Zastosowanie języka Kotlin w Android Studio

Zbigniew Kluczkowski

Co to jest?

Kotlin to nowoczesny, statycznie typowany język programowania:

- Stworzony przez JetBrains (2011)
- Oficjalny język do tworzenia aplikacji na Androida od 2017 roku
- Kompatybilny z Javą – działa na JVM

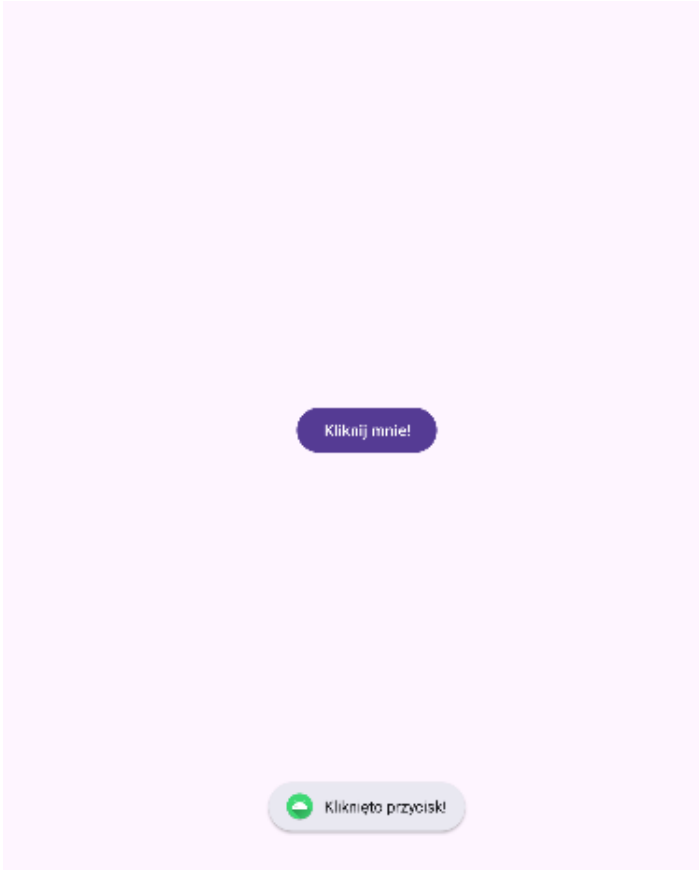
Cechy:

- Zwięzła i czytelna składnia (mniej kodu niż w Javie)
- Bezpieczeństwo typów i null-safety (eliminuje błędy null pointer exception)
- Wsparcie dla funkcji wyższego rzędu i lambd
- Doskonała interoperacyjność z istniejącym kodem Java
- Coroutines – uproszczona obsługa programowania asynchronicznego

Przykładowy kod

```
package com.example.przycisk

import android.os.Bundle
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val myButton: Button = findViewById(R.id.myButton)

        myButton.setOnClickListener {
          Toast.makeText(this, "Kliknięto przycisk!",
Toast.LENGTH_SHORT).show()
        }
    }
}
```

Przycisk i nasłuchiwacz oraz dopisanie akcji w postaci komunikatu o kliknięciu.



Elementy języka

- Klasy i obiekty
- Funkcje rozszerzeń (Extension Functions)
- Lambdy i funkcje wyższego rzędu
- Data classes (klasy danych)
- Null safety i bezpieczne wywołania (?., ?:)
- Coroutines do obsługi asynchroniczności

Narzędzia i biblioteki:

- Kotlin Standard Library
- Android KTX – rozszerzenia Kotlin dla Androida
- Coroutines (kotlinx.coroutines)
- Jetpack Compose – nowoczesny toolkit do tworzenia UI w Kotlinie
- Integracja z popularnymi frameworkami, np. Retrofit, Room, Dagger

Klasy w Kotlinie

```
class Osoba(val imie: String, var wiek: Int) {
    fun przedstawSie() {
        println("Cześć, mam na imię $imie i mam $wiek lat.")
    }
}
```

```
val liczba: Int = 10
val pi: Double = 3.14
val aktywny: Boolean = true
val imie: String = "Kotlin"
```

- val – właściwość tylko do odczytu (niemodyfikowalna)
- var – właściwość modyfikowalna

Obiekty w Kotlinie

```
val osoba = Osoba("Jan", 30)

osoba.przedstawSie()
```

Instancja klasy

```
object Menedzer {
    fun zarzadzaj() {
        println("Zarządzanie w toku")
    }
}
```

Singleton – pojedynczy obiekt bez instancji

Funkcje w Kotlinie

```
fun dodaj(a: Int, b: Int): Int {
    return a + b
}
```

```
fun dodaj(a: Int, b: Int) = a + b
```

```
package com.example.poleprostokata

import android.os.Bundle
import android.widget.*
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    // Funkcja licząca pole prostokąta
    fun obliczPole(szerokosc: Double, wysokosc:
Double): Double {
        return szerokosc * wysokosc
    }

    override fun onCreate(savedInstanceState: Bundle?)
{
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val editTextSzerokosc =
findViewById<EditText>(R.id.editTextSzerokosc)
        val editTextWysokosc =
findViewById<EditText>(R.id.editTextWysokosc)
        val buttonOblicz =
findViewById<Button>(R.id.buttonOblicz)
        val textViewWynik =
findViewById<TextView>(R.id.textViewWynik)

        buttonOblicz.setOnClickListener {
            try {
                val szerokosc =
editTextSzerokosc.text.toString().toDouble()
                val wysokosc =
editTextWysokosc.text.toString().toDouble()
                val wynik = obliczPole(szerokosc, wysokosc)
                textViewWynik.text = "Pole: $wynik"
            } catch (e: NumberFormatException) {
                Toast.makeText(this, "Podaj poprawne liczby!",
Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

Pętle i warunki w Kotlinie

```
for (i in 1..5) {
    println(i)
}
```

```
var i = 1
while (i <= 5) {
    println(i)
    i++
}
```

```
val x = 10

if (x > 5) {
    println("Większe niż 5")
} else {
    println("Mniejsze lub
równe 5")
}
```

```
val wynik = if (x > 5)
"Duże" else "Małe"
println(wynik)
```

WHEN, Lambda, Map i Filter

```
val ocena = 5

val opis = when (ocena) {
    6 -> "Celujący"
    5 -> "Bardzo dobry"
    4 -> "Dobry"
    else -> "Niżej niż dobry"
}
```

```
val liczby = listOf(1, 2, 3, 4)
val kwadraty = liczby.map { it * it }

println(kwadraty)  // [1, 4, 9, 16]
```

```
val liczby = listOf(1, 2, 3, 4, 5, 6)
val parzyste = liczby.filter { it % 2 == 0 }

println(parzyste)  // [2, 4, 6]
```

```
val wynik = listOf(1, 2, 3, 4, 5, 6)
    .filter { it % 2 == 0 }   // [2, 4, 6]
    .map { it * it }          // [4, 16, 36]

println(wynik)
```

Metoda / Operator	Opis	Przykład
print() / println()	Wypisanie tekstu w konsoli	println("Hello")
readLine()	Wczytanie danych od użytkownika (String)	val x = readLine()
toString()	Konwersja na String	123.toString()
toInt()	Konwersja na Int	"42".toInt()
toDouble()	Konwersja na Double	"3.14".toDouble()

Metody w języku Kotlin

Ogólne operacje i konwersje

Metody w języku Kotlin

Operacje na liczbach

Metoda	Opis	Przykład
Math.pow(a,b)	Potęgowanie	Math.pow(2.0, 3.0) → 8.0
Math.sqrt(x)	Pierwiastek kwadratowy	Math.sqrt(16.0) → 4.0
kotlin.random.Random.nextInt(a,b)	Losowanie liczby	Random.nextInt(1,101)

Metody w języku Kotlin

Operacje na STRING

Metoda	Opis	Przykład
length	Długość tekstu	"Hello".length → 5
uppercase()	Na wielkie litery	"abc".uppercase() → "ABC"
lowercase()	Na małe litery	"ABC".lowercase() → "abc"
substring(x,y)	Fragment tekstu	"Android".substring(0,3) → "And"
contains("tekst")	Czy zawiera fragment	"Hello".contains("He") → true
replace("x","y")	Zamiana fragmentu	"kot".replace("k","p") → "pot"

Metody w języku Kotlin

Listy i kolekcje

Metoda	Opis	Przykład
listOf()	Lista niemodyfikowalna	val l = listOf(1,2,3)
mutableListOf()	Lista modyfikowalna	val l = mutableListOf(1,2)
add()	Dodawanie elementu	lista.add(4)
remove()	Usuwanie elementu	lista.remove(2)
size	Rozmiar listy	lista.size
sorted()	Sortowanie	listOf(3,1,2).sorted() → [1,2,3]
reversed()	Odwracanie kolejności	listOf(1,2,3).reversed() → [3,2,1]
forEach { }	Iteracja po elementach	lista.forEach { println(it) }

Metody w języku Kotlin

Funkcje i klasy

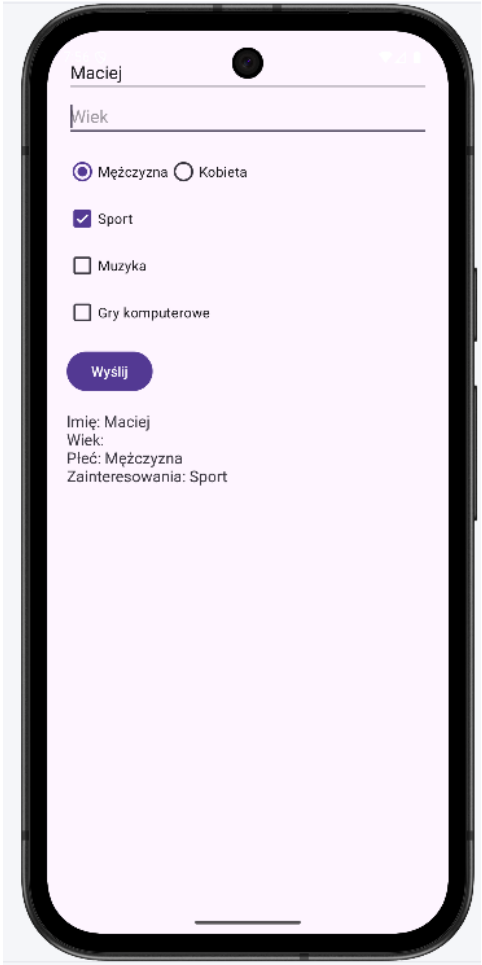
Słowo kluczowe / Metoda	Opis	Przykład
fun	Definicja funkcji	fun suma(a:Int,b:Int)=a+b
class	Definicja klasy	class Osoba(val imie:String)
apply{}	Konfiguruje obiekt	val sb=StringBuilder().apply{append("Hi")}
let{}	Używa obiektu, jeśli nie jest null	x?.let{println(it)}

Metody w języku Kotlin

NULL

Operator	Opis	Przykład
?.	Bezpieczne wywołanie	tekst?.length
?:	Operator Elvis (wartość domyślna)	val x = y ?: 0

Przykładowa aplikacja




```
package com.example.daneosoby

import android.os.Bundle
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import com.example.daneosoby.R.*

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(layout.activity_main)

        val editTextImie = findViewById<EditText>(id.editTextImie)
        val editTextWiek = findViewById<EditText>(id.editTextWiek)
        val radioGroupPlec =
findViewById<RadioGroup>(id.radioGroupPlec)
        val checkSport = findViewById<CheckBox>(id.checkSport)
        val checkMuzyka =
findViewById<CheckBox>(id.checkMuzyka)
        val checkGry = findViewById<CheckBox>(id.checkGry)
        val buttonWyslij = findViewById<Button>(id.buttonWyslij)
        val textViewWynik =
findViewById<TextView>(id.textViewWynik)

        buttonWyslij.setOnClickListener {
          val imie = editTextImie.text.toString()
           val wiek = editTextWiek.text.toString()
           val plec = when (radioGroupPlec.checkedRadioButtonId) {
              id.radioMezczyzna -> "Mężczyzna"
              id.radioKobieta -> "Kobieta"
              else -> "Nie wybrano"
           }

           val zainteresowania = mutableListOf<String>()
           if (checkSport.isChecked) zainteresowania.add("Sport")
           if (checkMuzyka.isChecked)
zainteresowania.add("Muzyka")
           if (checkGry.isChecked) zainteresowania.add("Gry")

           val podsumowanie = """
              Imię: $imie
              Wiek: $wiek
              Płeć: $plec
              Zainteresowania: ${zainteresowania.joinToString(", ")}
           """.trimIndent()

           textViewWynik.text = podsumowanie
        }
    }
}
```

Przykładowa aplikacja

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

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

        <EditText
            android:id="@+id/editTextImie"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Imię" />

        <EditText
            android:id="@+id/editTextWiek"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Wiek"
            android:inputType="number" />

        <RadioGroup
            android:id="@+id/radioGroupPlec"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginTop="10dp">

            <RadioButton
                android:id="@+id/radioMezczyzna"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Mężczyzna" />

            <RadioButton
                android:id="@+id/radioKobieta"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Kobieta" />
        </RadioGroup>

        <CheckBox
            android:id="@+id/checkSport"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Sport" />

        <CheckBox
            android:id="@+id/checkMuzyka"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Muzyka" />

        <CheckBox
            android:id="@+id/checkGry"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Gry komputerowe" />

        <Button
            android:id="@+id/buttonWyslij"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Wyślij"
            android:layout_marginTop="12dp" />

        <TextView
            android:id="@+id/textViewWynik"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text=""
            android:textSize="16sp"
            android:layout_marginTop="16dp" />

    </LinearLayout>
</ScrollView>
```

Czy jest prostszy od Javy?

Zastosowanie Jetpack Compose

Nowoczesny, deklaratywny framework UI dla Androida. Stworzony przez Google jako alternatywa dla tradycyjnego XML-owego layoutu.

Co daje?

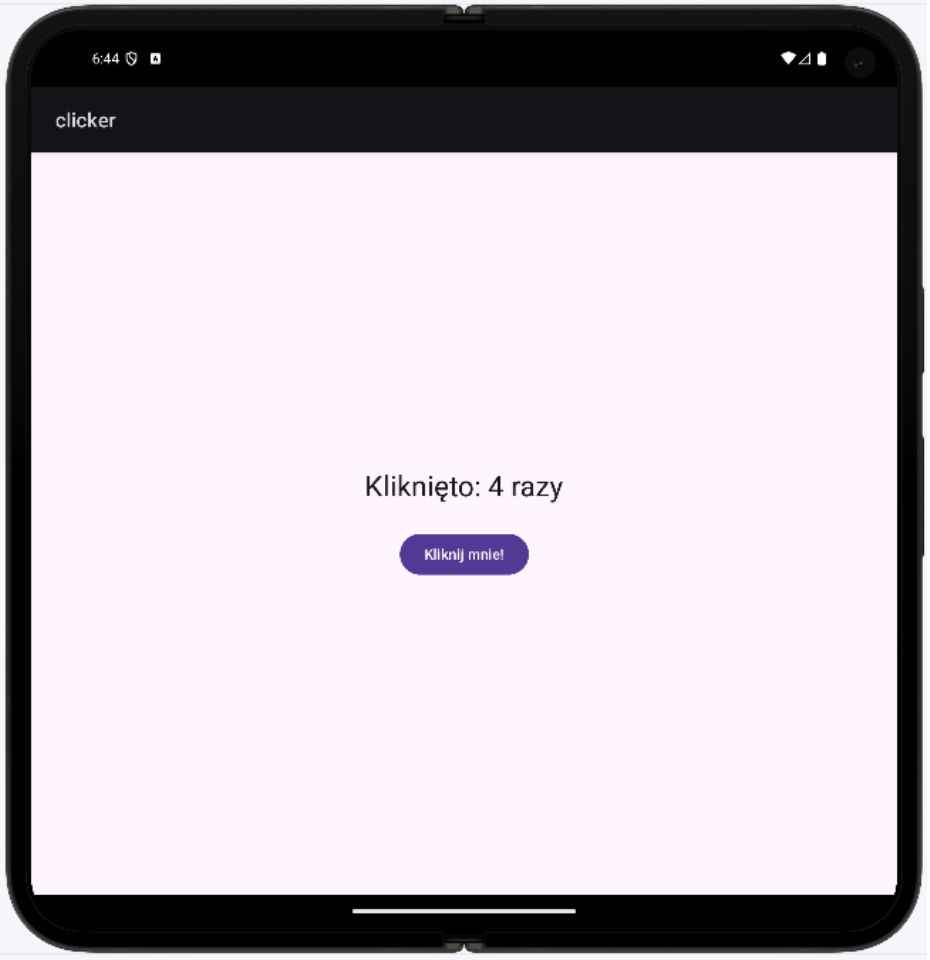
- Prostsze i szybsze tworzenie interfejsów
- Mniej kodu, łatwiejsze do utrzymania
- Deklaratywny sposób definiowania UI
- Natychmiastowa podgląd w Android Studio (Live Preview)
- Lepsza integracja z Kotlinem i nowoczesnymi bibliotekami.

Zastosowanie Jetpack Compose

Uwaga! Jetpack Compose działa tylko w Kotlinie. Nie musisz nic pisać w XAML. Wszystko, czego potrzebujesz to pliki językowe a nie opisowe. Przykład?

```
package com.example.clicker

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MyApp()
        }
    }
}


@Composable
fun MyApp() {
    // Ustawiamy motyw Material Design 3
    MaterialTheme {
        Surface(
            modifier = Modifier.fillMaxSize(),
            color =
MaterialTheme.colorScheme.background
        ) {
            CounterScreen()
        }
    }
}

@Composable
fun CounterScreen() {
    // Stan licznika przechowywany i
zapamiętywany między recomposition
    var count by remember { mutableStateOf(0) }
    // Layout kolumnowy - ułożenie pionowe
elementów
    Column(
        modifier =
Modifier.fillMaxSize().padding(16.dp),
        verticalArrangement =
Arrangement.Center,
        horizontalAlignment =
Alignment.CenterHorizontally
    ) {
        Text(text = "Kliknięto: $count razy", style =
MaterialTheme.typography.headlineMedium)
        Spacer(modifier = Modifier.height(24.dp))
        Button(onClick = { count++ }) {
            Text(text = "Kliknij mnie!")
        }
    }
}
```

Zastosowanie Jetpack Compose




```
plugins {
    alias(libs.plugins.com.android.application)
    alias(libs.plugins.org.jetbrains.kotlin.android)
    alias(libs.plugins.compose.compiler)
}

android {
    namespace = "com.example.clicker"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.clicker"
        minSdk = 26
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    buildFeatures {
        compose = true
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_17
        targetCompatibility = JavaVersion.VERSION_17
    }

    kotlinOptions {
        jvmTarget = "17"
    }
}

dependencies {
    implementation(libs.compose.ui)
    implementation(libs.compose.ui.tooling.preview)
    implementation(libs.compose.material3)
    implementation(libs.activity.compose)

    debugImplementation(libs.debug.compose.ui.tooling)

    debugImplementation(libs.debug.compose.ui.test.manifest)
    implementation(platform(libs.compose.bom))
    implementation(libs.compose.material3)

    implementation("com.google.android.material:material:1.10.0")
}
```

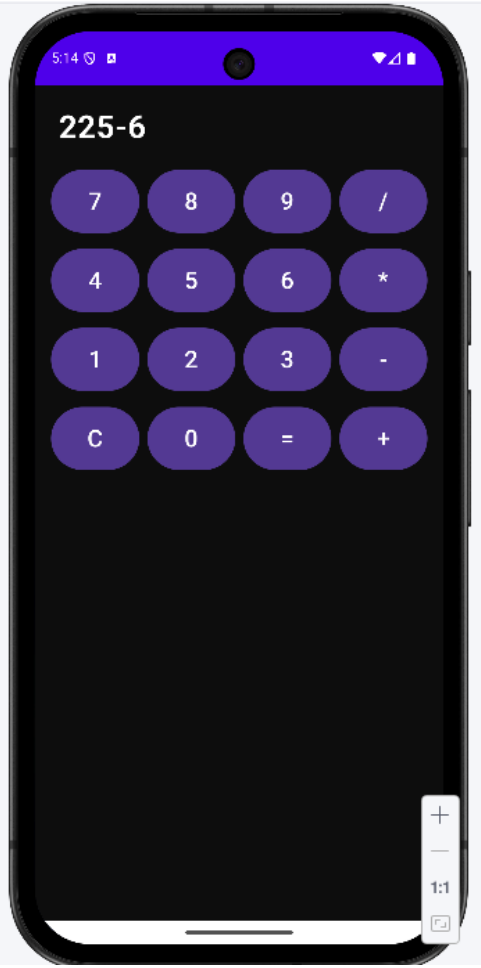
Przykłady zastosowania Jetpack Compose

```
@Composable
fun GreetingScreen() {
    var name by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(8.dp)
    ) {
        Text(text = "Witaj w aplikacji!")
        TextField(
            value = name,
            onValueChange = { name = it },
            label = { Text("Twoje imię") }
        )
        Button(onClick = { /* obsługa kliknięcia */ }) {
            Text("Powiedz Cześć")
        }
        if (name.isNotEmpty()) {
            Text(text = "Cześć, $name!")
        }
    }
}
```

```
@Composable
fun FruitList(fruits: List<String>) {
    LazyColumn {
        items(fruits) { fruit ->
            Card(
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(8.dp),
                elevation = 4.dp
            ) {
                Text(
                    text = fruit,
                    modifier = Modifier.padding(16.dp)
                )
            }
        }
    }
}
```

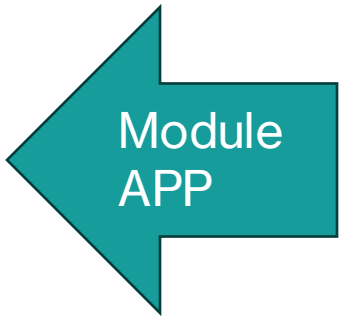
```
@Composable
fun NavExample() {
    val navController = rememberNavController()

    NavHost(navController, startDestination = "home") {
        composable("home") {
            Button(onClick = { navController.navigate("details")
}) {
                Text("Idź do szczegółów")
            }
        }
        composable("details") {
            Text("To jest ekran szczegółów")
        }
    }
}
```

Kalkulator z Jetpack Compose




```
plugins {
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
    id("org.jetbrains.kotlin.plugin.compose") version "2.0.0"
}

android {
    namespace = "com.example.compocalc"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.compocalc"
        minSdk = 26
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
    }

    buildFeatures {
        compose = true
    }

    composeOptions {
        kotlinCompilerExtensionVersion = "1.5.3" // dopasuj do wersji Compose
    }

    kotlinOptions {
        jvmTarget = "1.8"
    }
}
```

```
dependencies {
    val composeBom = platform("androidx.compose:compose-bom:2024.05.00")
    implementation(composeBom)

    implementation("androidx.compose.ui:ui")
    implementation("androidx.compose.ui:ui-tooling-preview")
    implementation("androidx.compose.material3:material3")
    implementation("androidx.activity:activity-compose:1.8.2")
    implementation("com.google.android.material:material:1.10.0")
    debugImplementation("androidx.compose.ui:ui-tooling")
    debugImplementation("androidx.compose.ui:ui-test-manifest")
}
```



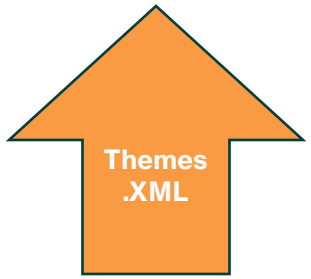
```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.compocalc">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Compocalc">
        <activity android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Kalkulator z Jetpack Compose

```
package com.example.compocalc

import android.os.Bundle
import androidx.activity.ComponentActivity
import
androidx.activity.compose.setContent
import
androidx.compose.foundation.background
import
androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import
androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun
onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MaterialTheme(
                colorScheme = lightColorScheme()
            ) {
                CalculatorApp()
            }
        }
    }
}

@Composable
fun CalculatorApp() {
    var input by remember {
mutableStateOf("") }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(Color(0xFF101010))
            .padding(16.dp),
        verticalArrangement =
Arrangement.spacedBy(16.dp)
    ) {
        // Wyświetlanie wprowadzonego
wyrażenia
        Text(
            text = input,
            fontSize = 32.sp,
            fontWeight = FontWeight.Bold,
            color = Color.White,
            modifier = Modifier
                .fillMaxWidth()
                .padding(8.dp)
        )

        val buttons = listOf(
            listOf("7", "8", "9", "/"),
            listOf("4", "5", "6", "*"),
            listOf("1", "2", "3", "-"),
            listOf("C", "0", "=", "+")
        )

        buttons.forEach { row ->
            Row(
                modifier = Modifier.fillMaxWidth(),
                horizontalArrangement =
Arrangement.spacedBy(8.dp)
            ) {
                row.forEach { label ->
                    Button(
                        onClick = {
                            when (label) {
                                "=" -> input = eval(input)
                                "C" -> input = ""
                                else -> input += label
                            }
                        },
                        modifier = Modifier
                            .weight(1f)
                            .height(64.dp)
                    ) {
                        Text(text = label, fontSize =
24.sp)
                    }
                }
            }
        }
    }
}

// Proste obliczenia wyrażeń
arytmetycznych (+, -, *, /)
fun eval(expr: String): String {
    return try {
        if (expr.isEmpty()) return ""

        // Podział na liczby
        val tokens = Regex("([+\\-
*/])").split(expr).map { it.trim() }.filter {
it.isNotEmpty() }
        // Pobranie operatorów jako String, nie
Char
        val operators = Regex("[0-
9.]+").replace(expr, "").map { it.toString() }

        if (tokens.isEmpty()) return ""

        var result = tokens[0].toDouble()
        for (i in operators.indices) {
            val op = operators[i]
            val num = tokens[i + 1].toDouble()
            result = when (op) {
                "+" -> result + num
                "-" -> result - num
                "*" -> result * num
                "/" -> result / num
                else -> result
            }
        }
        if (result % 1.0 == 0.0)
result.toInt().toString() else result.toString()
    } catch (e: Exception) {
        "Error"
    }
}
```

Kalkulator z Jetpack Compose

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools">
    <style name="Theme.Compocalc"
parent="Theme.Material3.DayNight">
        <!-- Primary brand color -->
        <item name="colorPrimary">#6200EE</item>
        <item name="colorPrimaryContainer">#BB86FC</item>

        <!-- Secondary brand color -->
        <item name="colorSecondary">#03DAC5</item>
        <item name="colorSecondaryContainer">#018786</item>

        <!-- Background & surface -->
        <item name="android:windowBackground">#FFFFFF</item>
        <item name="android:statusBarColor">#6200EE</item>
        <item name="android:navigationBarColor">#FFFFFF</item>

        <!-- Optional: Remove default ActionBar -->
        <item name="windowNoTitle">true</item>
        <item name="windowActionBar">false</item>
    </style>
</resources>
```



Gdzie jest ActivityMain.xml?

Generuje się sam w momencie dodawania kodu w Kotlin. Oczywiście można dodawać władne formatowania

Uwaga!
Często jest tak, że nie każdy szablon jest prawidłowo wyświetlany co generuje wiele błędów.

TOML – ważny element projektu

```
[versions]
kotlin = "2.0.0"
agp = "8.12.0"
compose-ui = "1.6.8"
compose-material3 = "1.2.1"
activity-compose = "1.9.0"
compose-bom = "2025.01.00"

[plugins]
org-jetbrains-kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
compose-compiler = { id = "org.jetbrains.kotlin.plugin.compose", version.ref = "kotlin" }
com-android-application = { id = "com.android.application", version.ref = "agp" }

[libraries]
compose-ui = { module = "androidx.compose.ui:ui", version.ref = "compose-ui" }
compose-ui-tooling-preview = { module = "androidx.compose.ui:ui-tooling-preview",
version.ref = "compose-ui" }
compose-material3 = { module = "androidx.compose.material3:material3", version.ref =
"compose-material3" }
activity-compose = { module = "androidx.activity:activity-compose", version.ref = "activity-
compose" }
debug-compose-ui-tooling = { module = "androidx.compose.ui:ui-tooling", version.ref =
"compose-ui" }
debug-compose-ui-test-manifest = { module = "androidx.compose.ui:ui-test-manifest",
version.ref = "compose-ui" }
compose-bom = { group = "androidx.compose", name = "compose-bom", version.ref =
"compose-bom" }
```

Co to za cudo?

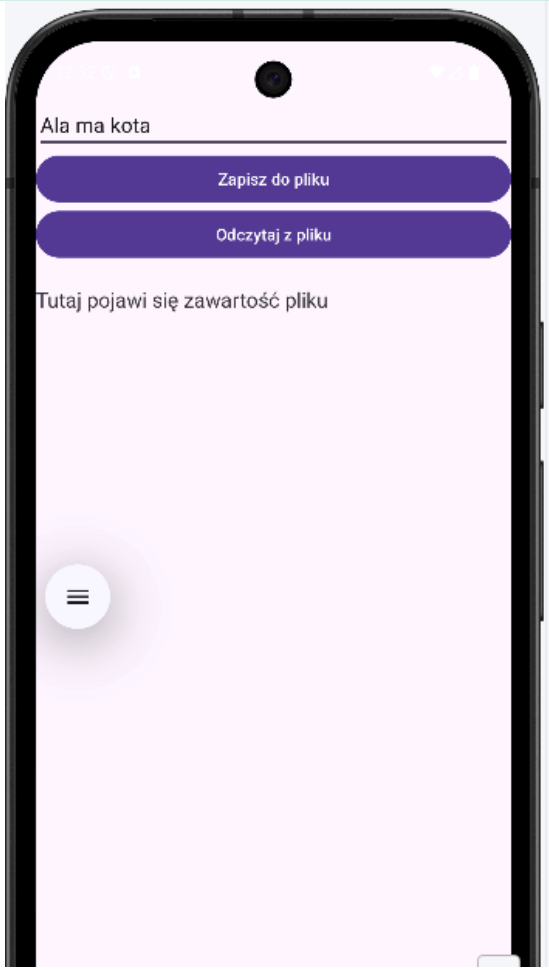
- Bardziej czytelny niż JSON (bez dużej liczby nawiasów klamrowych).
- Prostszy i mniej złożony niż YAML (brak wcięć wrażliwych na błąd).
- Doskonały do **konfiguracji zależności**, np. w gradle/libs.versions.toml do deklarowania wersji bibliotek w Androidzie.

Build.Gradle(APP)

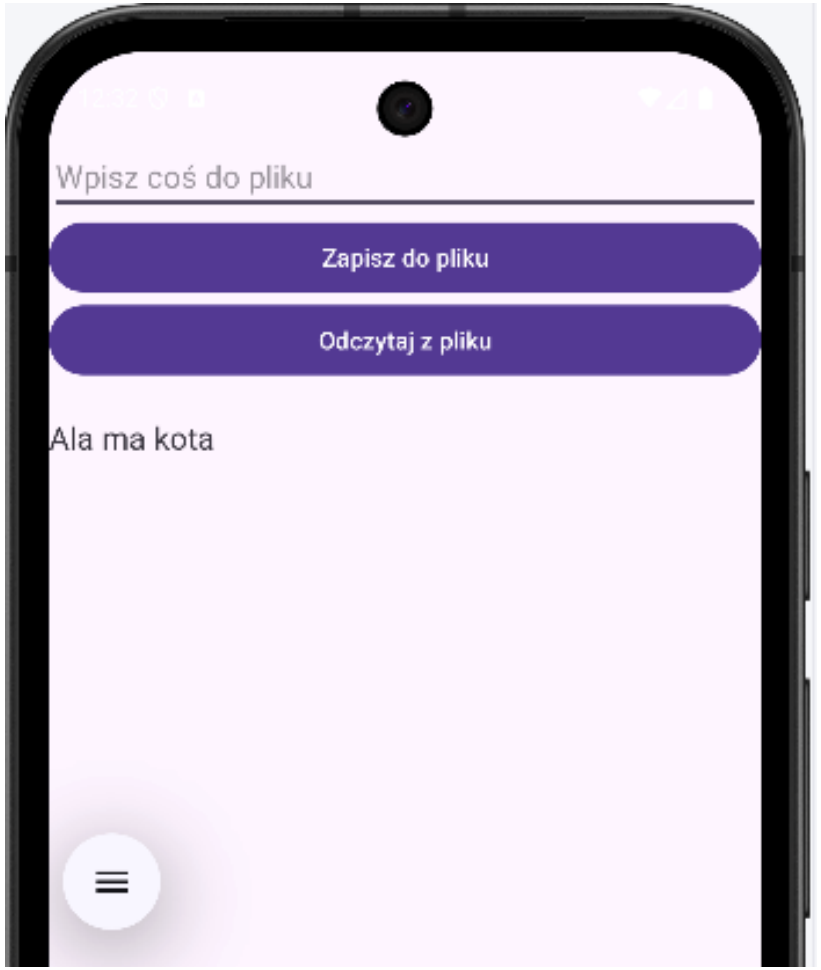
```
plugins {
    alias(libs.plugins.com.android.application)
    alias(libs.plugins.org.jetbrains.kotlin.android)
    alias(libs.plugins.compose.compiler)
}

android {
    namespace = "com.example.clicker"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.clicker"
        minSdk = 26
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    buildFeatures {
        compose = true
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_17
        targetCompatibility = JavaVersion.VERSION_17
    }

    kotlinOptions {
        jvmTarget = "17"
    }
}

dependencies {
    implementation(libs.compose.ui)
    implementation(libs.compose.ui.tooling.preview)
    implementation(libs.compose.material3)
    implementation(libs.activity.compose)
    debugImplementation(libs.debug.compose.ui.tooling)
    debugImplementation(libs.debug.compose.ui.test.manifest)
    implementation(platform(libs.compose.bom))
    implementation(libs.compose.material3)
    implementation("com.google.android.material:material:1.10.0")
}
```

Zwróć uwagę na zależności.

Współpraca języka Kotlin z plikiem i bazą danych



Użytkownik wprowadza dane. Po naciśnięciu przycisku zostaną zapisane do pliku. Po naciśnięciu przycisku dane pojawią się pod przyciskiem.

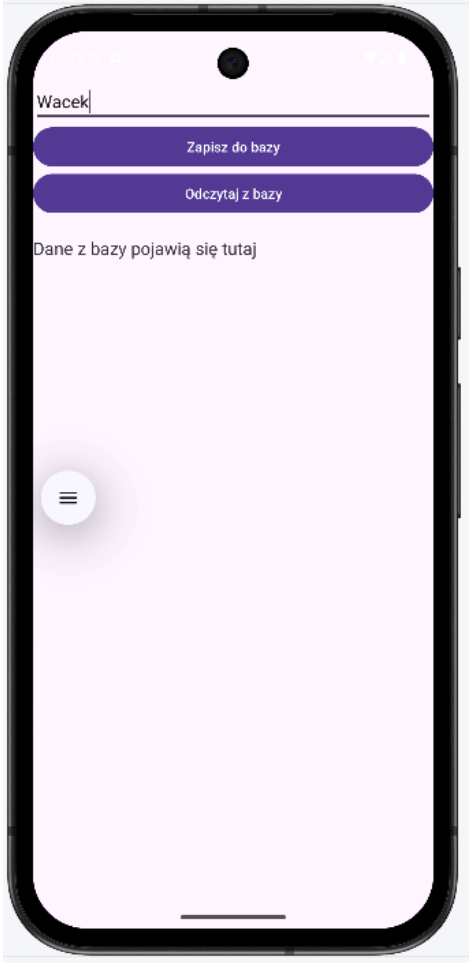


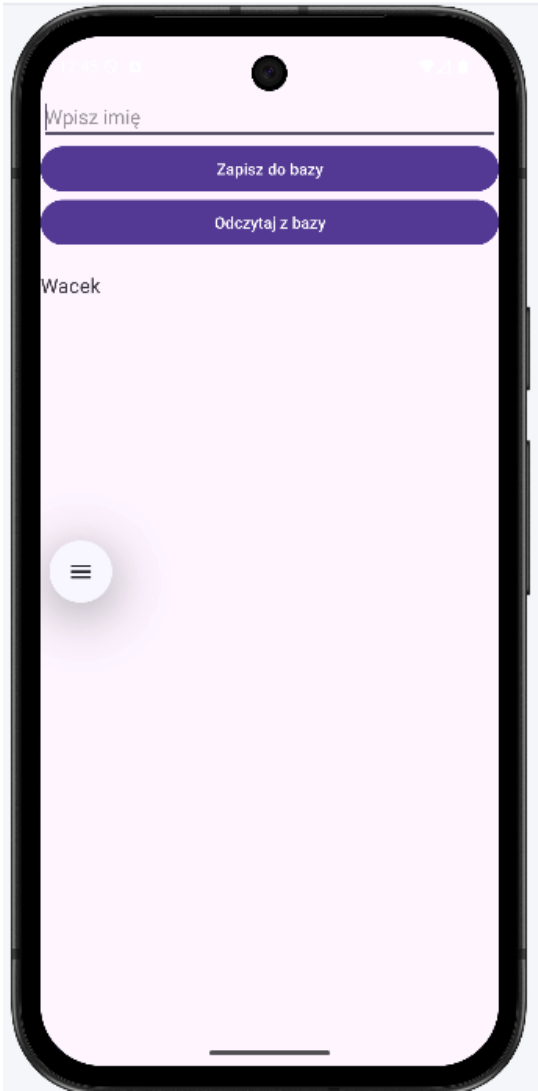
Współpraca języka Kotlin z plikiem i bazą danych

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:paddingTop="50dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/editText"
        android:hint="Wpisz coś do pliku"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/buttonSave"
        android:text="Zapisz do pliku"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/buttonRead"
        android:text="Odczytaj z pliku"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/textView"
        android:text="Tutaj pojawi się zawartość pliku"
        android:textSize="18sp"
        android:layout_marginTop="20dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

```
package com.example.fileexample

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.*
import java.io.*

class MainActivity : AppCompatActivity() {

    private val fileName = "moje_dane.txt"

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val editText =
findViewById<EditText>(R.id.editText)
        val buttonSave =
findViewById<Button>(R.id.buttonSave)
        val buttonRead =
findViewById<Button>(R.id.buttonRead)
        val textView =
findViewById<TextView>(R.id.textView)

        // 👉 Zapis do pliku
        buttonSave.setOnClickListener {
            val text = editText.text.toString()
            if (text.isNotEmpty()) {
                try {
                    openFileOutput(fileName,
MODE_PRIVATE).use {
                        it.write(text.toByteArray())
                    }
                Toast.makeText(this, "Zapisano do pliku!",
Toast.LENGTH_SHORT).show()
                    editText.text.clear()
                } catch (e: Exception) {
                    Toast.makeText(this, "Błąd zapisu:
${e.message}", Toast.LENGTH_SHORT).show()
                }
            }
        }

        // 👉 Odczyt z pliku
        buttonRead.setOnClickListener {
            try {
                val fileInput = openFileInput(fileName)
                val reader =
BufferedReader(InputStreamReader(fileInput))
                val content = reader.readText()
                reader.close()
                textView.text = content
            } catch (e: FileNotFoundException) {
                Toast.makeText(this, "Plik nie istnieje!",
Toast.LENGTH_SHORT).show()
            } catch (e: Exception) {
                Toast.makeText(this, "Błąd odczytu:
${e.message}", Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

Współpraca języka Kotlin z plikiem i bazą danych





Podobnie jak w poprzednim przykładzie, ale tym razem korzystamy z klasy pomocniczej, dzięki której łączymy się z bazą danych oraz zapisujemy i pobieramy z niej dane.

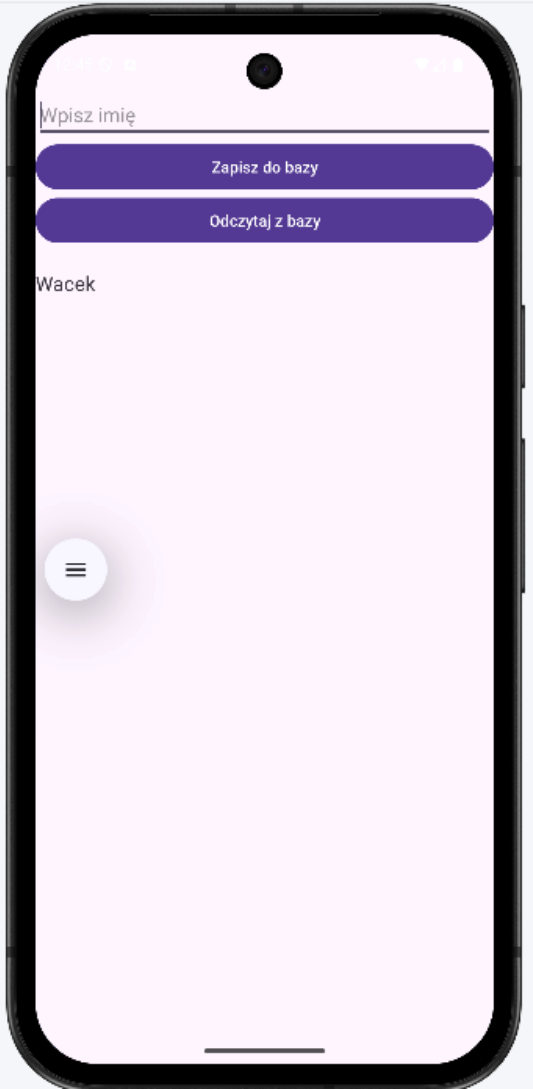
Współpraca języka Kotlin z plikiem i bazą danych

Klasa pomocnicza:

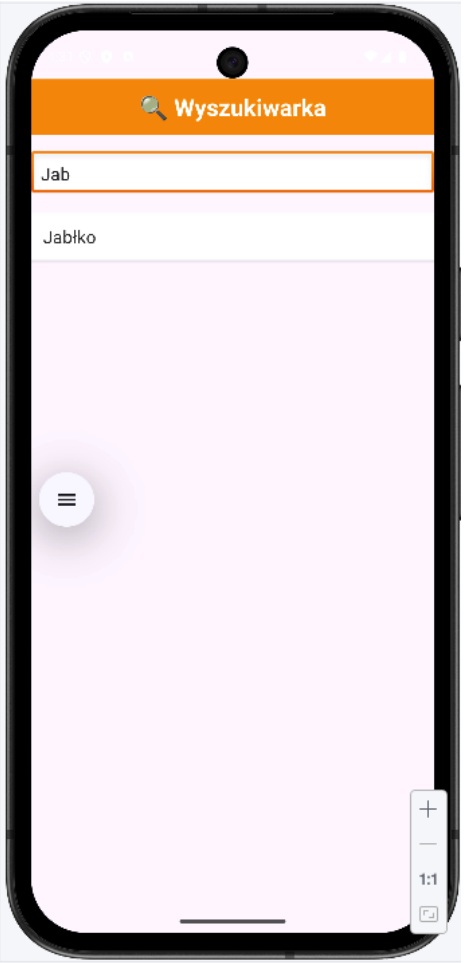
```
package com.example.databasedemo

import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_NAME = "moja_baza.db"
        private const val DATABASE_VERSION = 1
        private const val TABLE_NAME = "osoby"
        private const val COLUMN_ID = "id"
        private const val COLUMN_NAME = "imie"
    }

    override fun onCreate(db: SQLiteDatabase) {
        val createTable = ("CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT," +
                "$COLUMN_NAME TEXT)")
        db.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    // 👉 Zapis do bazy
    fun insertName(imie: String): Boolean {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_NAME, imie)
        val result = db.insert(TABLE_NAME, null, values)
        db.close()
        return result != -1L
    }

    // 👉 Odczyt z bazy (lista imion)
    fun getAllNames(): List<String> {
        val lista = ArrayList<String>()
        val db = readableDatabase
        val cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
        if (cursor.moveToFirst()) {
            do {
                val imie = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_NAME))
                lista.add(imie)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return lista
    }
}
```

Współpraca języka Kotlin z plikiem i bazą danych

```
package com.example.databasedemo

import
androidx.appcompat.app.AppCompatActivi
ty
import android.os.Bundle
import android.widget.*

class MainActivity :
AppCompatActivity() {

    lateinit var db: DatabaseHelper
    lateinit var editText: EditText
    lateinit var buttonSave: Button
    lateinit var buttonLoad: Button
    lateinit var textView: TextView

    override fun
onCreate(savedInstanceState: Bundle?)
{

super.onCreate(savedInstanceState)

setContentView(R.layout.activity_main)

        db = DatabaseHelper(this)

        editText =
findViewById(R.id.editTextName)
        buttonSave =
findViewById(R.id.buttonSave)
        buttonLoad =
findViewById(R.id.buttonLoad)
        textView =
findViewById(R.id.textViewData)

        // 👉 Zapis do bazy
        buttonSave.setOnClickListener
{
            val name =
editText.text.toString()
            if (name.isNotEmpty()) {
                val success =
db.insertName(name)
                if (success) {

Toast.makeText(this, "Zapisano!",
Toast.LENGTH_SHORT).show()

editText.text.clear()
                } else {

Toast.makeText(this, "Błąd zapisu!",
Toast.LENGTH_SHORT).show()
                }
            }
        }
```



Wyszukiwarka



Wyszukiwarka z listy.

```
package com.example.wyszukiwarka

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import android.widget.EditText

class MainActivity : AppCompatActivity() {

    private lateinit var recyclerView: RecyclerView
    private lateinit var editTextSearch: EditText
    private lateinit var adapter: ItemAdapter

    private val listaElementow = listOf(
        "Jabłko", "Banan", "Gruszka", "Pomarańcza", "Winogrono",
        "Arbuz", "Truskawka", "Kiwi", "Mango", "Ananas"
    )

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        recyclerView = findViewById(R.id.recyclerView)
        editTextSearch = findViewById(R.id.editTextSearch)

        recyclerView.layoutManager = LinearLayoutManager(this)
        adapter = ItemAdapter(listaElementow)
        recyclerView.adapter = adapter

        // Obsługa wyszukiwania
        editTextSearch.addTextChangedListener(object :
TextWatcher {
            override fun afterTextChanged(s: Editable?) {}

            override fun beforeTextChanged(s: CharSequence?, start:
Int, count: Int, after: Int) {}

            override fun onTextChanged(s: CharSequence?, start: Int,
before: Int, count: Int) {
                val filteredList = listaElementow.filter {
                    it.contains(s.toString(), ignoreCase = true)
                }
                adapter.updateList(filteredList)
            }
        })
    }
}
```

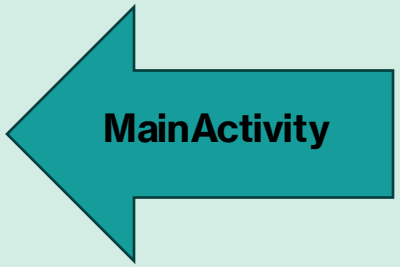
Wyszukiwarka

```
package com.example.wyszukiwarka

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class ItemAdapter(private var lista: List<String>) :
    RecyclerView.Adapter<ItemAdapter.ItemViewHolder>() {

    class ItemViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val textView: TextView = itemView.findViewById(R.id.textViewItem)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ItemViewHolder {
        val view =
            LayoutInflater.from(parent.context).inflate(R.layout.item_row, parent, false)
        return ItemViewHolder(view)
    }

    override fun getItemCount(): Int = lista.size

    override fun onBindViewHolder(holder: ItemViewHolder, position: Int) {
        holder.textView.text = lista[position]
    }

    // Funkcja aktualizująca listę (po filtrze)
    fun updateList(newList: List<String>) {
        lista = newList
        notifyDataSetChanged()
    }
}
```


```
package com.example.wyszukiwarka

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class ItemAdapter(private var lista: List<String>) :
    RecyclerView.Adapter<ItemAdapter.ItemViewHolder>() {

    class ItemViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val textView: TextView = itemView.findViewById(R.id.textViewItem)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ItemViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_row,
parent, false)
        return ItemViewHolder(view)
    }

    override fun getItemCount(): Int = lista.size

    override fun onBindViewHolder(holder: ItemViewHolder, position: Int) {
        holder.textView.text = lista[position]
    }

    fun updateList(newList: List<String>) {
        lista = newList
        notifyDataSetChanged()
    }
}
```

Wyszukiwarka

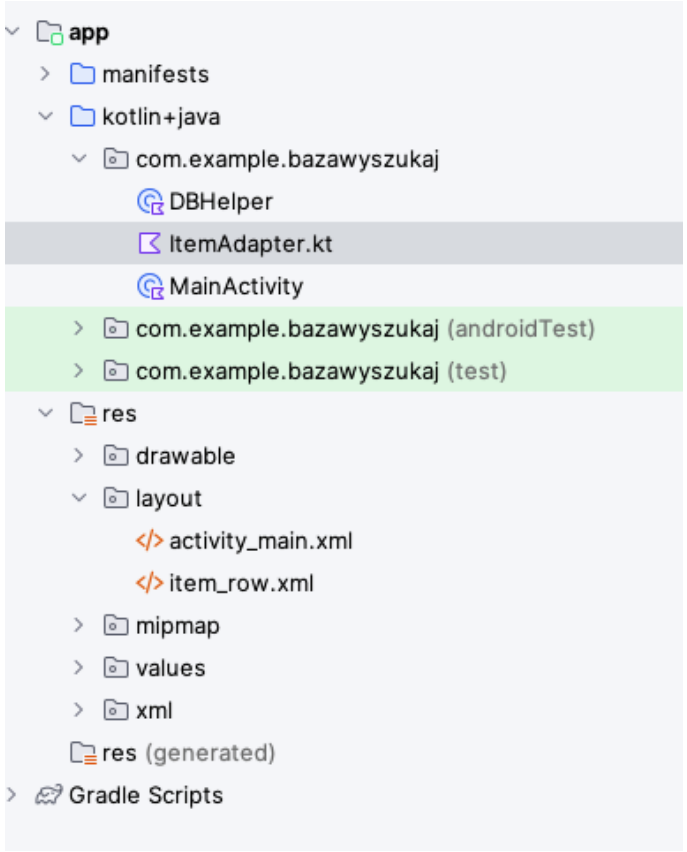
```
package com.example.wyszukiwarka

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import android.widget.EditText
import java.io.BufferedReader
import java.io.InputStreamReader

class MainActivity : AppCompatActivity() {

    private lateinit var recyclerView: RecyclerView
    private lateinit var editTextSearch: EditText
    private lateinit var adapter: ItemAdapter
    private var listaElementow: List<String> = listOf()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        recyclerView = findViewById(R.id.recyclerView)
        editTextSearch = findViewById(R.id.editTextSearch)

        // Wczytanie danych z pliku assets/dane.txt
        listaElementow = wczytajDaneZPliku("dane.txt")

        recyclerView.layoutManager = LinearLayoutManager(this)
        adapter = ItemAdapter(listaElementow)
        recyclerView.adapter = adapter

        // Obsługa wyszukiwania
        editTextSearch.addTextChangedListener(object : TextWatcher {
            override fun afterTextChanged(s: Editable?) {}
            override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int)
{}
            override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
                val filteredList = listaElementow.filter {
                    it.contains(s.toString(), ignoreCase = true)
                }
                adapter.updateList(filteredList)
            }
        })
    }

    private fun wczytajDaneZPliku(nazwaPliku: String): List<String> {
        val lista = mutableListOf<String>()
        try {
            val reader = BufferedReader(InputStreamReader(assets.open(nazwaPliku)))
            var linia: String? = reader.readLine()
            while (linia != null) {
                if (linia.isNotBlank()) lista.add(linia)
                linia = reader.readLine()
            }
            reader.close()
        } catch (e: Exception) {
            e.printStackTrace()
        }
        return lista
    }
}
```

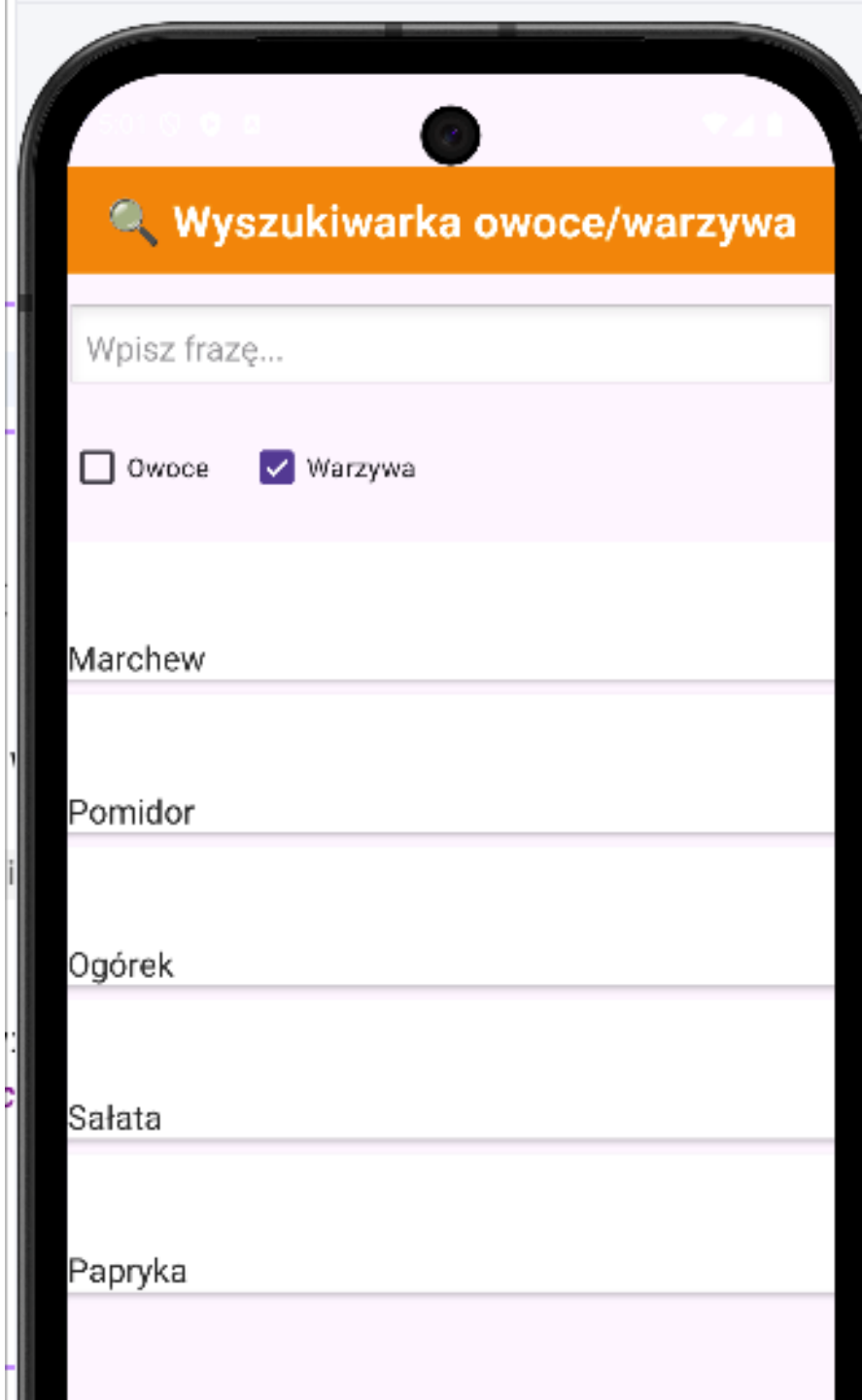


Wyszukiwarka

Wyszukiwarka danych z bazy SQLite

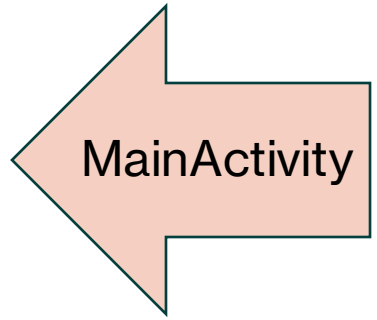
Wyszukuje dane w bazie za pomocą wpisywania i wyboru.





Wyszukiwarka

```
ackage com.example.bazawyszukaj

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import android.widget.CheckBox
import android.widget.EditText
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {

    private lateinit var recyclerView: RecyclerView
    private lateinit var editTextSearch: EditText
    private lateinit var checkBoxOwoce: CheckBox
    private lateinit var checkBoxWarzywa: CheckBox
    private lateinit var adapter: ItemAdapter
    private lateinit var dbHelper: DBHelper
    private var listaElementow: List<Element> = listOf()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        recyclerView = findViewById(R.id.recyclerView)
        editTextSearch = findViewById(R.id.editTextSearch)
        checkBoxOwoce = findViewById(R.id.checkBoxOwoce)
        checkBoxWarzywa = findViewById(R.id.checkBoxWarzywa)

        dbHelper = DBHelper(this)
        listaElementow = dbHelper.getAllItems()

        recyclerView.layoutManager = LinearLayoutManager(this)
        adapter = ItemAdapter(listaElementow)
        recyclerView.adapter = adapter

        editTextSearch.addTextChangedListener(object : TextWatcher
{
            override fun afterTextChanged(s: Editable?) { filtrujListe() }
            override fun beforeTextChanged(s: CharSequence?, start:
Int, count: Int, after: Int) {}
            override fun onTextChanged(s: CharSequence?, start: Int,
before: Int, count: Int) {}
        })

        checkBoxOwoce.setOnCheckedChangeListener { _, _ ->
filtrujListe() }
        checkBoxWarzywa.setOnCheckedChangeListener { _, _ ->
filtrujListe() }
    }

    private fun filtrujListe() {
        val tekst = editTextSearch.text.toString()
        val pokazOwoce = checkBoxOwoce.isChecked
        val pokazWarzywa = checkBoxWarzywa.isChecked

        val filteredList = listaElementow.filter {
            it.nazwa.contains(tekst, ignoreCase = true) &&
                    ((it.typ == "owoc" && pokazOwoce) || (it.typ ==
"warzywo" && pokazWarzywa))
        }
        adapter.updateList(filteredList)
    }
}
```



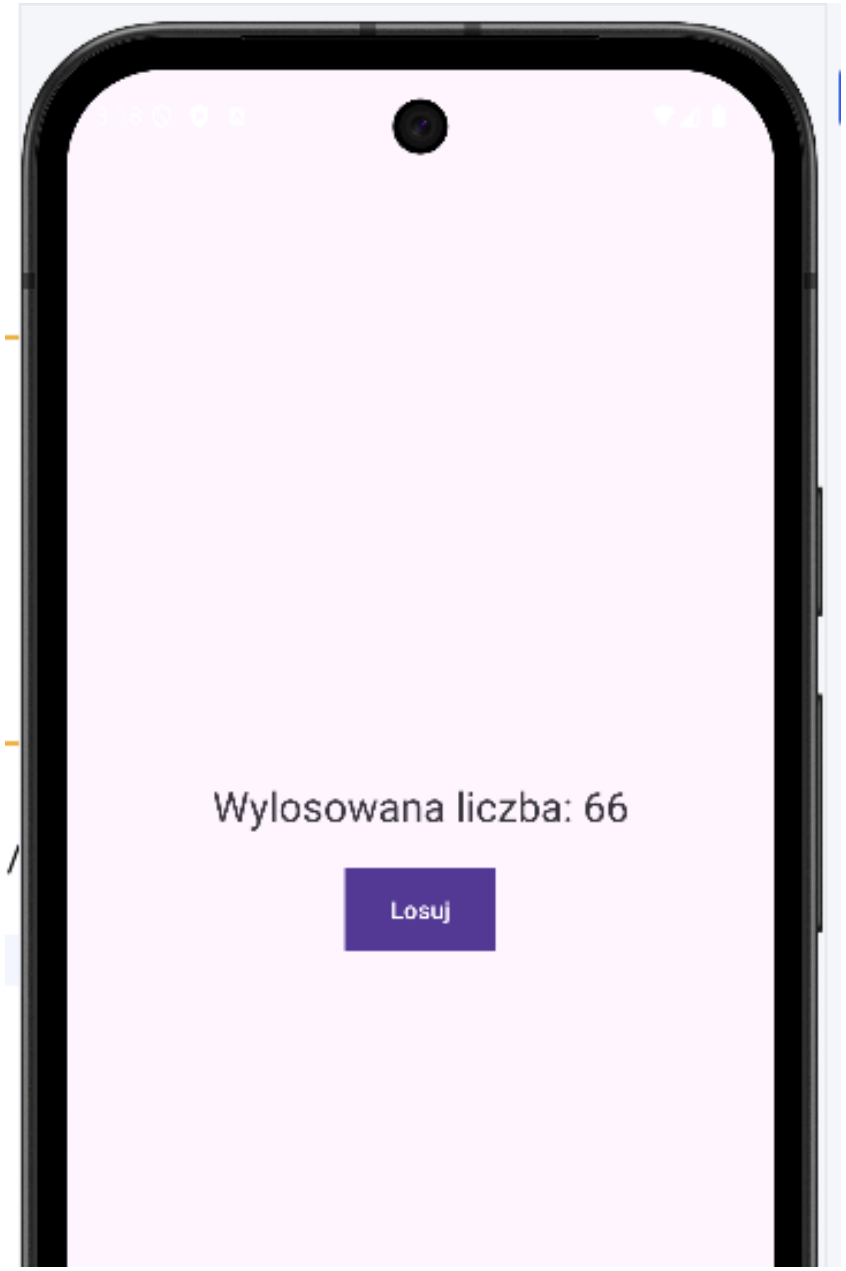
Wyszukiwarka

Adapter

```
package com.example.bazawyszukaj

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

data class Element(val nazwa: String, val typ: String)

class ItemAdapter(private var lista: List<Element>) :
    RecyclerView.Adapter<ItemAdapter.ItemViewHolder>() {

    class ItemViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val textView: TextView = itemView.findViewById(R.id.textViewItem)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ItemViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_row, parent, false)
        return ItemViewHolder(view)
    }

    override fun getItemCount(): Int = lista.size

    override fun onBindViewHolder(holder: ItemViewHolder, position: Int) {
        val element = lista[position]
        holder.textView.text = element.nazwa
    }

    fun updateList(newList: List<Element>) {
        lista = newList
        notifyDataSetChanged()
    }
}
```

Helper

```
package com.example.bazawyszukaj

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import android.content.ContentValues

class DBHelper(context: Context) :
    SQLiteOpenHelper(context, "wyszukiwarka.db", null, 2)
{

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = """
            CREATE TABLE items (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nazwa TEXT,
                typ TEXT
            )
        """.trimIndent()
        db?.execSQL(createTable)

        val dane = listOf(
            Element("Jabłko", "owoc"),
            Element("Banan", "owoc"),
            Element("Gruszka", "owoc"),
            Element("Pomarańcza", "owoc"),
            Element("Winogrono", "owoc"),
            Element("Arbuz", "owoc"),
            Element("Truskawka", "owoc"),
            Element("Kiwi", "owoc"),
            Element("Mango", "owoc"),
            Element("Ananas", "owoc"),
            Element("Marchew", "warzywo"),
            Element("Pomidor", "warzywo"),
            Element("Ogórek", "warzywo"),
            Element("Sałata", "warzywo"),
            Element("Papryka", "warzywo")
        )

        for (item in dane) {
            val cv = ContentValues()
            cv.put("nazwa", item.nazwa)
            cv.put("typ", item.typ)
            db?.insert("items", null, cv)
        }
    }

    override fun onUpgrade(db: SQLiteDatabase?,
oldVersion: Int, newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS items")
        onCreate(db)
    }

    fun getAllItems(): List<Element> {
        val lista = mutableListOf<Element>()
        val db = readableDatabase
        val cursor = db.rawQuery("SELECT * FROM items",
null)
        if (cursor.moveToFirst()) {
            do {
                val nazwa =
cursor.getString(cursor.getColumnIndexOrThrow("nazwa
"))
                val typ =
cursor.getString(cursor.getColumnIndexOrThrow("typ"))
                lista.add(Element(nazwa, typ))
            } while (cursor.moveToNext())
        }
        cursor.close()
        return lista
    }
}
```



Zadania do samodzielnego wykonania

Zadanie 1

Program losuje i wyświetla dowolną liczbę z przedziału <0,100> a następnie wyświetla w etykiecie.

Zadania do samodzielnego wykonania

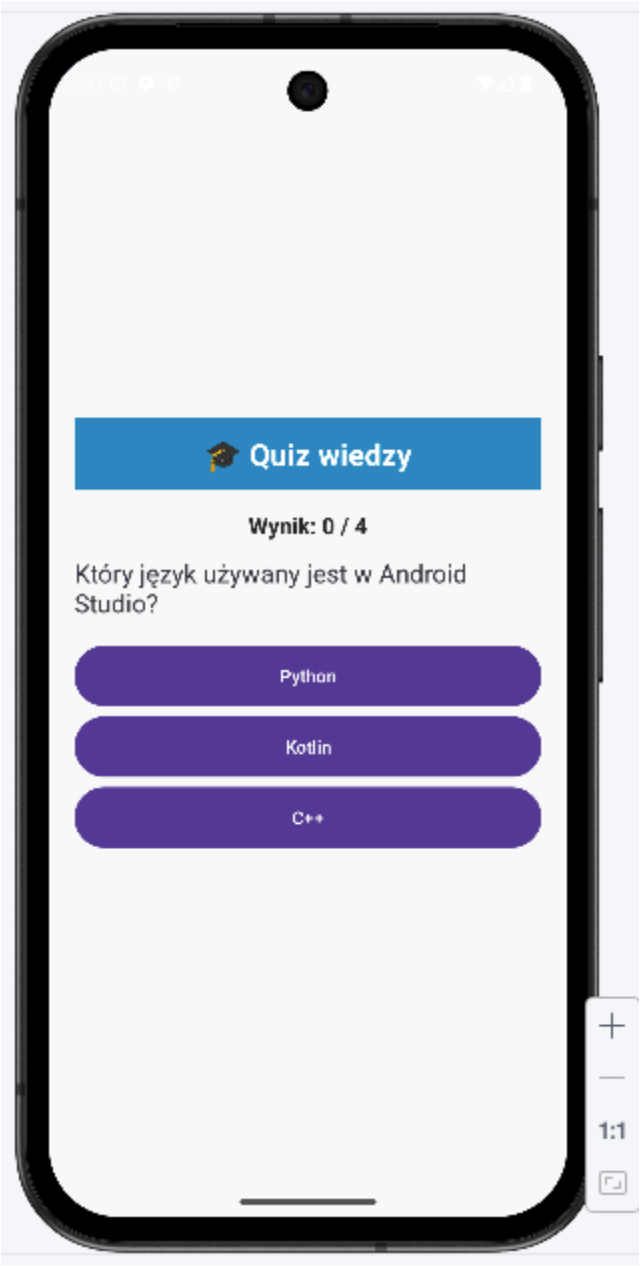
Zadanie 2

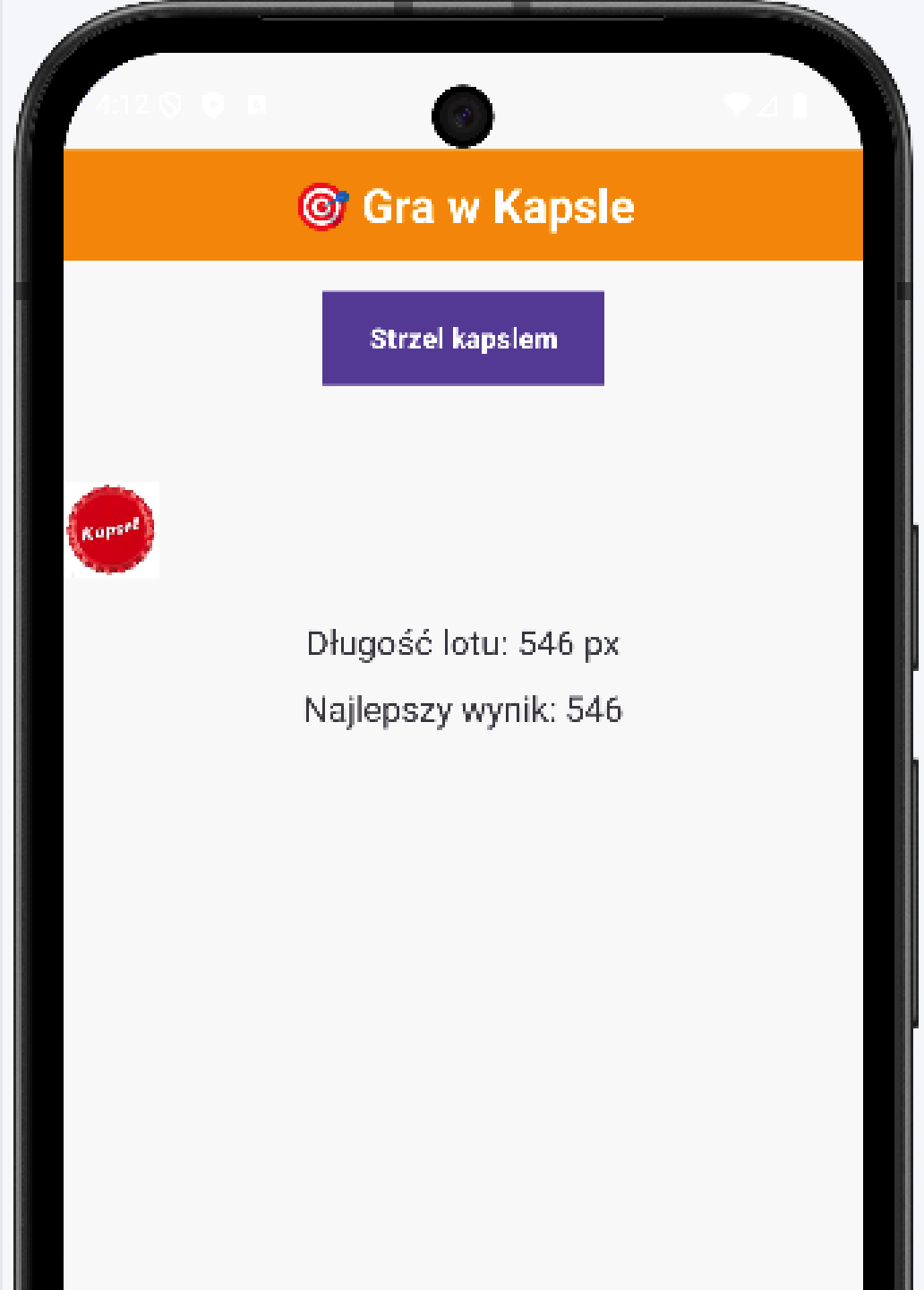
Lista zadań z wyświetlaniem wprowadzonych danych. Zawiera nagłówek z kolorem holo_light_blue oraz ikonę notatnika.

Zadania do samodzielnego wykonania

Zadanie 3 (na 5)

Napisz aplikację w formie quizu o programowaniu. Za każdą poprawną odpowiedź jest jeden punkt. Aplikacja zawiera wyświetlanie wyniku.



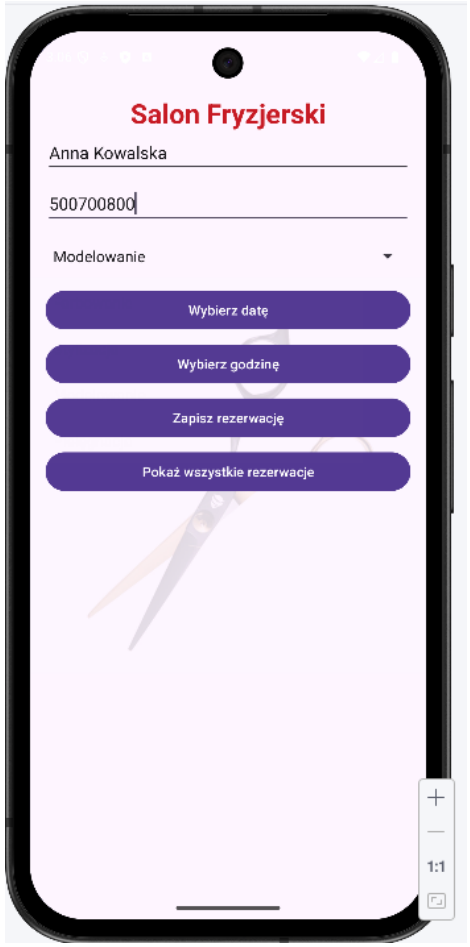


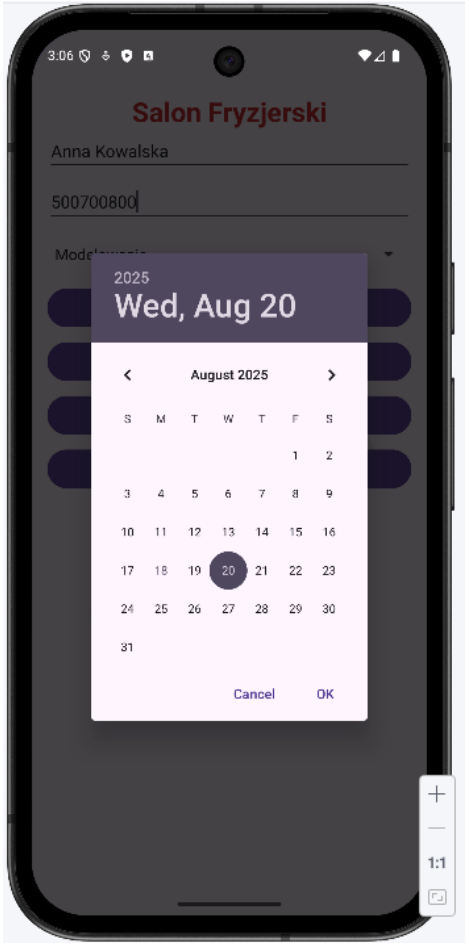
Zadania do samodzielnego wykonania

Zadanie 4 (zadanie na 5)

Gra w kapsle. Użytkownik za pomocą przycisku strzela kapslem. Kapsel porusza się w prawo. Odczytywana jest długość lotu w pikselach i zapisywany jest najlepszy wynik.

Zadania do samodzielnego wykonania

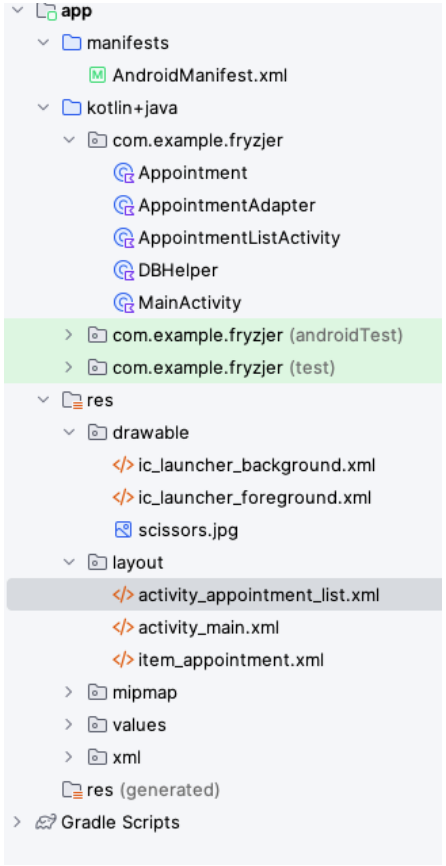


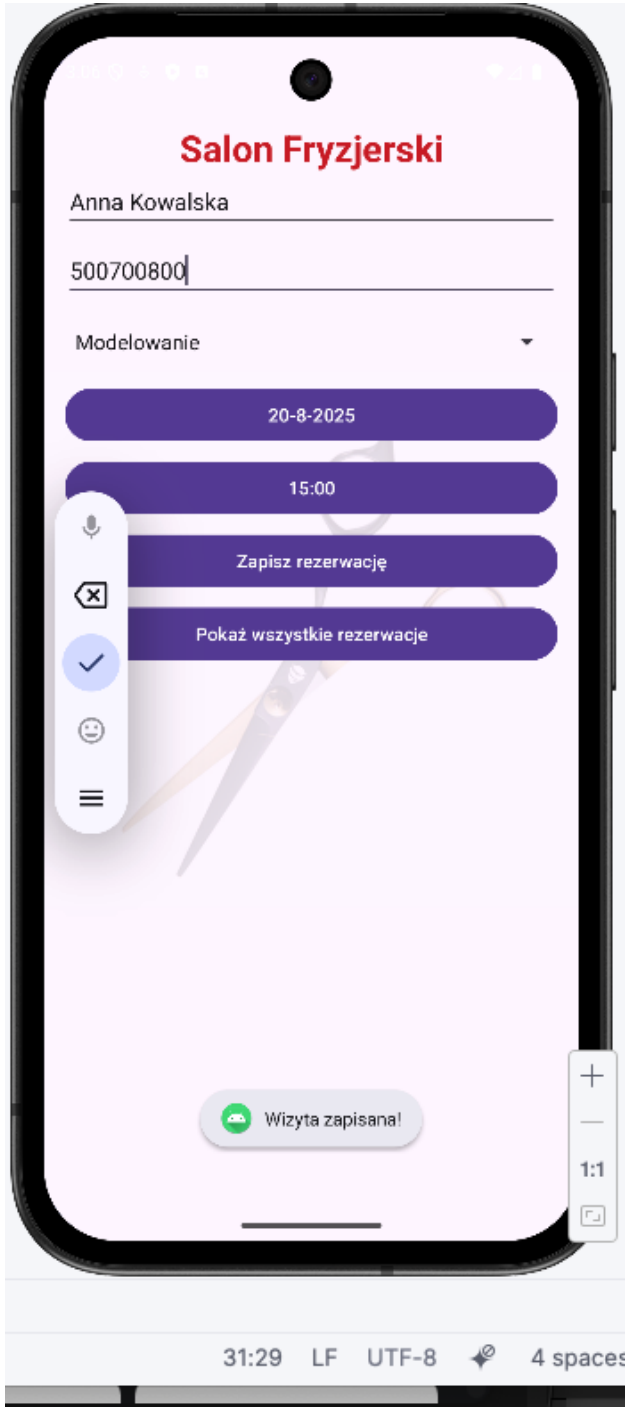


Zadanie 5 (na 5)

Aplikacja do zarządzania wizytami u fryzjera. Zawiera listę klientów, terminy oraz rodzaj usługi.

Zadania do samodzielnego wykonania





Podsumowanie - pytania

1. Jak nazywamy pojedynczy obiekt bez instancji?
2. Jaki rodzaj pętli lub instrukcji warunkowej zastępuje WHEN?
3. Do czego służy APPLY?
4. Do czego służy MAP i FILTER?
5. Co znajduje się w pliku TOML?
6. Co nie jest wymagane przy zastosowaniu Jetpack Compose?
7. Co musi zawierać Build Module.APP?
8. Czy w przypadku języka Kotlin musimy stosować dodatkowe zależności?