UAIM

Laboratorium 5

Karol Żelazowski 324953

Spis treści

1. Opis kodu

1.1. Klasy w Javie

1.1.1. Klasa MainActivity

```Java
package com.example.uaim_lab;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
import java.util.Date;

public class MainActivity extends AppCompatActivity {
    private ArrayList<Task> tasks;
    private ArrayAdapter<Task> adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tasks = new ArrayList<>();
        tasks.add(new Task("Zadanie 1", "Opis zadania 1", new
Date(System.currentTimeMillis() + 3600000), "Niewykonane"));
        tasks.add(new Task("Zadanie 2", "Opis zadania 2", new
Date(System.currentTimeMillis() - 3600), "Niewykonane"));

        for (Task task : tasks) {
            if (task.getDeadline().before(new Date()) &&
task.getStatus().equals("Niewykonane")) {
                task.setStatus("Przeterminowane");
            }
        }

        ListView listView = findViewById(R.id.listView);
        adapter = new TaskAdapter(this, tasks);
        listView.setAdapter(adapter);

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
```

```
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
                Intent intent = new Intent(MainActivity.this,
TaskDetailActivity.class);
                Task task = tasks.get(position);
                intent.putExtra("title", task.getTitle());
                intent.putExtra("description", task.getDescription());
                intent.putExtra("deadline", task.getDeadline().getTime());
                intent.putExtra("status", task.getStatus());
                intent.putExtra("position", position);
                startActivityForResult(intent, 1);
            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 1 && resultCode == RESULT_OK) {
            int position = data.getIntExtra("position", -1);
            String status = data.getStringExtra("status");
            if (position != -1) {
                tasks.get(position).setStatus(status);
                updateTaskStatuses();
                adapter.notifyDataSetChanged();
            }
        }
    }

    private void updateTaskStatuses() {
        for (Task task : tasks) {
            if (task.getDeadline().before(new Date()) &&
task.getStatus().equals("Niewykonane")) {
                task.setStatus("Przeterminowane");
            }
        }
    }
}
```

Klasa `MainActivity` jest główną aktywnością aplikacji, odpowiadającą za zarządzanie interfejsem użytkownika i logiką dotyczącą listy zadań. Klasa dziedziczy po `AppCompatActivity`, co pozwala jej korzystać z podstawowych funkcji aktywności w systemie Android.

W metodzie `onCreate`, wykonywanej przy uruchomieniu aplikacji, następuje inicjalizacja kluczowych elementów. Tworzona jest lista zadań (`tasks`), zawierająca przykładowe obiekty klasy `Task`. Zadania te są weryfikowane pod kątem przeterminowania – jeśli ich termin

wykonania już upłynął i ich status to „Niewykonane”, zostają oznaczone jako „Przeterminowane”. Dzięki temu stan zadań jest zawsze aktualny w momencie załadowania aplikacji.

Lista zadań jest prezentowana w komponencie `ListView`, do którego dane są dostarczane przez niestandardowy adapter `TaskAdapter`. Adapter ten jest odpowiedzialny za wyświetlanie informacji o zadaniach w atrakcyjny i przejrzysty sposób.

Dodatkowo, `MainActivity` obsługuje zdarzenia kliknięcia na elementy listy za pomocą metody `setOnItemClickListener`. Gdy użytkownik kliknie na wybrane zadanie, otwierana jest aktywność szczegółów zadania (`TaskDetailActivity`). Do tej aktywności przekazywane są szczegółowe dane zadania – tytuł, opis, termin wykonania, status oraz pozycja w liście – przy użyciu obiektu `Intent`.

Klasa implementuje również metodę `onActivityResult`, która jest wywoływana po powrocie z aktywności szczegółów zadania. Jej zadaniem jest aktualizacja statusu wybranego zadania na podstawie danych zwróconych z aktywności szczegółowej. Jeśli użytkownik zmienił status zadania, lista jest automatycznie odświeżana przy użyciu metody `notifyDataSetChanged` adaptera.

Pomocnicza metoda `updateTaskStatuses` odpowiada za sprawdzanie aktualnych statusów zadań względem ich terminów. Jeśli jakiekolwiek zadanie jest przeterminowane, jego status zostaje odpowiednio zmodyfikowany.

1.1.2. Klasa TaskDetailActivity

```java
package com.example.uaim_lab;


// TaskDetailActivity.java

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import java.text.SimpleDateFormat;
import java.util.Date;

public class TaskDetailActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_task_detail);

        Intent intent = getIntent();
        String title = intent.getStringExtra("title");
        String description = intent.getStringExtra("description");
        long deadlineMillis = intent.getLongExtra("deadline", 0);
```

```
        String status = intent.getStringExtra("status");
        int position = intent.getIntExtra("position", -1);

        TextView titleView = findViewById(R.id.taskDetailTitle);
        TextView descriptionView = findViewById(R.id.taskDetailDescription);
        TextView deadlineView = findViewById(R.id.taskDetailDeadline);
        TextView statusView = findViewById(R.id.taskDetailStatus);
        Button doneButton = findViewById(R.id.doneButton);
        Button notDoneButton = findViewById(R.id.notDoneButton);
        Button backButton = findViewById(R.id.backButton);

        titleView.setText(title);
        descriptionView.setText(description);
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm");
        deadlineView.setText(sdf.format(new Date(deadlineMillis)));
        statusView.setText(status);

        if (status.equals("Przeterminowane")) {
            doneButton.setEnabled(false); // Uniemożliwienie oznaczenia jako
wykonane
        }

        doneButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                sendResult("Wykonane", position);
            }
        });

        notDoneButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                sendResult("Niewykonane", position);
            }
        });

        backButton.setOnClickListener(new View.OnClickListener(){
           @Override
           public void onClick(View v) {
               Intent intent = new Intent(TaskDetailActivity.this,
MainActivity.class);
               intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
               startActivity(intent);
               finish();
           }
```

```
        });
    }

    private void sendResult(String status, int position) {
        Intent resultIntent = new Intent();
        resultIntent.putExtra("status", status);
        resultIntent.putExtra("position", position);
        setResult(RESULT_OK, resultIntent);
        finish();
    }
}
```

Klasa `TaskDetailActivity` reprezentuje ekran szczegółów zadania w aplikacji. Jest to odrębna aktywność, która umożliwia użytkownikowi przeglądanie szczegółowych informacji o wybranym zadaniu oraz modyfikowanie jego statusu.

W metodzie `onCreate`, wywoływanej przy inicjalizacji aktywności, następuje konfiguracja interfejsu użytkownika. Aplikacja odbiera dane przekazane z aktywności głównej za pomocą obiektu `Intent`. Dane te obejmują tytuł, opis, termin wykonania, status oraz pozycję zadania w liście. Te informacje są następnie wyświetlane w odpowiednich elementach interfejsu:

`TextView` dla tytułu, opisu, terminu i statusu – pola są wypełniane wartościami odpowiednio sformatowanymi. W interfejsie znajdują się również trzy przyciski:

Przycisk „Wykonane” – pozwala użytkownikowi oznaczyć zadanie jako wykonane. Przyciski są dynamicznie dostosowywane do stanu zadania: Jeśli zadanie jest „Przeterminowane”, przycisk „Wykonane” zostaje wyłączony (`setEnabled(false)`), ponieważ wykonanie takiego zadania nie jest możliwe. Przycisk „Niewykonane” – umożliwia oznaczenie zadania jako niewykonane. Przycisk „Powrót” – umożliwia powrót do głównego ekranu. Przejście realizowane jest przez utworzenie nowego `Intent` z flagami `FLAG_ACTIVITY_CLEAR_TOP` oraz `FLAG_ACTIVITY_NEW_TASK`, które zapewniają zamknięcie obecnej aktywności i przejście bez duplikowania stosu. Obie akcje dotyczące zmiany statusu zadania (przyciski „Wykonane” i „Niewykonane”) wywołują metodę `sendResult`. Metoda ta:

- Tworzy nowy obiekt `Intent` i zapisuje w nim status zadania oraz jego pozycję.
- Wywołuje metodę `setResult` z kodem `RESULT_OK`, aby poinformować aktywność wywołującą o powodzeniu operacji.
- Zamyka aktywność przy użyciu `finish`.

1.1.3. Klasa TaskAdapter

```
package com.example.uaim_lab;                                    Java

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
```

```java
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import android.widget.ArrayAdapter;
import java.text.SimpleDateFormat;
import java.util.ArrayList;

public class TaskAdapter extends ArrayAdapter<Task> {
    public TaskAdapter(Context context, ArrayList<Task> tasks) {
        super(context, 0, tasks);
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup
parent) {
        Task task = getItem(position);

        if (convertView == null) {
            convertView =
LayoutInflater.from(getContext()).inflate(R.layout.task_item, parent, false);
        }

        TextView titleView = convertView.findViewById(R.id.taskTitle);
        TextView deadlineView = convertView.findViewById(R.id.taskDeadline);
        ImageView statusView = convertView.findViewById(R.id.taskStatus);

        titleView.setText(task.getTitle());
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm");
        deadlineView.setText(sdf.format(task.getDeadline()));

        if ("Wykonane".equals(task.getStatus())) {
            statusView.setImageResource(R.drawable.green_circle);
        } else if ("Przeterminowane".equals(task.getStatus())) {
            statusView.setImageResource(R.drawable.red_cross);
        } else {
            statusView.setImageResource(R.drawable.gray_circle);
        }

        return convertView;
    }
}
```

Klasa `TaskAdapter` jest niestandardowym adapterem służącym do wyświetlania obiektów klasy `Task` w komponencie `ListView`. Dziedziczy po klasie `ArrayAdapter`, umożliwiając przetwarzanie listy danych i ich dostosowanie do wymagań interfejsu użytkownika. Dzięki temu użytkownik widzi zadania w przejrzystym i estetycznym układzie, z odpowiednimi ikonami oraz informacjami tekstowymi.

Konstruktor klasy przyjmuje jako parametry kontekst aplikacji oraz listę obiektów `Task`. Adapter przekazuje te dane do klasy nadrzędnej `ArrayAdapter`, z ustawieniem 0 jako identyfikatora szablonu, ponieważ szablon widoku jest definiowany później w metodzie `getView`.

Metoda `getView`, kluczowa w działaniu adaptera, odpowiada za generowanie widoku dla każdego elementu listy. W jej implementacji:

- Pobranie zadania: Obiekt `Task` dla aktualnej pozycji listy jest pozyskiwany za pomocą `getItem(position)`.
- Recykling widoków: Sprawdzane jest, czy istnieje już zrecyklowany widok (`convertView`). Jeśli nie, nowy widok jest tworzony z użyciem szablonu `R.layout.task_item` poprzez `LayoutInflater`).
- Powiązanie widoku z danymi.

Metoda zwraca kompletny widok reprezentujący jedno zadanie.

1.1.4. Klasa Task

```java
package com.example.uaim_lab;
// Task.java
import java.util.Date;

public class Task {
    private String title;
    private String description;
    private Date deadline;
    private String status;

    public Task(String title, String description, Date deadline, String status) {
        this.title = title;
        this.description = description;
        this.deadline = deadline;
        this.status = status;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public Date getDeadline() {
        return deadline;
    }

    public String getStatus() {
        return status;
    }
```

```
    public void setStatus(String status) {
        this.status = status;
    }
}
```

Klasa `Task` zawiera konstruktor ze zmiennymi:
- `String title`: Nazwa zadania
- `String description`: Szczegółowy opis zadania
- `Date deadline`: Termin zadania
- `String status`: Status zadania

Metody dostępowe:

- `getTitle()`: Zwraca tytuł zadania.
- `getDescription()`: Zwraca opis zadania.
- `getDeadline()`: Zwraca termin realizacji zadania jako obiekt klasy `Date`.
- `getStatus()`: Zwraca aktualny status zadania.

Mutator:
- `setStatus(String status)`: Pozwala zmienić status zadania, np. na „Wykonane” lub „Przeterminowane”.

1.2. Pliki XML

1.2.1. activity_main

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Ten plik XML definiuje layout głównej aktywności aplikacji (`MainActivity`). `ListView` odpowiada za wyświetlanie listy elementów

1.2.2. activity_task_detail

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp">
```

```
    <TextView
        android:id="@+id/taskDetailTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="22sp"
        android:paddingBottom="10dp" />

    <TextView
        android:id="@+id/taskDetailDescription"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:paddingBottom="10dp" />

    <TextView
        android:id="@+id/taskDetailDeadline"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"
        android:paddingBottom="10dp" />

    <TextView
        android:id="@+id/taskDetailStatus"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"
        android:paddingBottom="20dp" />

    <Button
        android:id="@+id/doneButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Oznacz jako wykonane" />

    <Button
        android:id="@+id/notDoneButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Oznacz jako niewykonane" />

    <Button
        android:id="@+id/backButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Powrót" />
</LinearLayout>
```

Ten plik XML definiuje layout używany w `TaskDetailActivity`, który odpowiada za wyświetlanie szczegółów wybranego zadania.

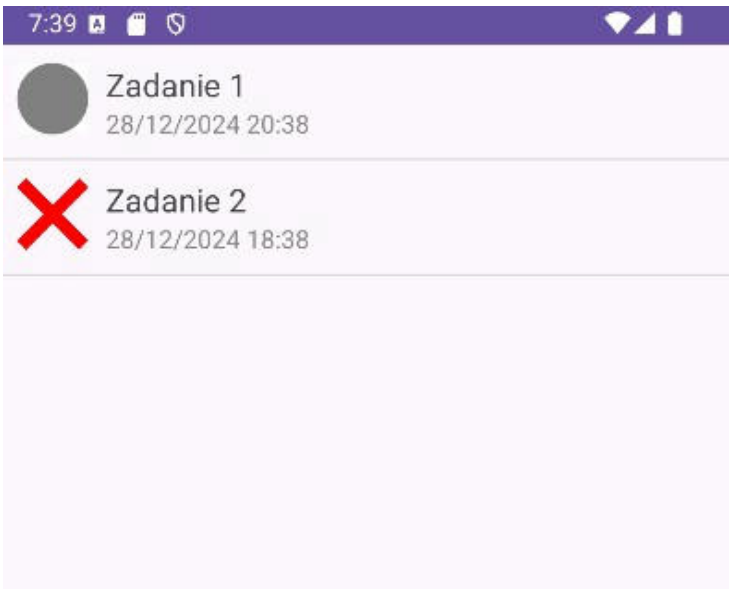
1.2.3. task_item

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="10dp">

    <ImageView
        android:id="@+id/taskStatus"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_marginEnd="10dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/taskTitle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp" />

        <TextView
            android:id="@+id/taskDeadline"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="14sp"
            android:textColor="#888888" />
    </LinearLayout>
</LinearLayout>
```

Ten plik XML definiuje layout dla pojedynczego elementu listy zadań wyświetlanej w aplikacji. Layout ten jest używany przez `TaskAdapter` do renderowania każdego zadania w widoku `ListView`.

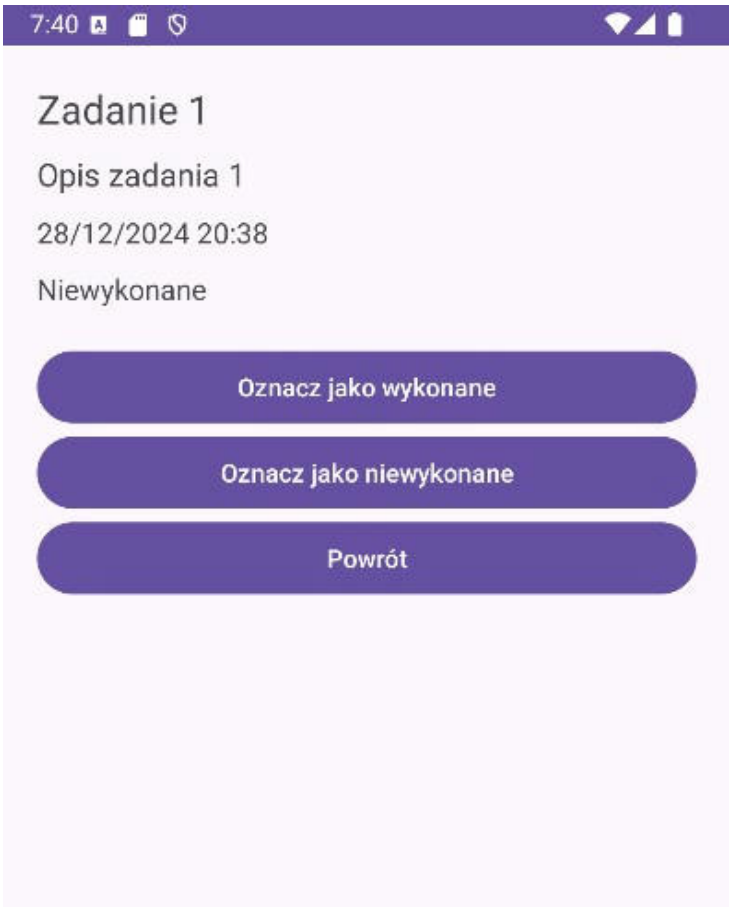
2. Funkcjonalność

2.1. Lista zbiorcza



Rysunek 1: Widok zbiorcze listy zadań

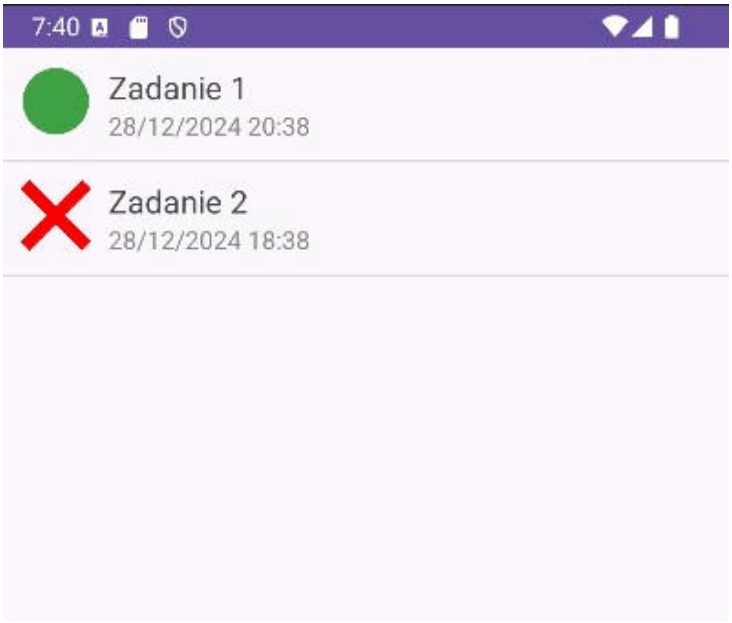
2.2. Widok pojedynczego zadania



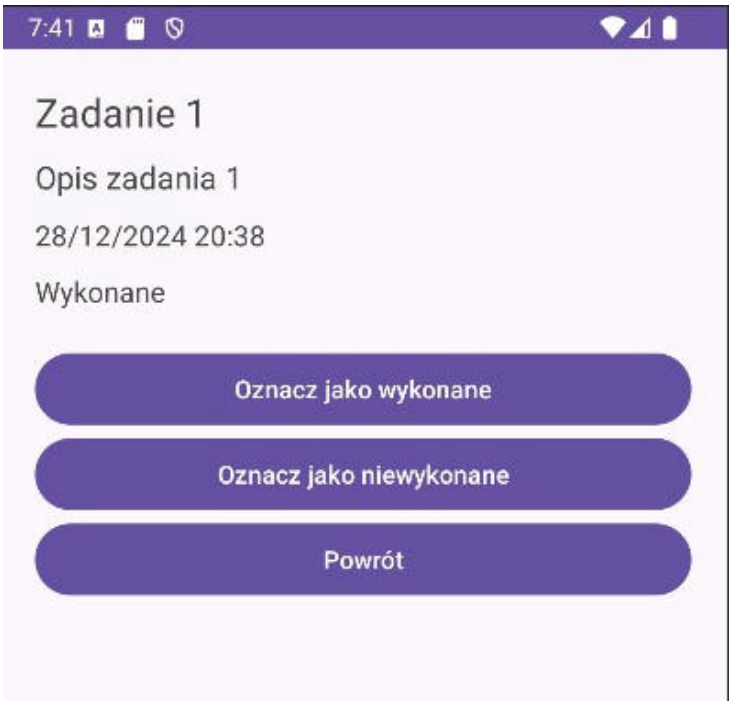
Rysunek 2: Widok pojedynczego zadania ze szczegółami zadania do wykonania

2.3. Zmiana statusu zadania z niewykonane na wykonane

Po kliknięciu na przycisk „Oznacz jako wykonane” zadanie zmienia status na „wykonane”

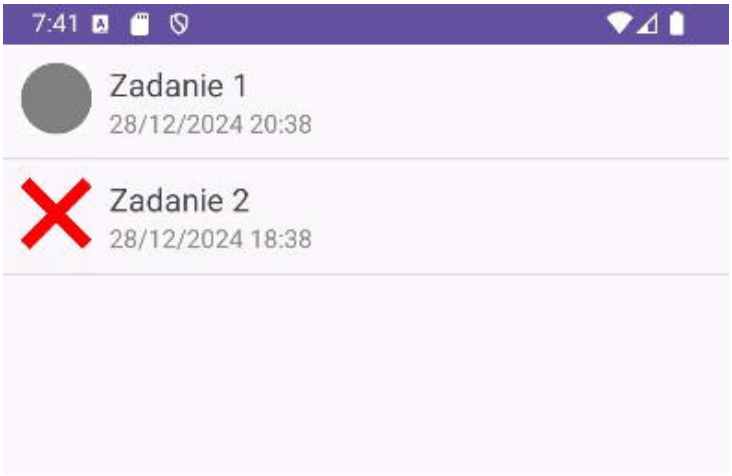


Rysunek 3: Zmiana statusu



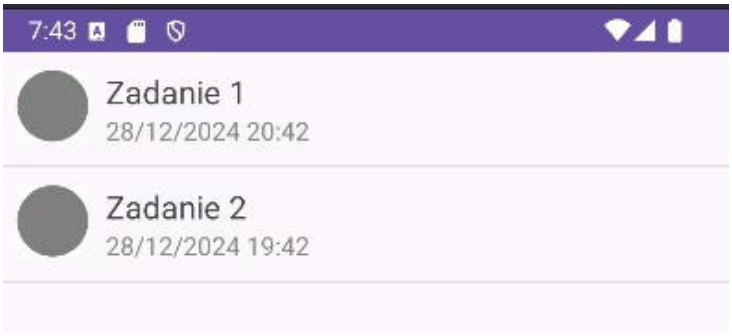
Rysunek 4: Zmiana statusu - w widoku szczegółowym

2.4. Zmiana statusu z wykonane na niewykonane

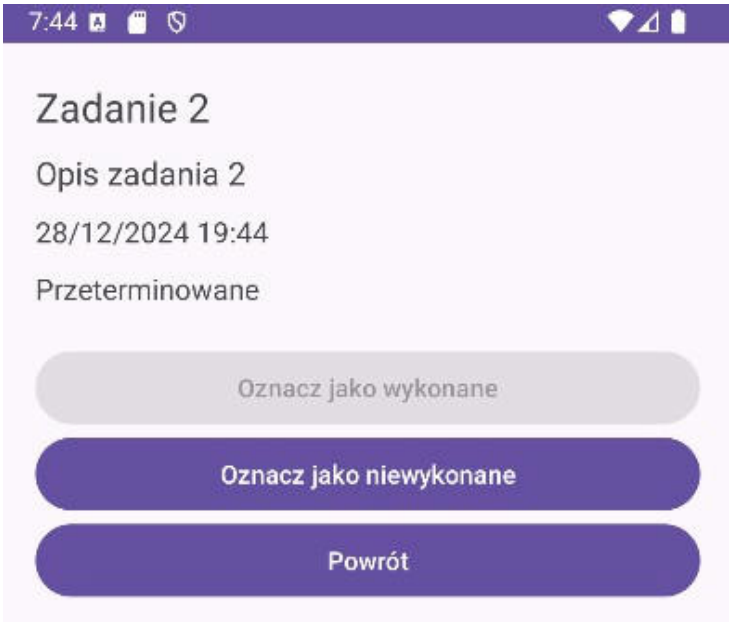


Rysunek 5: Zmiana statusu na niewykonane

2.5. Zmiana statusu z niewykonane na przeterminowane



Rysunek 6: Dwa zadania z statusem niewykonane



Rysunek 7: Zmiana statusu na przeterminowane zadania, którego czas upłynął