Aplikacja „*Recipes*” typu „*Książka Kucharska*”

Jakub Niewiarowski

Nr. Indeksu 149718

Przedmiot: Aplikacje Mobilne

Kierunek / Semestr: Informatyka Niestacjonarnie / 6

Spis treści

1. Opis aplikacji

Aplikacja mobilna pt. „Recipes” została napisana w języku Java. Implementuje ona książkę kucharską tj. posiada bazę przepisów, podzielonych na kategorie. Użytkownik może przeglądać przepisy i wyświetlić ich szczegóły (składniki i sposób przygotowania). Dodatkową funkcjonalnością jest stoper, wyświetlany pod szczegółami przepisu. Pozwala on na mierzenie czasu w sekundach minutach i godzinach. Posiada możliwość zatrzymania oraz resetu odmierzonego czasu.

Aplikacja została stworzona na Android SDK API w wersji 33 i spełnia minimalne wymogi:

- Aplikacja ma składać się z dwóch aktywności: głównej wyświetlającej listę potraw oraz aktywności szczegółów uruchamianej po kliknięciu wybranej potrawy z listy i wyświetlającej listę składników oraz sposób przygotowania potrawy.
- Aplikacja ma korzystać z fragmentów.
- We fragmencie szczegółów należy zagnieździć fragment dynamiczny minutnika, pozwalającego odliczać czas pozostały do końca danego etapu przygotowywania potrawy (np. 14 minut na ugotowanie ryżu).
- Minutnik ma działać z dokładnością do sekundy.
- Minutnik ustawiany ręcznie.
- Dane pobierane z tablicy zdefiniowanej w pliku przechowywanym lokalnie na urządzeniu.
- Aplikacja powinna mieć wersję układu dla smartfonów i osobną dla tabletów.
- Aplikacja powinna działać poprawnie po zmianie orientacji urządzenia.
- Kod aplikacji w Javie.
- Karty kategorii zamiast listy nazw tras mają używać widoku RecyclerView z układem siatki (grid), w którym poszczególne pozycje (potrawy) będą prezentowane w postaci obrazka i nazwy, dla których użyto widoku CardView. Kliknięcie wybranej pozycji (potrawy) powoduje wyświetlenie szczegółów, czyli nazwę potrawy, składników, większego obrazka, sposobu przygotowania, jakiegoś dodatkowego opisu.
- Na ekranie szczegółów ma się pojawić przycisk FAB (floating action button), który będzie odpowiedzialny za wysłanie składników na potrawę poza aplikację np. do notatnika, wiadomości itp. Uwaga! Uruchomienie tej aplikacji nie jest wymagane, aczkolwiek wskazane.
- W aplikacji należy zastosować motywy.
- W aplikacji należy korzystać z fragmentów.
- Aplikacja ma działać poprawnie przy zmianie orientacji urządzenia.
- Każda aktywność ma mieć pasek aplikacji w postaci paska narzędzi.
- Ekran szczegółów ma być przewijany w pionie razem z paskiem aplikacji.
- Na ekranie szczegółów obrazek ma się pojawić na pasku aplikacji, ale ma się razem z nim zwijać.
- Przechodzenie pomiędzy kartami ma się odbywać także za pomocą gestu przeciągnięcia.

2. Opis zasobów

Zasoby aplikacji są zawarte w plikach *strings.xml*, *colors.xml*, *themes.xml* oraz w podfolderach *drawable* i *layout*.

strings.xml
Plik zawiera niezmienne ciągi znaków używane w statycznych elementach aplikacji

```xml
<resources>
    <string name="app_name">Recipes</string>

    <string name="author_activity_title">O autorze</string>

    <string name="detail_button"> Pokaż przepis </string>
    <string name="recipe_name"> Nazwa potrawy </string>
    <string name="ingredients">Składniki:</string>
    <string name="preparation">Sposób przygotowania:</string>

    <string name="start">Start</string>
    <string name="stop">Stop</string>
    <string name="reset">Reset</string>

    <string name="author_content">Recipes 2023\nJakub Niewiarowski\njakub.niewiarowski@student.put.poznan.pl</string>

    <string name="action_share">Share</string>

    <string name="title_top">Witaj,\nprzewiń w prawo, aby zobaczyć przepisy!</string>
    <string name="top_text">Aplikacje Recipes stworzona na zaliczenie przedmiotu Aplikacje Mobilne.\n
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam sed lorem eu sem luctus condimentum. Vestibulum
        Etiam sed eros lectus. Morbi dictum nunc velit, ac elementum ante bibendum nec. Pellentesque habitant morbi tr
    </string>

    <string name="home_tab">Start</string>
    <string name="category1_tab">Obiadowe</string>
    <string name="category2_tab">Desery</string>
</resources>
```

colors.xml
Plik zawiera kolory wykorzystywane do nadania aplikacji innego niż standardowy wyglądu.

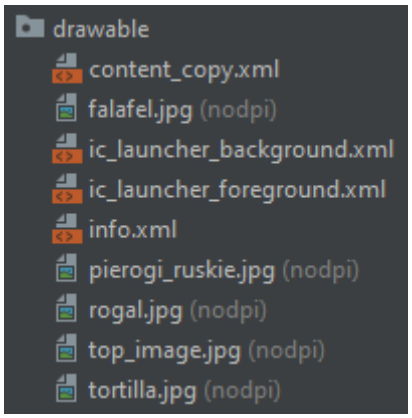
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#EF4136</color>
    <color name="colorPrimaryDark">#F15A29</color>
    <color name="colorAccent">#F7941D</color>

    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

themes.xml
plik służy jedynie do zdefiniowania motywu aplikacji, opartego o jeden ze standardowych.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="Theme.Recipes" parent="Theme.AppCompat.Light.NoActionBar" />
</resources>
```

W Podkatalogach *drawable* znajdują się ikony *(ic_launcher_foreground.xml, content_copy.xml, info.xml*) i zdjęcia wykorzystywane w aplikacji.



W katalogu layout znajdują się pliku układu wszystkich wyświetlanych w aplikacji elementów. Pierwszym wyświetlanym układem *animation_fragment.xml*, który zawiera tylko 2 elementy: tło i ikonę aplikacji.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <FrameLayout
        android:id="@+id/sky"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/black">
        <ImageView
            android:id="@+id/icon"
            android:layout_width="500dp"
            android:layout_height="500dp"
            android:layout_gravity="center"
            android:transformPivotX="250dp"
            android:transformPivotY="250dp"
            android:src="@drawable/ic_launcher_foreground" />
    </FrameLayout>
</LinearLayout>
```

Fragment ten jest wyświetlany w ramach układu *activity_fragment.xml*, który służy do wyświetlania pojedynczych fragmentów.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SingleFragmentActivity">

</androidx.constraintlayout.widget.ConstraintLayout>
```

Plik *main_activity.xml* zawiera układ aktywności głównej. Zawiera on pasek narzędzi (element *include*, wstawiający fragment *toolbar_main.xml*), zwijający się nagłówek (*CollapsingToolbarLayout*), pasek zakładek (*TabLayout*) oraz element *ViewPager2*, w którym wyświetlane są zawartości zakładek w zależności od wybranej zakładki.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" >

        <com.google.android.material.appbar.CollapsingToolbarLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:layout_scrollFlags="scroll|exitUntilCollapsed">

            <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:scaleType="centerCrop"
                android:src="@drawable/top_image"
                app:layout_collapseMode="parallax" />

                <include
                    layout="@layout/toolbar_main"
                    android:id="@+id/toolbar"
                    android:layout_width="match_parent"
                    android:layout_height="?attr/actionBarSize"
                    android:layout_gravity="top"
                    app:layout_collapseMode="pin"/>



        </com.google.android.material.appbar.CollapsingToolbarLayout>

        <com.google.android.material.tabs.TabLayout
            android:id="@+id/tab_layout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/ThemeOverlay.AppCompat.Light"
            android:layout_gravity="bottom"
            app:layout_behavior="com.google.android.material.appbar.AppBarLayout$Scrolli..."/>

    </com.google.android.material.appbar.AppBarLayout>

    <androidx.viewpager2.widget.ViewPager2
        android:id="@+id/pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="10dp"
        app:layout_behavior="com.google.android.material.appbar.AppBarLayout$Scrolli..." />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Układ *toolbar_main.xml* implementuje tylko element *Toolbar*. Pasek narzędzi znajduje się w osobnym pliku zamiast bezpośrednio w układach go wykorzystujących, ponieważ w ten sposób można zmienić wygląd paska edytując tylko jeden plik.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.Toolbar
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="@color/colorPrimary"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

</androidx.appcompat.widget.Toolbar>
```

Zakładka główna, wyświetlana w *main_activity.xml* jako pierwsza, znajduje się w pliku *fragment_top.xml*. Zawiera on a jedynie dwa elementy tekstowe zagnieżdżone w *NestedScrollView*, aby można było je przewijać.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" >
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAppearance="@style/TextAppearance.AppCompat.Large"
                android:text="@string/title_top" />
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/top_text" />
        </LinearLayout>
    </androidx.core.widget.NestedScrollView>
</LinearLayout>
```

Pliki *fragment_tab1.xml* i *fragment_tab2.xml* Zawierają ten sam element *RecyclerView*, ale z innymi identyfikatorami. Element ten pozwala na wyświetlanie listy elementów *CardView*, stanowiącej listę przepisów. Istnieją dwie zakładki, ponieważ aplikacja dzieli przepisy na dwie kategorie.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.recyclerview.widget.RecyclerView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tab1_recycler"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical" >
</androidx.recyclerview.widget.RecyclerView>
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.recyclerview.widget.RecyclerView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tab2_recycler"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical" >
</androidx.recyclerview.widget.RecyclerView>
```

Układ elementów listy przepisów jest zaimplementowany w pliku card_captioned_image.xml. Karta ma lekko zaokrąglone rogi i zawiera zdjęcie przepisu oraz nazwę przepisu pod zdjęciem.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:layout_margin="5dp"
    card_view:cardElevation="2dp"
    card_view:cardCornerRadius="4dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical">
        <ImageView
            android:id="@+id/info_image"
            android:layout_height="0dp"
            android:layout_width="match_parent"
            android:layout_weight="1.0"
            android:scaleType="centerCrop"/>
        <TextView
            android:id="@+id/info_text"
            android:layout_marginLeft="4dp"
            android:layout_marginBottom="4dp"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"/>
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

Układ *activity_detail.xml* zawiera układ aktywności szczegółów przepisu. Implementuje on taki sam pasek narzędzi i nagłówek, jak aktywność główna. Pod nagłówkiem znajduje się element *FragmentContainerView*, który jest odpowiedzialny za wyświetlanie fragmentu ze składnikami i sposobem przygotowania przepisu. Poniżej znajduje się element FAB – jest to przycisk kopiujący składniki do schowka.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent" >

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" >

        <com.google.android.material.appbar.CollapsingToolbarLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:layout_scrollFlags="scroll|exitUntilCollapsed" >

            <ImageView
                android:id="@+id/recipe_image"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:scaleType="centerCrop"
                app:layout_collapseMode="parallax" />

            <include
                layout="@layout/toolbar_main"
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                android:layout_gravity="top"
                app:layout_collapseMode="pin"/>

        </com.google.android.material.appbar.CollapsingToolbarLayout>

    </com.google.android.material.appbar.AppBarLayout>

    <androidx.fragment.app.FragmentContainerView
        android:name="com.mobile.recipes.RecipeDetailFragment"
        android:id="@+id/detail_fragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="10dp"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"/>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end|bottom"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="70dp"
        android:backgroundTint="@color/colorPrimary"
        android:tint="@color/white"
        app:borderWidth="0dp"
        android:src="@drawable/content_copy"
        android:onClick="onClickCopy"/>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Układ fragmentu ze składnikami i sposobem przygotowania przepisu znajduje się w pliku *fragment_recipe_detail.xml*. Zawiera on jedynie elementy tekstowe (ze stałym tekstem z *strings.xml* oraz tekstem zmienianym w zależności od przepisu) oraz *FrameLayout*, w którym wyświetlany jest fragment dynamiczny minutnika. Wszystko znajduje się w elemencie *NestedScrollView*, aby była możliwość przewinięcia przepisu.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.core.widget.NestedScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    app:layout_behavior="com.google.android.material.appbar.AppBarLayout$Scrolli..." >
<LinearLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:orientation="vertical">
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textAlignment="center"
        android:id="@+id/recipeTitle"/>

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:paddingTop="5dp"
        android:id="@+id/recipeCount"/>

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:paddingTop="5dp"
        android:text="Składniki:"/>

    <TextView
    android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/recipeIngredients"/>

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:paddingTop="5dp"
        android:text="Sposób przygotowania:"/>

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/recipePreparation"/>

    <FrameLayout
        android:id="@+id/timer_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    </LinearLayout>
</androidx.core.widget.NestedScrollView>
```

Układ fragmentu minutnika to plik *fragment_timer.xml*. Zawiera on tekst wyświetlający czas oraz przyciski startu, stopu i resetu minutnika. Użyty tutaj został *RelativeLayout* aby poukładać przyciski obok siebie i czas nad przyciskami.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.core.widget.NestedScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    app:layout_behavior="com.google.android.material.appbar.AppBarLayout$Scrolli..." >
<LinearLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:orientation="vertical">
    <TextView
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textAlignment="center"
        android:id="@+id/recipeTitle"/>

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:paddingTop="5dp"
        android:id="@+id/recipeCount"/>

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:paddingTop="5dp"
        android:text="Składniki:"/>

    <TextView
    android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/recipeIngredients"/>

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:paddingTop="5dp"
        android:text="Sposób przygotowania:"/>

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/recipePreparation"/>

    <FrameLayout
        android:id="@+id/timer_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    </LinearLayout>
</androidx.core.widget.NestedScrollView>
```

Ostatnim układem jest *activity_author.xml* zawierający tylko pasek narzędzi i tekst o autorze aplikacji. Otwierany jest po kliknięciu ikony „i” na pasku narzędzi.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    tools:context=".AuthorActivity">

    <include
        layout="@layout/toolbar_main"
        android:id="@+id/toolbar" />

    <TextView
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/author_content"/>

</LinearLayout>
```

AndroidManifest.xml

Określa podstawowe właściwości aplikacji. Ustawia Startową aktywność na *AnimationActivity*. Dla aktywności *DetailActivity* i *AuthorActivity* ustala aktywność nadrzędną na *MainActivity* w celu zaimplementowania przycisku cofnięcia. Dla *MainActivity* ustala motyw.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:theme="@style/Theme.Recipes"
        tools:targetApi="33">
        <activity
            android:name=".AnimationActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".DetailActivity"
            android:exported="false">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".MainActivity" />
        </activity>
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:theme="@style/Theme.Recipes">
        </activity>
        <activity
            android:name=".AuthorActivity"
            android:label="@string/author_activity_title">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".MainActivity" />
        </activity>
    </application>
</manifest>
```

3. Aktywności i fragmenty

Pierwszym uruchamianym fragmentem jest *AnimationActivity*. Uruchamia on na starcie *AnimationFragment* i zaimplementowany *Handler*, który po 5 sekundach uruchamia aktywność *MainActivity*.

```
package com.mobile.recipes;

import ...

1 usage
public class AnimationActivity extends SingleFragmentActivity {
    1 usage
    Handler handler = new Handler();

    1 usage
    @Override
    protected Fragment createFragment() {
        handler.postDelayed(mLaunchTask, delayMillis: 5000);
        return AnimationFragment.newInstance();
    }

    1 usage
    private Runnable mLaunchTask = new Runnable() {
        public void run() {
            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
            startActivity(intent);
        }
    };
}
```

AnimationFragment w funkcji *OnCreateView()* przypisuje do zmiennych odpowiednie elementy układu i rozpoczyna animację tych elementów w funkcji *startAnimation()*.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_animation, container, attachToRoot: false);
    mSceneView = view;
    mIconView = view.findViewById(R.id.icon);
    mSkyView = view.findViewById(R.id.sky);
    Resources resources = getResources();
    mStartColor = resources.getColor(R.color.black);
    mEndColor = resources.getColor(R.color.white);
    startAnimation();
    return view;
}
```

startAnimation() wykonuje animacje przejścia elementu ikony z pozycji spoza ekranu (*iconYStart*) na pozycję pivota elementu tła (*iconYEnd*) oraz animację zmiany koloru tła z czarnego na biały.

```
private void startAnimation() {
    float iconYStart = mSceneView.getTop()-1500;
    float iconYEnd = mSkyView.getPivotY();

    ObjectAnimator heightAnimator = ObjectAnimator
            .ofFloat(mIconView, propertyName: "y", iconYStart, iconYEnd)
            .setDuration(3000);
    heightAnimator.setInterpolator(new DecelerateInterpolator());
    ObjectAnimator sunsetSkyAnimator = ObjectAnimator
            .ofInt(mSkyView, propertyName: "backgroundColor", mStartColor, mEndColor)
            .setDuration(3000);
    sunsetSkyAnimator.setEvaluator(new ArgbEvaluator());
    heightAnimator.start();
    sunsetSkyAnimator.start();
}
```

Następną uruchamianą aktywnością jest *MainActivity*. która jest główną aktywnością aplikacji. Jest ona odpowiedzialna za wyświetlenie strony startowej oraz dwóch stron z przepisami różnych kategorii w formacie zakładek. Funkcja *OnCreateView()* przypisuje kolejno *Adapter* dla kategorii, element *Pager* kategorii, element zakładek kategorii oraz pasek zakładek.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // Stan i układ
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // Adapter kategorii przepisów
    SectionsPagerAdapter pagerAdapter =
            new SectionsPagerAdapter( fragmentActivity: this);
    // Pager kategorii przepisów
    ViewPager2 pager = (ViewPager2) findViewById(R.id.pager);
    pager.setAdapter(pagerAdapter);
    // Menu zakładek
    TabLayout tabLayout = findViewById(R.id.tab_layout);
    new TabLayoutMediator(tabLayout, pager,
            (tab, position) -> tab.setText(pagerAdapter.getPageTitle(position))
    ).attach();
    // Pasek narzędzi
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
}
```

Następnie, ustawiany jest układ paska narzędzi oraz przypisywana jest akcja po kliknięciu ikonki „i”

```java
// Ustawienie układu paska narzędzi
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return super.onCreateOptionsMenu(menu);
}

// Obsługa kliknięcia w link do strony "o autorze"
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.author_action) {
        Intent intent = new Intent( packageContext: this, AuthorActivity.class);
        startActivity(intent);
        return true;
    }
    else {
        return super.onOptionsItemSelected(item);
    }
}
```

Układ paska narzędzi składa się z dwóch ikonek

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto" >
    <item android:id="@+id/action_share"
        android:title="Share"
        android:orderInCategory="2"
        app:showAsAction="ifRoom"
        app:actionProviderClass="androidx.appcompat.widget.ShareActionProvider" />
    <item android:id="@+id/author_action"
        android:title="O autorze"
        android:icon="@drawable/info"
        android:orderInCategory="1"
        app:showAsAction="ifRoom" />
</menu>
```

Na koniec *ActivityMain*, zaimplementowany jest adapter, odpowiedzialny za sekcje (zakładki):

- *TopFragment* – strona startowa
- *Tab1Fragment* – strona z przepisami obiadowymi
- *Tab2Fragment* – przepisy deserowe

```
// Adapter kategorii przepisów
2 usages
private class SectionsPagerAdapter extends FragmentStateAdapter {
    1 usage
    public SectionsPagerAdapter(FragmentActivity fragmentActivity) { super(fragmentActivity); }

    @Override
    public int getItemCount() { return 3; } // 2 kategorie + strona startowa

    // Fragmenty stron
    1 usage
    @Override
    public Fragment createFragment(int position) {
        switch (position) {
            case 0:
                return new TopFragment(); // Strona startowa
            case 1:
                return new Tab1Fragment(); // Przepisy na obiad
            case 2:
                return new Tab2Fragment(); // Przepisy na deser
        }
        return null;
    }

    // Tytuły stron
    1 usage
    public CharSequence getPageTitle(int position) {
        switch (position) {
            case 0:
                return getResources().getText(R.string.home_tab);
            case 1:
                return getResources().getText(R.string.category1_tab);
            case 2:
                return getResources().getText(R.string.category2_tab);
        }
        return null;
    }
}
```

TopFragment posiada w swojej implementacji jedynie wyświetlenie układu, który zawiera jedynie tekst.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fragment_top, container,  attachToRoot: false);
}
```

Graficznie *MainActivity* z fragmentem *TopFragment* przedstawia się następująco:

Na samej górze znajduje się pomarańczowy pasek narzędzi. Obraz z nazwą aplikacji jest elementem zwijającym się w układzie *fragment_top.xml*. Niżej znajduje się pasek zakładek zaimplementowany przez własny *Adapter*. Reszta ekranu to tekst z układu strony startowej *fragment_top.xml*.

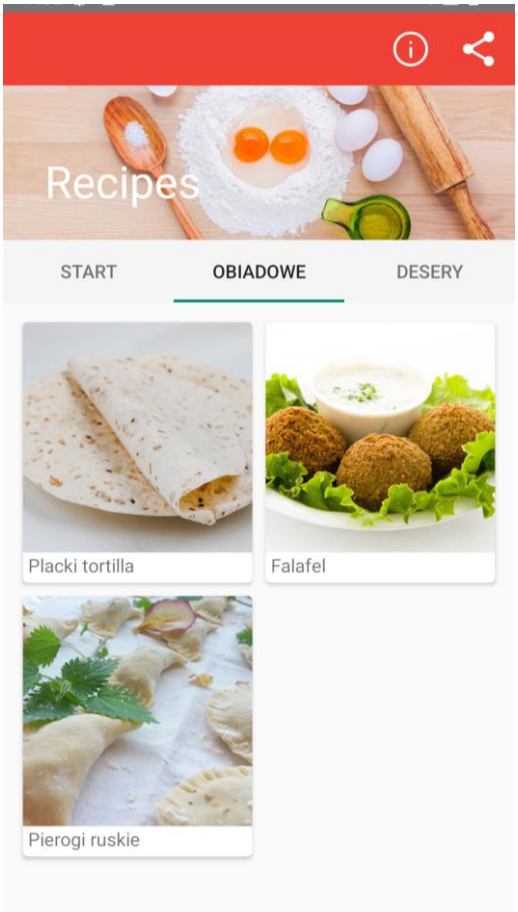
Tab1Fragment i *Tab2Fragment* implementują listę przepisów w postaci siatki kafelków. Jedyną różnica pomiędzy tymi dwoma fragmentami jest lista, z której pobierane są dane od przepisach. Listy są zaimplementowane w ramach klasy *Recipe* będącej szablonem każdego przepisu przechowywanego w aplikacji (więcej o klasie *Recipe* w punkcie 4.).

Fragmenty te implementują tylko funkcję *OnCreateView()*, która kolejno tworzy obiekt *RecyclerView*, pobiera zdjęcia i nazwy przepisów, tworzy adapter kafelków z pobranymi danymi przepisów, tworzy menedżera siatki i ustawia go jako menedżera układu *RecyclerView*, a na koniec, ustawia nasłuchiwanie na kliknięcie przepisu z listy. Po kliknięciu na kafelek z przepisem, tworzony jest obiekt intencji z aktywnością szczegółów przepisu. Do obiektu przypisywany jest identyfikator przepisu z listy i nazwa kategorii (aby aktywność szczegółów wiedziała z jakiej listy pobrać przepis). Intencja ta jest wywoływana i wyświetla się aktywność szczegółów przepisu.

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    // Ustawienie RecyclerView z układu listy przepisów
    RecyclerView recipeRecycler = (RecyclerView)inflater.inflate(R.layout.fragment_tab1, container,  attachToRoot: false);
    // Pobranie nazw i zdjęć przepisów z odpowiedniej listy
    String[] recipeNames = new String[Recipe.recipesDinner.length];
    for (int i = 0; i < recipeNames.length; i++) {
        recipeNames[i] = Recipe.recipesDinner[i].getName();
    }
    int[] recipeImages = new int[Recipe.recipesDinner.length];
    for (int i = 0; i < recipeImages.length; i++) {
        recipeImages[i] = Recipe.recipesDinner[i].getImageId();
    }
    // Ustawienie adaptera kafelków z obrazkami i tekstem
    CaptionedImagesAdapter adapter = new CaptionedImagesAdapter(recipeNames, recipeImages);
    recipeRecycler.setAdapter(adapter);
    // Ustawienie menedżera listy (siatki) kafelków
    GridLayoutManager layoutManager = new GridLayoutManager(getActivity(),  spanCount: 2);
    recipeRecycler.setLayoutManager(layoutManager);
    // Nasłuchiwanie kliknięcia w pozycje na liście
    adapter.setListener(new CaptionedImagesAdapter.Listener() {
        1 usage
        public void onClick(int position) {
            Intent intent = new Intent(getActivity(), DetailActivity.class);
            intent.putExtra(DetailActivity.EXTRA_RECIPE_ID, position);
            intent.putExtra(DetailActivity.EXTRA_RECIPE_CATEGORY,  value: "dinner");
            getActivity().startActivity(intent);
        }
    });

    return recipeRecycler;
}
```

Graficznie *Tab1Fragment* i *Tab2Fragment* przedstawiają się następująco:



Pasek narzędzi nagłówek i zakładki pochodzą z *MainActivity*. Lista przepisów jest implementowana w fragmencie *Tab1Fragment*. Wygląd *Tab2Fragment* różni się jedynie przepisami.

CaptionImageAdapter jest zaimplementowany w osobnej klasie. Zmienne przechowują nazwy i zdjęcia do wyświetlenia oraz *Listener* do nasłuchiwania kliknięcia na daną pozycję listy przepisów. Listy zdjęć i nazw przekazuje się przy tworzeniu obiektu tej klasy, *Listener* ustawia się później w osobnej metodzie, co było widoczne w kodzie fragmentu *Tab1Fragment*.

```
class CaptionedImagesAdapter extends RecyclerView.Adapter<CaptionedImagesAdapter.ViewHolder> {
    4 usages
    private String[] captions;
    2 usages
    private int[] imageIds;
    3 usages
    private Listener listener;

    4 usages  2 implementations
    interface Listener {
        1 usage  2 implementations
        void onClick(int position);
    }

    2 usages
    public void setListener(Listener listener) { this.listener = listener; }

    2 usages
    public CaptionedImagesAdapter(String[] captions, int[] imageIds) {
        this.captions = captions;
        this.imageIds = imageIds;
    }
```

Właściwe przypisywanie zdjęć i nazw do elementów *CardView* (kafelków) odbywa się w metodzie *OnBindViewHolder()*. Na podstawie przekazanej pozycji z listy pobierane są odpowiednie zdjęcie i nazwa.

```
@Override
public void onBindViewHolder(ViewHolder holder, final int position) {
    final CardView cardView = holder.cardView; // Zaciągnięcie CardView z układu
    // Zaciągnięcie do zmiennych elementu obrazu z układu oraz odpowiedniego zdjęcia przepisu
    ImageView imageView = (ImageView)cardView.findViewById(R.id.info_image);
    Drawable drawable = ContextCompat.getDrawable(cardView.getContext(), imageIds[position]);
    // Wstawienie zdjęcia do elementu obrazu CardView
    imageView.setImageDrawable(drawable);
    imageView.setContentDescription(captions[position]);
    // Wstawienie nazyw przepisu do elementu tekstu CardView
    TextView textView = (TextView)cardView.findViewById(R.id.info_text);
    textView.setText(captions[position]);
    // Nastawienie nasłuchiwania kliknięcia w daną pozycję z listy
    cardView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (listener != null) {
                listener.onClick(position);
            }
        }
    });
}
```

ViewHolder jest zaimplementowany w wewnętrznej klasie.

```
public static class ViewHolder extends RecyclerView.ViewHolder {

    2 usages
    private CardView cardView;

    1 usage
    public ViewHolder(CardView v) {
        super(v);
        cardView = v;
    }
}
```

Po kliknięciu w pozycję z listy, wyświetlane są szczegóły przepisu – uruchamiana jest aktywność *DetailActivity*. Zawiera ona zmienne statyczne *EXTRA_RECIPE_ID* i *EXTRA_RECIPE_CATEGORY* przekazywane przez Intent tworzony wcześniej przy kliknięciu w element listy. Wykorzystywane są one do pobrania odpowiedniego przepisu z listy.

W metodzie *OnCreate()* ustawiany jest układ aktywności i układ fragmentu zawierającego tekst przepisu i minutnik. Do zwijanego nagłówka jest wstawiane zdjęcie danego opisu. Ustawiany jest także pasek narzędzi, identyczny jak w *MainActivity*, ale tutaj do przycisku udostępniania wstawiany jest cały tekst przepisu, sklejony w jeden łańcuch znaków.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // Stan i układ
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_detail);
    // Fragment opisu przepisu
    RecipeDetailFragment fragment = (RecipeDetailFragment)
            getSupportFragmentManager().findFragmentById(R.id.detail_fragment);
    // Przekazywanie fragmentowi identufikatora przepisu i kategorii
    int recipeID = (int) getIntent().getExtras().get(EXTRA_RECIPE_ID);
    String recipeCategory = (String) getIntent().getExtras().get(EXTRA_RECIPE_CATEGORY);
    fragment.setRecipeID(recipeID, recipeCategory);
    // Pobieranie zdjęcia i wstawianie go w element zwijanego obrazka w nagłówku układu
    Recipe recipe = recipeCategory.equals("dinner") ? Recipe.recipesDinner[recipeID] :
            recipeCategory.equals("dessert") ? Recipe.recipesDessert[recipeID] :
                    Recipe.recipesDinner[recipeID];
    ImageView imageView = (ImageView)findViewById(R.id.recipe_image);
    imageView.setImageDrawable(ContextCompat.getDrawable( context: this, recipe.getImageId()));
    //Pobierz przepis do dostawcy udostepniania
    shareContent = recipe.toString();
    // Ustawienie paska narzędzi (tak samo jak w MainActivity)
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    ActionBar actionBar = getSupportActionBar();
    actionBar.setDisplayHomeAsUpEnabled(true);
}
```

```java
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    MenuItem menuItem = menu.findItem(R.id.action_share);
    shareActionProvider = (ShareActionProvider) MenuItemCompat.getActionProvider(menuItem);
    setShareActionIntent(shareContent);
    return super.onCreateOptionsMenu(menu);
}

1 usage
private void setShareActionIntent(String text) {
    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.setType("text/plain");
    intent.putExtra(Intent.EXTRA_TEXT, text);
    shareActionProvider.setShareIntent(intent);
}
```

Aktywność ta obsługuje także FAB. Za pomocą obiektu *ClipboardManager* składniki przepisu są kopiowane do schowka urządzenia. *Handler* wyświetla przez 1 sekundę informację o skopiowaniu do schowka za pomocą alertu.

```java
public void onClickCopy(View view) {
    int recipeID = (int) getIntent().getExtras().get(EXTRA_RECIPE_ID);
    String recipeCategory = (String) getIntent().getExtras().get(EXTRA_RECIPE_CATEGORY);

    String recipeName = recipeCategory.equals("dinner") ? Recipe.recipesDinner[recipeID].getName() :
            recipeCategory.equals("dessert") ? Recipe.recipesDessert[recipeID].getName() :
                    Recipe.recipesDinner[recipeID].getName();
    String recipeIngredients = recipeCategory.equals("dinner") ? Recipe.recipesDinner[recipeID].getIngredients() :
            recipeCategory.equals("dessert") ? Recipe.recipesDessert[recipeID].getIngredients() :
                    Recipe.recipesDinner[recipeID].getIngredients();

    ClipboardManager clipboard = (ClipboardManager) getSystemService(Context.CLIPBOARD_SERVICE);
    ClipData clip = ClipData.newPlainText( label: "Składniki na " + recipeName, recipeIngredients);
    clipboard.setPrimaryClip(clip);

    final Handler handler = new Handler();
    final AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setMessage("Skopiowano do schowka");
    final AlertDialog dialog = builder.create();
    dialog.show();
    handler.postDelayed(new Runnable() {
        public void run() { dialog.dismiss(); }
    }, delayMillis: 1000);
}
```

We fragmencie *RecipeDetailFragment*, odpowiedzialnym za obsługę tekstu szczegółów przepisu metoda *OnCreate()* tworzy przy pierwszym uruchomieniu transakcję fragmentu minutnika w celu obsługi zmian w dynamicznym fragmencie minutnika.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (savedInstanceState == null) {
        TimerFragment timer = new TimerFragment();
        FragmentTransaction ft = getChildFragmentManager().beginTransaction();
        ft.add(R.id.timer_container, timer);
        ft.addToBackStack( name: null);
        ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
        ft.commit();
    }
    else {
        recipeID = savedInstanceState.getLong( key: "recipeID");
    }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fragment_recipe_detail, container,  attachToRoot: false);
}
```

Dlatego też, teksty z obiektu danego przepisu są przypisywane do elementów tekstowych układu w metodzie *OnStart()*

```
@Override
public void onStart() {
    super.onStart();
    View view = getView();
    if (view != null) {
        Recipe recipe = recipeCategory.equals("dinner") ? Recipe.recipesDinner[(int) recipeID] :
                recipeCategory.equals("dessert") ? Recipe.recipesDessert[(int) recipeID] :
                        Recipe.recipesDinner[(int) recipeID];

        TextView title = (TextView) view.findViewById(R.id.recipeTitle);
        title.setText(recipe.getName());

        TextView count = (TextView) view.findViewById(R.id.recipeCount);
        count.setText(recipe.getCount());

        TextView ingredients = (TextView) view.findViewById(R.id.recipeIngredients);
        ingredients.setText(recipe.getIngredients());

        TextView preparation = (TextView) view.findViewById(R.id.recipePreparation);
        preparation.setText(recipe.getPreparation());
    }
}
```

Minutnik jest obsługiwany przez *TimerFragment*. Utrzymuje on w zmiennych czas w sekundach, jaki minął, informację, czy minutnik był uruchomiony.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (savedInstanceState != null) {
        seconds = savedInstanceState.getInt( key: "seconds");
        running = savedInstanceState.getBoolean( key: "running");
    }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View layout = inflater.inflate(R.layout.fragment_timer, container,  attachToRoot: false);
    runTimer(layout);
    Button startButton = (Button)layout.findViewById(R.id.start_button);
    startButton.setOnClickListener(this);
    Button stopButton = (Button)layout.findViewById(R.id.stop_button);
    stopButton.setOnClickListener(this);
    Button resetButton = (Button)layout.findViewById(R.id.reset_button);
    resetButton.setOnClickListener(this);
    return layout;
}
```

Na podstawie tych zmiennych, tekst minutnika zmienia swoją zawartość. Jest to kontrolowane przez funkcję *runTimer()*, w której jest implementowany *Handler* próbujący zmienić co sekundę tekst minutnika.

```
private void runTimer(View view) {
    final TextView timeView = (TextView) view.findViewById(R.id.time_view);
    final Handler handler = new Handler();
    handler.post(new Runnable() {
        @Override
        public void run() {
            int hours = seconds / 3600;
            int minutes = (seconds % 3600) / 60;
            int secs = seconds % 60;
            String time = String.format("%d:%02d:%02d", hours, minutes, secs);
            timeView.setText(time);
            if (running) {
                seconds++;
            }
            handler.postDelayed( r: this,  delayMillis: 1000);
        }
    });
}
```

Kliknięcia są implementowane przez poniższe 4 funkcje. Odpowiednie przyciski zatrzymują, wznawiają lub resetują minutnik.

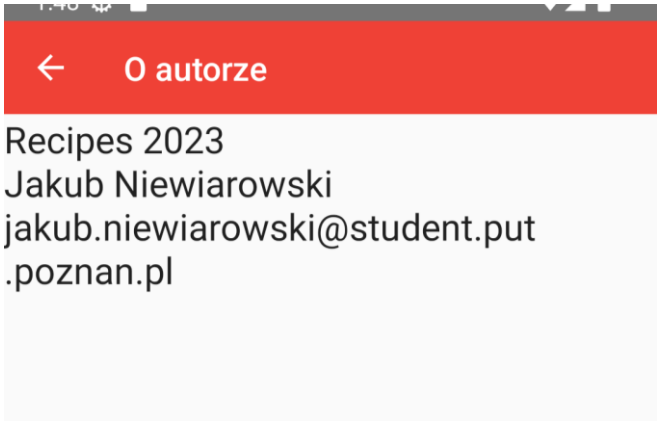
```java
@Override
public void onClick(final View v) {
    int id = v.getId();
    if (id == R.id.start_button) {
        onClickStart();
    } else if (id == R.id.stop_button) {
        onClickStop();
    } else if (id == R.id.reset_button) {
        onClickReset();
    }
}

1 usage
private void onClickStart() { running = true; }
1 usage
private void onClickStop() { running = false; }
1 usage
private void onClickReset() {
    running = false;
    seconds = 0;
}
```

Szczegóły przepisu, które implementują *DetailActivity*, *RecipeDetailFragment* oraz *TimerFragment*, wyglądają następująco:

Za wyświetlenie informacji o autorze odpowiada aktywność *AuthorActivity*. Zawiera ona jedynie metodę *OnCreate()*, która ustawia układ z tekstem oraz pasek narzędzi pozwalający cofnąć się do aktywności nadrzędnej.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_author);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    ActionBar actionBar = getSupportActionBar();
    actionBar.setDisplayHomeAsUpEnabled(true);
}
```



4. Klasa *Recipe*

Klasa *Recipe* implementuje dane przepisów oraz listę przepisów. Przepisy wstawiane są do jednej z list (tablic) w zależności od kategorii (obiadowe/deserowe). Przepisy są dodawane do list statycznie, przy inicjalizacji tablic, bezpośrednio w kodzie.

Sama klasa zawiera tylko osobne zmienne dla nazwy, ilości (np. sztuk pierogów), jednostki miary, składników, sposobu przygotowania i identyfikatora zdjęcia przepisu. Oprócz tego zawiera dwie tablice z obiektami konkretnych przepisów.

```
public class Recipe  {
    3 usages
    private String name;
    2 usages
    private int count;
    2 usages
    private String unit;
    3 usages
    private String ingredients;
    3 usages
    private String preparation;
    2 usages
    private int imageId;
    12 usages
    public static final Recipe[] recipesDinner = {
```

Oprócz tego, zawiera zwykły konstruktor i gettery wszystkich zmiennych.

```
private Recipe(String name, int imageId, int count, String unit, String ingredients, String preparation) {
        this.name = name;
        this.imageId = imageId;
        this.count = count;
        this.unit = unit;
        this.ingredients = ingredients;
        this.preparation = preparation;
}

6 usages
public String getName() { return name; }

3 usages
public int getImageId() { return imageId;}

1 usage
public String getCount() { return count + " " + unit; }

4 usages
public String getIngredients() { return ingredients; }

1 usage
public String getPreparation() { return preparation; }

public String toString() {
        return this.name + "\n" + this.ingredients + "\n" + this.preparation;
}
```