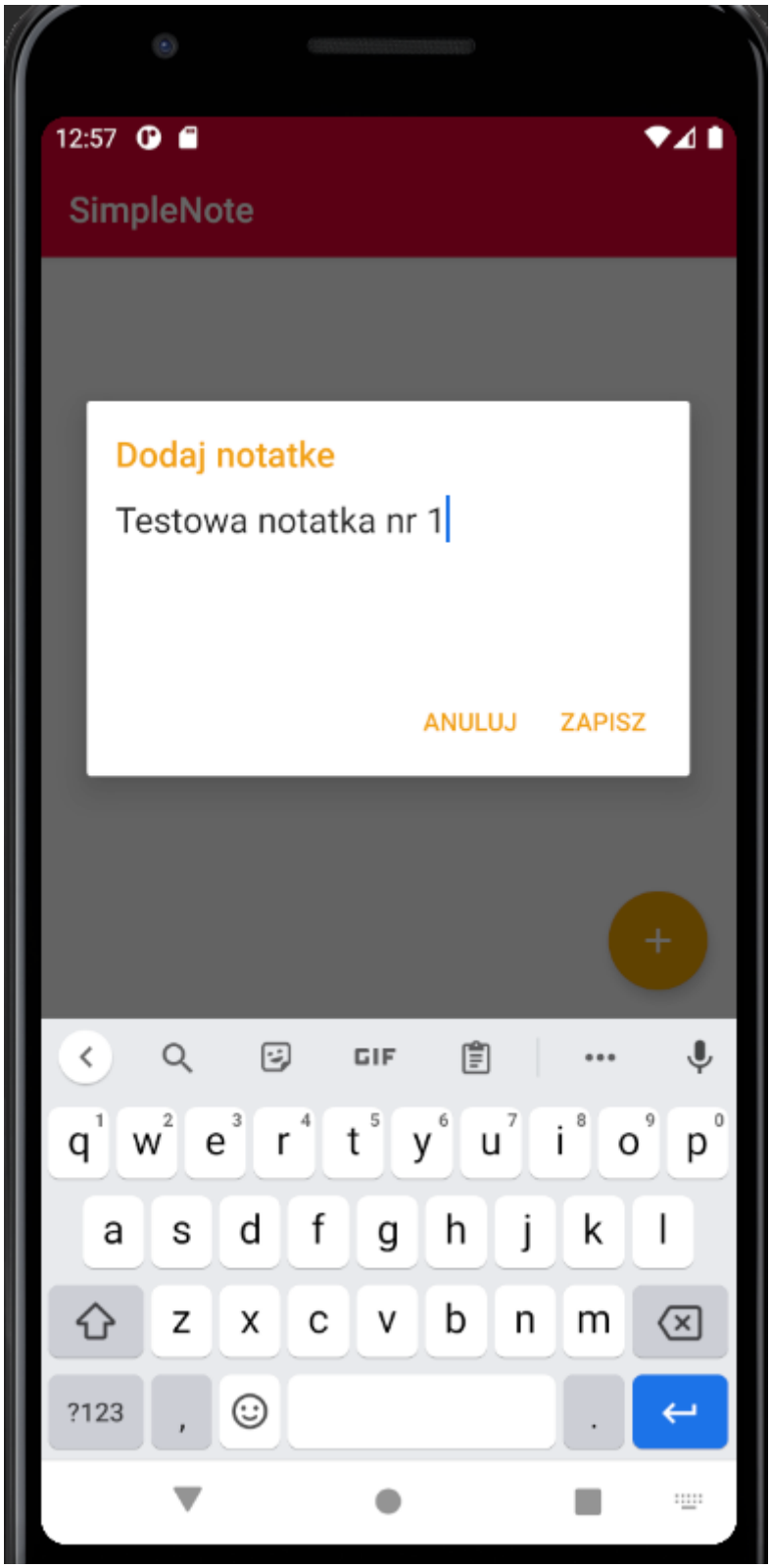
Menu główne aplikacji:



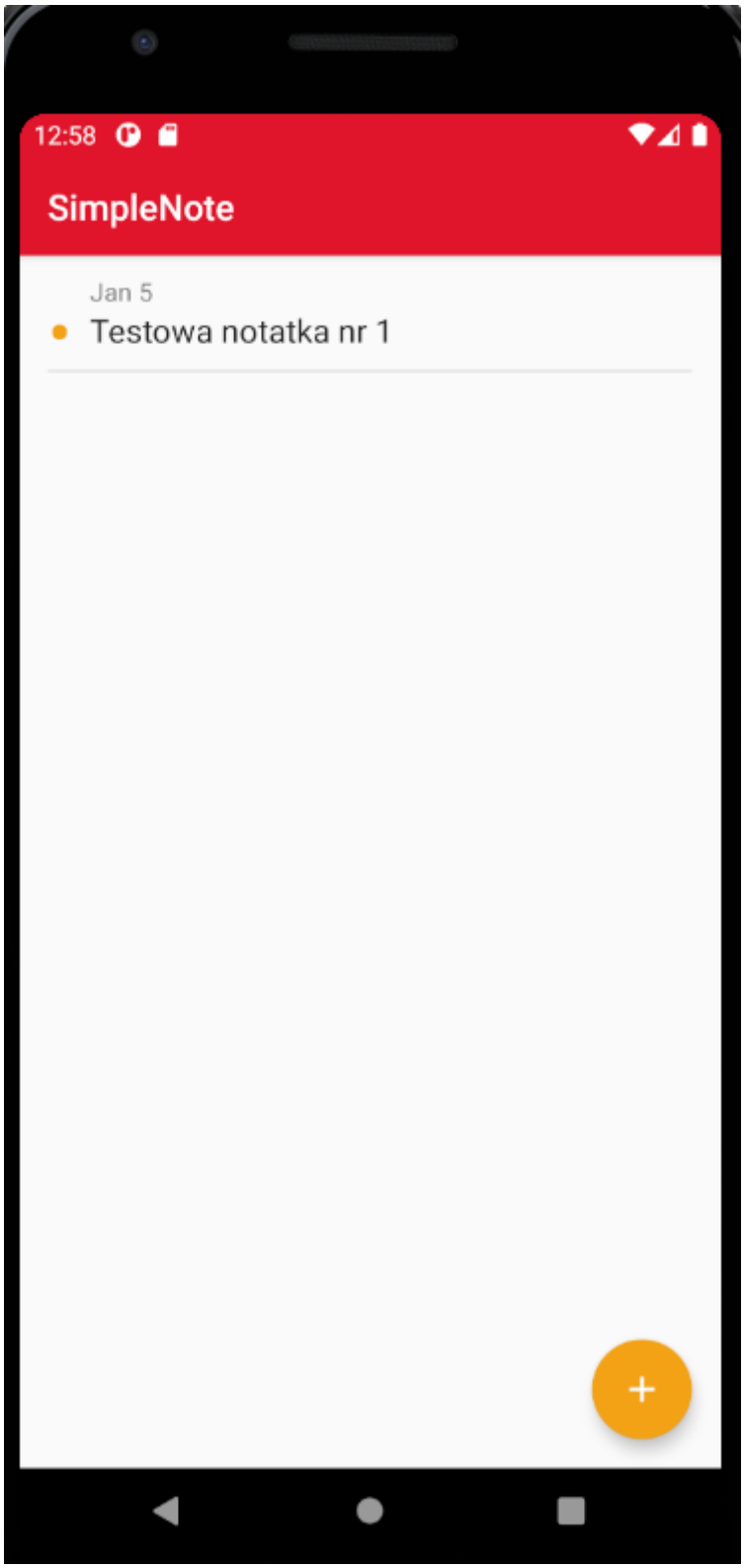
Ekran dodawania nowej notatki



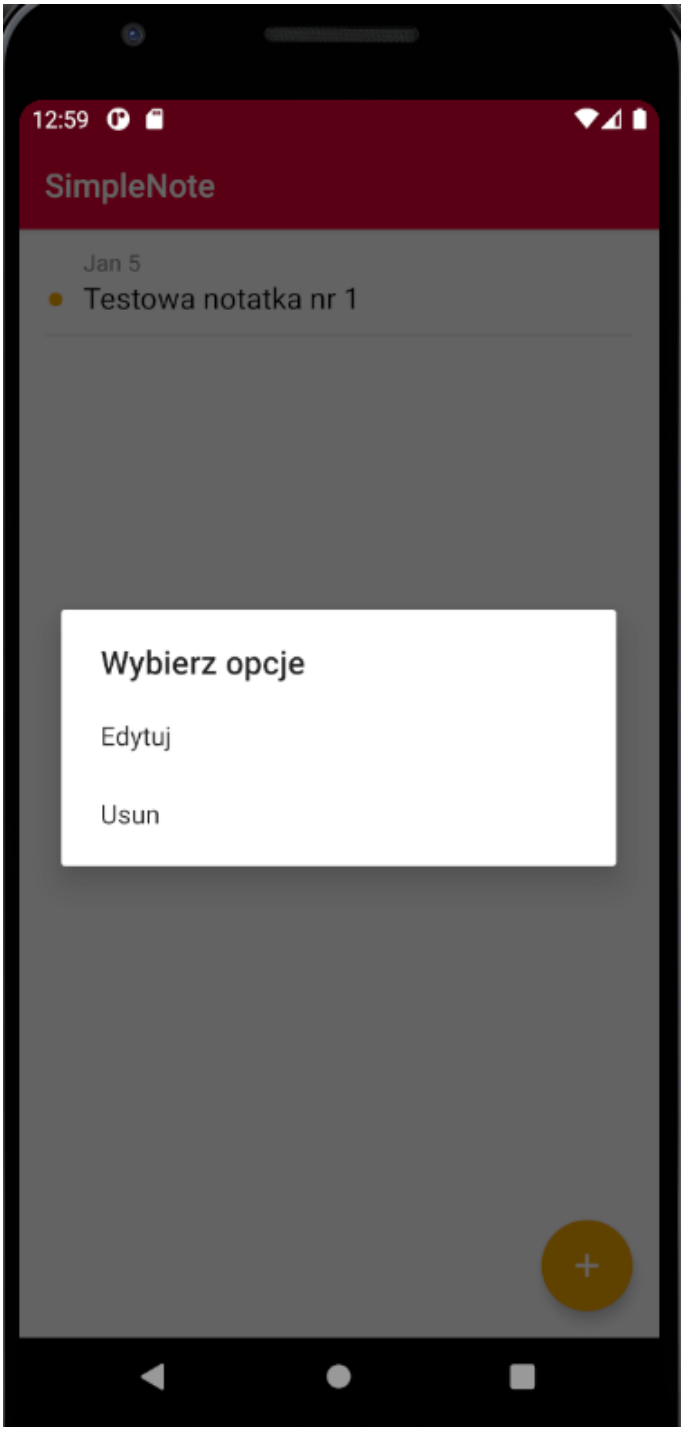
Wpisywanie tekstu do notatki:



Widok dodanej notatki:

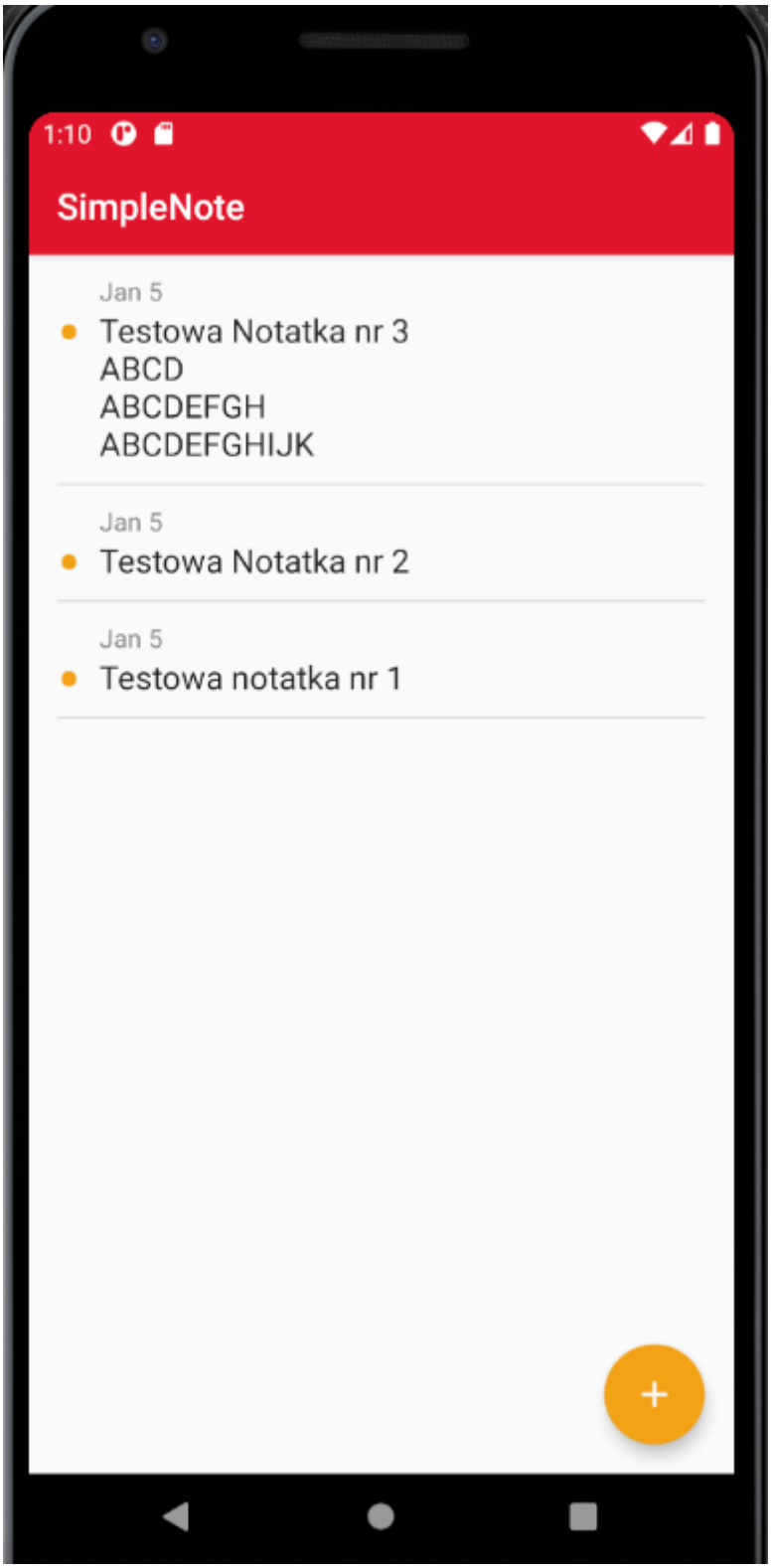


Widok edycji lub usunięcia notatki (należy przytrzymać palec na notatce przez ponad sekundę, aby pojawiło się okno):



Po wybraniu opcji edycji pojawia się takie samo okno jak w przypadku wpisywania tekstu do nowej notatki. Opcja „usuń” usuwa notatkę.

Widok kilku dodanych notatek:



Rozdział 4 – fragmenty kodu

Główny widok z okienkiem, które pojawia się po dłuższym naciśnięciu na notatkę (okno edycji i usunięcia):

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        coordinatorLayout = findViewById(R.id.coordinator_layout);
        recyclerView = findViewById(R.id.recycler_view);
        noNotesView = findViewById(R.id.empty_notes_view);

        db = new DatabaseHelper( context: this);

        notesList.addAll(db.getAllNotes());

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener((view) -> { showNoteDialog( shouldUpdate: false,  note: null,  position: -1); });

        mAdapter = new NotesAdapter( context: this, notesList);
        RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());
        recyclerView.setLayoutManager(mLayoutManager);
        recyclerView.setItemAnimator(new DefaultItemAnimator());
        recyclerView.addItemDecoration(new MyDividerItemDecoration( context: this, LinearLayoutManager.VERTICAL,  margin: 16));
        recyclerView.setAdapter(mAdapter);

        toggleEmptyNotes();


//        Długie naciśnięcie elementu RecyclerView powoduje otwarcie okna dialogowego ostrzeżenia
//        z opcjami do wyboru: Edytuj i usuń
        recyclerView.addOnItemTouchListener(new RecyclerTouchListener( context: this,
                recyclerView, new RecyclerTouchListener.ClickListener() {
            @Override
            public void onClick(View view, final int position) {
            }

            @Override
            public void onLongClick(View view, int position) { showActionsDialog(position); }
        }));
    }
```

Dodawanie i aktualizacja notatki w bazie danych:

```
//  Wstawianie nowej dodatki w bazie danych i odświeżanie listy
    private void createNote(String note) {
        // wstawienie notatki do bazy danych i pobranie
        // nowo wstawiony identyfikator notatki
        long id = db.insertNote(note);

        // pobierz nowo wstawioną notatkę z bazy danych
        Note n = db.getNote(id);

        if (n != null) {
            // dodanie nowej notatki do listy tablic na pozycji 0
            notesList.add( i: 0, n);

            // odsiwezenie listy
            mAdapter.notifyDataSetChanged();

            toggleEmptyNotes();
        }
    }

//      Aktualizacja notatki w bazie danych i aktualizacja
//      pozycji na liście
    private void updateNote(String note, int position) {
        Note n = notesList.get(position);
        // aktualizacja tekstu notatki
        n.setNote(note);

        // aktualizacja notatki w bazie danych
        db.updateNote(n);

        // odswiezenie listy
        notesList.set(position, n);
        mAdapter.notifyItemChanged(position);

        toggleEmptyNotes();
    }
```

Usuwanie i otwarcie okna z możliwością edycji notatki:

```
//Ustawianie otatki z SQLite i usuwanie pozycji z listy
private void deleteNote(int position) {
    // usunięcie notatki z bazy danych
    db.deleteNote(notesList.get(position));

    // usunięcie notatki z listy
    notesList.remove(position);
    mAdapter.notifyItemRemoved(position);

    toggleEmptyNotes();
}

//Otwarcie okna z opcje usuniecia i edycji notatki
private void showActionsDialog(final int position) {
    CharSequence colors[] = new CharSequence[]{"Edytuj", "Usun"};

    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Wybierz opcje");
    builder.setItems(colors, (dialog, which) -> {
            if (which == 0) {
                showNoteDialog( shouldUpdate: true, notesList.get(position), position);
            } else {
                deleteNote(position);
            }
    });
    builder.show();
}
```

Widok na początek kodu bazy danych

```java
public class DatabaseHelper extends SQLiteOpenHelper {

    // Wersja bazy danych
    private static final int DATABASE_VERSION = 1;

    // Nazwa bazy danych
    private static final String DATABASE_NAME = "notes_db";


    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION);
    }

    // Tworzenie tabeli
    @Override
    public void onCreate(SQLiteDatabase db) {


        db.execSQL(Note.CREATE_TABLE);
    }

    // Aktualizacja bazy
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

        db.execSQL("DROP TABLE IF EXISTS " + Note.TABLE_NAME);


        onCreate(db);
    }
```

```java
// Aktualizacja bazy
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    db.execSQL("DROP TABLE IF EXISTS " + Note.TABLE_NAME);


    onCreate(db);
}

public long insertNote(String note) {
    // pobiez baze danych, poniewaz chcemy zapisac dane
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    // `id` i `timestamp` jest zwiekszane automatycznie.
    values.put(Note.COLUMN_NOTE, note);

    // dodaj wiersz
    long id = db.insert(Note.TABLE_NAME,  nullColumnHack: null, values);

    // zamknij baze danych
    db.close();

    return id;
}
```

Pobieranie bazy danych i aktualizacja

```java
public Note getNote(long id) {
    // pobranie bazy danych
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.query(Note.TABLE_NAME,
            new String[]{Note.COLUMN_ID, Note.COLUMN_NOTE, Note.COLUMN_TIMESTAMP},
            selection: Note.COLUMN_ID + "=?",
            new String[]{String.valueOf(id)}, groupBy: null, having: null, orderBy: null, limit: null);

    if (cursor != null)
        cursor.moveToFirst();

    // przygotowanie obiektu notatki
    Note note = new Note(
            cursor.getInt(cursor.getColumnIndex(Note.COLUMN_ID)),
            cursor.getString(cursor.getColumnIndex(Note.COLUMN_NOTE)),
            cursor.getString(cursor.getColumnIndex(Note.COLUMN_TIMESTAMP)));


    cursor.close();

    return note;
}
```

```java
public List<Note> getAllNotes() {
    List<Note> notes = new ArrayList<>();

    // Zaznacz wszystkie zapytania
    String selectQuery = "SELECT  * FROM " + Note.TABLE_NAME + " ORDER BY " +
            Note.COLUMN_TIMESTAMP + " DESC";

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery,  selectionArgs: null);

    // przelgądanie wszystkich wierszy i dodanie do listy
    if (cursor.moveToFirst()) {
        do {
            Note note = new Note();
            note.setId(cursor.getInt(cursor.getColumnIndex(Note.COLUMN_ID)));
            note.setNote(cursor.getString(cursor.getColumnIndex(Note.COLUMN_NOTE)));
            note.setTimestamp(cursor.getString(cursor.getColumnIndex(Note.COLUMN_TIMESTAMP)));

            notes.add(note);
        } while (cursor.moveToNext());
    }


    db.close();

    // zwrocenie listy notatek
    return notes;
}
```

Szczegółowy kod i program można znaleźć pod adresem:

https://github.com/Eszy94/SimpleNote.git

W celu zainstalowania i uruchomienia aplikacji należy posiadać najnowszą wersję Android Studio (z grudnia 2020 roku) i Java SDK w wersji 15.

Aplikacja działa zarówno na wirtualnym urządzeniu, jak i na urządzeniu fizycznym, po uprzednim skonfigurowaniu telefonu w tryb deweloperski.