POLITECHNIKA RZESZOWSKA
KATEDRA INFORMATYKI I AUTOMATYKI
DR INŻ. MATEUSZ POMIANEK

ĆWICZENIE LABORATORYJNE - KOTLIN 1

Konfiguracja środowiska deweloperskiego i pierwsza aplikacja Android

Programowanie Aplikacji Mobilnych (Android / Kotlin / Jetpack Compose)

Spis treści

1. Instalacja Android Studio

1.1. Pobieranie i instalacja

Android Studio **Panda** jest oficjalnym IDE do tworzenia aplikacji Android. Pobieramy ze strony:
https://developer.android.com/studio

#	Akcja	Szczegóły
1	**Pobierz instalator**	Wejdź na developer.android.com/studio → kliknij "Download Android Studio". Akceptujesz umowę licencyjną Google. Rozmiar: ~1.2 GB.
2	**Windows**	Uruchom .exe jako administrator. Wybierz opcję "Android Studio" + "Android Virtual Device". Ścieżka domyślna: *C:\Program Files\Android\Android Studio*
3	**macOS**	Otwórz .dmg → przeciągnij do Applications. Pierwsze uruchomienie: kliknij prawym → Open (ominięcie Gatekeeper). Simulator wymaga Xcode Command Line Tools.
4	**Linux (Ubuntu)**	Rozpakowujesz archiwum .tar.gz i uruchamiasz bin/studio.sh. Dodaj do PATH lub utwórz desktop shortcut. *sudo apt-get install -y libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1*

#	Akcja	Szczegóły
5	**Setup Wizard**	Przy pierwszym uruchomieniu: Standard setup → Next. Pobiera SDK (Android 14, Build Tools, Emulator). Czas: 5-20 minut (zależnie od łącza).

Uwaga - problemy z instalacją

Jeśli emulator nie uruchomi się - sprawdź, czy wirtualizacja sprzętowa jest włączona w BIOS/UEFI (Intel VT-x / AMD-V).

Windows: sprawdź "Hyper-V" w funkcjach Windows lub WHPX w ustawieniach AVD Manager. Korporacyjne laptopy mogą blokować VT-x - skontaktuj się z administratorem lub użyj urządzenia fizycznego.

Linux: dodaj użytkownika do grupy kvm: ***sudo usermod -aG kvm $USER*** (wymaga wylogowania)

1.2. Konfiguracja SDK i narzędzi

Po zainstalowaniu Android Studio otwieramy SDK Manager i weryfikujemy komponenty:

Ścieżka: File → Settings → Android SDK (lub Android Studio → Settings na macOS)

SDK Platforms	Android 14 (API 34) zainstalowany Android 15 (API 35) zainstalowany
SDK Tools	Android SDK Build-Tools 35.x Android Emulator Android SDK Platform-Tools Intel x86 Emulator Accelerator (HAXM) - Windows/Mac
SDK Command-line Tools	Zainstaluj: latest version - umożliwia użycie sdkmanager z terminala
Google Play Services	Opcjonalne dla ćwiczenia, wymagane przy Google Maps / Firebase w projekcie semestralnym

Ścieżka SDK

Zapamiętaj gdzie jest zainstalowany SDK - domyślnie: ~/Library/Android/sdk (macOS) lub C:\Users\<user>\AppData\Local\Android\Sdk (Windows).
Możesz zmienić przez: File → Project Structure → SDK Location.
Zmienna ANDROID_HOME w systemie jest wymagana przez narzędzia CLI (np. Flutter, React Native).

2. Tworzenie wirtualnego urządzenia (AVD)

AVD (Android Virtual Device) to emulator systemu Android. Tworzymy go przez AVD Manager wbudowany w Android Studio.

2.1. Konfiguracja AVD Manager

#	Akcja	Szczegóły i parametry
1	**Otwórz AVD Manager**	Tools → Device Manager (lub ikona telefonu w prawym górnym panelu → Manage Devices). Kliknij + Create Virtual Device.

#	Akcja	Szczegóły i parametry
2	**Wybierz hardware**	Wybierz **Pixel 9 Pro** z kategorii Phone. Rozmiar: 6.3", gęstość: 480 dpi. Kliknij Next.
3	**Wybierz system image**	Zakładka **Recommended** → pobierz/wybierz **API Level 35 (Android 15)** lub API Level 34 (Android 14). Wybierz architekturę x86_64 (dla PC) lub arm64 (Apple Silicon Mac).
4	**Konfiguracja AVD**	AVD Name: Pixel_9_Pro_API35 RAM: 2048 MB (zwiększ do 4096 MB jeśli komputer ma ≥ 16 GB RAM) Internal Storage: 2048 MB Enable Device Frame: tak Startup orientation: Portrait
5	**Uruchom emulator**	Kliknij zielony trójkąt przy AVD. Pierwsze uruchomienie: 1-3 minuty. Patrz na stan w **'Device Manager'**. Emulator musi osiągnąć stan Running.

2.2. Weryfikacja połączenia ADB

W terminalu Android Studio **View → Tool Windows → Terminal**

Terminal

```
HelloMobile/
adb devices

# Oczekiwany wynik:
# List of devices attached
# emulator-5554   device
```

Jeśli status to `offline` → zrestartuj emulator.

Jeśli status to `unauthorized` → odblokuj emulator i zaakceptuj klucz RSA.

Przydatne polecenia ADB do zapamiętania

`adb devices`	Lista podłączonych urządzeń/emulatorów
`adb kill-server, adb start-server`	Restart serwera ADB (gdy urządzenie „offline”)
`adb logcat`	Wyświetl logi systemowe (Ctrl+C aby przerwać)
`adb logcat --pid=<PID>`	Logi tylko dla konkretnej aplikacji (po PID)
`adb logcat -c`	Wyczyść bufor logów
`adb install nazwa_aplikacji.apk`	Zainstaluj plik APK
`adb install -r nazwa_aplikacji.apk`	Zainstaluj ponownie (nadpisz istniejącą)
`adb uninstall nazwa.pakietu`	Odinstaluj aplikację
`adb shell pm list packages`	Wyświetl listę zainstalowanych pakietów
`adb shell`	Otwórz powłokę systemu Android
`adb shell <polecenie>`	Wykonaj pojedyncze polecenie w shell
`adb shell getprop ro.build.version.re-`	Sprawdź wersję Androida

`lease`	
`adb shell getprop ro.build.version.sdk`	Sprawdź poziom API
`adb shell getprop ro.product.model`	Sprawdź model urządzenia
`adb reverse tcp:8080 tcp:8080`	Port forwarding (przydatne przy lokalnym API)
`adb forward tcp:8080 tcp:8080`	Przekierowanie portu z urządzenia na komputer
`adb pull /ścieżka/plik.txt`	Skopiuj plik z urządzenia na komputer
`adb push plik.txt /sdcard/`	Skopiuj plik z komputera na urządzenie
`adb shell screencap -p /sdcard/screen.png`	Zrzut ekranu urządzenia
`adb shell screenrecord /sdcard/video.mp4`	Nagranie ekranu (max 180 s)
`adb reboot`	Restart urządzenia

3. Tworzenie projektu Android

3.1. New Project Wizard

#	Akcja	Szczegóły
1	**Nowy projekt**	File → New → New Project (lub **Ctrl+Shift+N** Windows / **⌘Shift+N** macOS)
2	**Szablon**	Wybierz kategorię **Phone and Tablet** → szablon **Empty Activity**. To domyślny punkt startowy dla Jetpack Compose. Kliknij Next.
3	**Konfiguracja**	Name: np. **HelloMobile**
		Package name: pl.edu.prz.hellomobile (zmień na własną domenę odwróconą).
		Save location: do folderu na dysku (unikaj spacji i polskich znaków w ścieżce!).
		Language: Kotlin, minimum SDK: API 26 (Android 8.0) - pokrywa ~94% aktywnych urządzeń.
		Build configuration language: Kotlin DSL (.kts)
4	**Finish**	Kliknij Finish. Gradle synchronizuje projekt. Pierwsze synchronizowanie: 2-5 minut. Patrz na pasek postępu na dole okna.

Konwencja dla Package name

Package name to unikalny identyfikator aplikacji w Google Play. Konwencja: odwrócona domena + nazwa app, np. com.przykład.mojaapka.

Dla ćwiczenia użyj formatu: pl.put.pam.imienazwisko (bez polskich znaków i spacji).

Package name nie może być później zmieniony bez utraty danych użytkownika na Google Play.

3.2. Struktura wygenerowanego projektu

Po zakończeniu synchronizacji Gradle widzisz w panelu Project (**View → Tool Windows → Project**) następującą strukturę:

Kotlin / Struktura projektu

```
HelloMobile/
├── app/                              ← moduł aplikacji
│   ├── src/
│   │   ├── main/
│   │   │   ├── java/pl/edu/.../
│   │   │   │   ├── MainActivity.kt   ← główna Activity (ENTRY POINT)
│   │   │   │   └── ui/theme/         ← pliki motywu Material3
│   │   │   ├── res/
│   │   │   │   ├── drawable/         ← grafika wektorowa (ikony)
│   │   │   │   ├── mipmap-*/         ← ikony aplikacji (różne gęstości)
│   │   │   │   ├── values/
│   │   │   │   │   ├── strings.xml   ← teksty (i18n)
│   │   │   │   │   ├── colors.xml    ← definicje kolorów
│   │   │   │   │   └── themes.xml    ← motywy Material3
│   │   │   │   └── xml/              ← konfiguracja backup/network
│   │   │   └── AndroidManifest.xml   ← OBOWIĄZKOWY manifest aplikacji
│   │   └── test/ + androidTest/      ← testy jednostkowe i instrumentalne
│   ├── build.gradle.kts              ← zależności modułu
│   └── proguard-rules.pro            ← reguły obfuskacji
├── gradle/
│   └── libs.versions.toml            ← Version Catalog (wersje zależności)
├── build.gradle.kts                  ← root build config
└── settings.gradle.kts               ← deklaracja modułów
```

Najważniejsze pliki do zrozumienia na tym etapie:

AndroidManifest.xml	Serce aplikacji. Deklaruje: Activity, uprawnienia (permissions), konfigurację sprzętową, meta-dane. Wymagany przez system Android.
MainActivity.kt	Punkt wejścia. Klasa dziedziczy po ComponentActivity. Metoda setContent { } definiuje drzewo UI jako funkcje Composable.
build.gradle.kts (app)	Zależności (libraries), konfiguracja kompilacji: compileSdk, minSdk, versionCode, versionName. Każda biblioteka Jetpack dodawana tutaj.
libs.versions.toml	Centralne miejsce definicji wersji bibliotek. Zastępuje rozrzucone numery wersji po plikach gradle. Obowiązkowy standard od AGP 8.
ui/theme/Theme.kt	Konfiguracja motywu Material3: kolory (light/dark), typografia, kształty. Plik generowany automatycznie przez Compose.

4. Analiza i modyfikacja kodu startowego

4.1. Analiza MainActivity.kt

Otwórz plik **app/src/main/java/.../MainActivity.kt**. Przeanalizuj każdą linię:

MainActivity.kt

```
package pl.edu.prz.hellomobile
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge     // fullscreen "edge to edge"
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
```

```
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import pl.edu.prz.hellomobile.ui.theme.HelloMobileTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
 enableEdgeToEdge()                                                    // (1)
 setContent {                                                          // (2)
           HelloMobileTheme {                                          // (3)
          Scaffold(                                                    // (4)
                 modifier = Modifier.fillMaxSize()) { innerPadding ->
                     Greeting(                                        //(5)
                         name = "Android",
                         modifier = Modifier.padding(innerPadding)
             )
                 }
           }
        }
    }
}

@Composable                                                            // (6)
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",                                         // (7)
        modifier = modifier
    )
}

@Preview(showBackground = true)                                        // (8)
@Composable
    fun GreetingPreview() {
        HelloMobileTheme {
            Greeting("Android")
        }
    }
```

(1) enableEdgeToEdge()	Rozszerza aplikację na całą powierzchnię ekranu, włącznie z paskiem statusu i nawigacji. Standard od Android 15.
(2) setContent { }	Zastępuje stary setContentView(R.layout.xml). Przyjmuje funkcję Composable jako korzeń drzewa UI.
(3) HelloMobileTheme { }	Opakowuje UI w Material3 Design System, dostarcza kolory, typografię, kształty. Definicja w ui/theme/.
(4) Scaffold { }	Podstawowy layout Material3: topBar, bottomBar, floatingActionButton, content. innerPadding uwzględnia status bar i nav bar.
(5) Greeting(...)	Wywołanie własnej funkcji Composable. Compose buduje drzewo UI z takich wywołań.
(6) @Composable	Anotacja oznaczająca funkcję jako element UI Compose. Może być wywoływana tylko z innych @Composable lub setContent.
(7) "Hello $name!"	Kotlin string template - interpolacja zmiennej. Odpowiednik String.format() z Javy, ale czytelniejszy.
(8) @Preview	Generuje podgląd w Compose Preview (prawy panel Xcode-like). Nie wymaga emulatora. showBackground = true dodaje białe tło.

4.2. Zadanie podstawowe - modyfikacja Greeting

Zadanie 4.2 - Spersonalizuj ekran powitalny

Zmodyfikuj funkcję Greeting tak, aby wyświetlała Twoje imię i nazwisko.

Dodaj drugi Text z nazwą swojej uczelni.

Zmień kolor pierwszego tekstu na zielony: Color(0xFF00C47A).

Użyj Compose Preview (zakładka Design w Android Studio) do podglądu bez uruchamiania emulatora.
Oczekiwany efekt po modyfikacji - *patrz przykład kodu poniżej.*

Kotlin - Greeting.kt

```
@Composable fun Greeting(name: String, modifier: Modifier=Modifier) {

 // Zadanie: zmodyfikuj ten kod
 Column(modifier=modifier.padding(16.dp),
    verticalArrangement=Arrangement.spacedBy(8.dp)) {
    Text(text="Cześć, $name!",
       style=MaterialTheme.typography.headlineMedium,
       color=Color(0xFF00C47A) // ← zmień kolor       )
       Text(text="Wydział Elektrotechniki i Informatyki",
          style=MaterialTheme.typography.bodyMedium,
          color=MaterialTheme.colorScheme.onSurfaceVariant)
    }
 }

 // Pamiętaj o importach:
 // import androidx.compose.foundation.layout.Column
 // import androidx.compose.foundation.layout.Arrangement
 // import androidx.compose.foundation.layout.padding
 // import androidx.compose.ui.graphics.Color
 // import androidx.compose.ui.unit.dp
```

5. Rozbudowa aplikacji o interaktywny licznik

5.1. Niezbędne podstawy Kotlina

Przed pisaniem kodu Compose przypomnijmy niezbędne elementy języka Kotlin:

Przypomnienie podstaw Kotlina

```
// 1. VAL vs VAR
val imię = "Anna"        // niezmienne

var licznik = 0          // zmienne
licznik++                // OK: modyfikacja var

// 2. STRING TEMPLATES
val wiadomość = "Masz $licznik wiadomości"
// $zmienna

val info = "Wersja: ${BuildConfig.VERSION_NAME}"
// ${wyrażenie}

// 3. FUNKCJA z domyślnym parametrem
fun pozdrow(imię: String, prefiks: String = "Cześć") =
    "$prefiks, $imię!"
```

```
// 4. NULL SAFETY
var tekst: String? = null   // może być null
val dług = tekst?.length    // safe call: null jeśli tekst == null

val dług2 = tekst?.length ?: 0
// Elvis: 0 jeśli null

// 5. LAMBDA
val przycisk: () -> Unit = { println("Kliknięto!") }

// lub w skróconej formie jako ostatni parametr:
Button(
    onClick = { licznik++ }
) {
    Text("Kliknij")
}
```

5.2. Implementacja licznika - krok po kroku

Utwórz nowy plik Kotlin: kliknij prawym na pakiet w Project → New → Kotlin Class/File → File → nazwa CounterScreen.

Kotlin - CounterScreen.kt

```
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.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

@Composable
fun CounterScreen() {
  var count by remember { mutableStateOf(0) }
  val message = when {
    count == 0  -> "Zacznij liczyć!"
    count < 10  -> "Dobry start"
    count < 50  -> "Nie zatrzymuj się!"
    else        -> "Wow, $count - nieźle!"
  }
  Column(
    modifier = Modifier
      .fillMaxSize()
      .padding(24.dp),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
  ) {
    Text(
      text = "$count",
      fontSize = 80.sp,
      fontWeight = FontWeight.Bold,
      color = MaterialTheme.colorScheme.primary
    )
    Spacer(modifier = Modifier.height(32.dp))
    Row(horizontalArrangement = Arrangement.spacedBy(16.dp)) {
      FilledTonalButton(onClick = { if (count > 0) count-- }) {
        Text("-", fontSize = 20.sp)
      }
      OutlinedButton(onClick = { count = 0 }) {
        Text("Reset")
      }
      Button(onClick = { count++ }) {
        Text("+", fontSize = 20.sp)
      }
    }
    Spacer(modifier = Modifier.height(24.dp))
```

```
        Text(
          text = message,
          style = MaterialTheme.typography.bodyLarge,
          color = MaterialTheme.colorScheme.onSurfaceVariant
        )
      }
    }
    @Preview(showBackground = true)
    @Composable
    fun CounterPreview() {
      HelloMobileTheme {
        CounterScreen()
      }
    }
```

5.3. Podłączenie CounterScreen do MainActivity

```
setContent {
  HelloMobileTheme {
    Surface(
      modifier = Modifier.fillMaxSize(),
      color = MaterialTheme.colorScheme.background
    ) {
      CounterScreen() // ← zamieniono Greeting na CounterScreen
    }
  }
}
```

#	Akcja	Uruchomienie i weryfikacja
1	**Buduj projekt**	Kliknij Build → Make Project (**Ctrl+F9**). Sprawdź panel Build w dolnej belce - brak błędów to wartość 0 errors.
2	**Uruchom na emulatorze**	Wybierz emulator Pixel 9 Pro w górnym pasku → kliknij Run (**Shift+F10** lub zielony trójkąt). Czas buildu: 30-90 sekund.
3	**Testuj interakcję**	Kliknij przyciski + i - na emulatorze. Obserwuj jak liczba zmienia się na ekranie w czasie rzeczywistym. Sprawdź komunikat przy 10, 50+.
4	**Live Edit**	Zmień tekst w jednym z przycisków (np. "Kliknij!" → "+1") i zapisz plik. Live Edit powinno automatycznie zaktualizować emulator **bez pełnego restartu**.

Jak działa State w Compose?

remember { mutableStateOf(0) } tworzy obiekt stanu powiązany z danym miejscem w drzewie Composable. Gdy count się zmienia, Compose wie, które części UI zależą od tej wartości i przerysowuje *tylko je* - nie cały ekran. To nazywa się inteligentnym rekomponowaniem (smart recomposition).

Compose 'obserwuje' dostęp do zmiennych state podczas kompozycji i zapamiętuje zależności. Zmiana stanu → rekomponowanie → nowy UI tree.

6. Zadania do wykonania

Poniższe zadania są wymagane do zaliczenia ćwiczenia. Wykonaj je w podanej kolejności, gdyż każde kolejne bazuje na poprzednim.

Zadanie 1 - Konfiguracja środowiska

Wymagania

1.1 Zainstaluj Android Studio i pokaż prowadzącemu ekran Welcome Screen z widoczną wersją (About → Android Studio Panda).

1.2 Utwórz i uruchom AVD - emulator Pixel 9 Pro lub Pixel 8 z API ≥ 34.
Pokaż działający emulator z ekranem głównym Androida.

1.3 W terminalu Android Studio wykonaj: adb devices - pokaż prowadzącemu wynik z widocznym emulator-xxxx device.

1.4 W SDK Manager (File → Settings → Android SDK) zrób screenshot pokazujący zainstalowane komponenty: SDK Platform 34 lub 35 + Build-Tools.

Zadanie 2 - Projekt i analiza kodu

Wymagania

2.1 Utwórz projekt HelloMobile zgodnie z sekcją 3.1. Package name np.: pl.prz.pam.<twoje_inicjały> (np. pl.put.pam.ajk).

2.2 Zmodyfikuj funkcję Greeting (sekcja 4.2). Wyświetl imię, nazwisko i wydział w trzech osobnych liniach tekstu z różnymi stylami.

2.3 Ustaw kolor tytułu na zgodny z paletą Material3. Użyj MaterialTheme.colorScheme.primary zamiast hardcoded Color().

2.4 Uruchom Compose Preview i zrób screenshot całego IDE z widocznym podglądem w panelu Design.

2.5 Opisz w komentarzu w kodzie co robi każda z anotacji: @Composable i @Preview.

Zadanie 3 - Licznik interaktywny

Wymagania

3.1 Zaimplementuj CounterScreen z sekcji 5.2, licznik z trzema przyciskami (+, Reset, -).

3.2 Dodaj walidację: przycisk - nie pozwala zejść poniżej 0 (zaimplementowane w przykładzie). Dodatkowo: przycisk + nie pozwala przekroczyć 99.

3.3 Dodaj wyświetlanie historii: pod licznikiem pokaż ostatnie 5 zmian w formie listy, np. '+1', '-1', 'Reset'. Użyj remember { mutableStateListOf() }.

3.4 Zmień wygląd przycisku + na wyróżniający się (Button z niestandardowymi kolorami; użyj ButtonDefaults.buttonColors()).

3.5 Uruchom na emulatorze i zademonstruj działanie prowadzącemu.

Zadanie 4 - Rozszerzenie

Wymagania

4.1 Dodaj TextField (pole tekstowe) nad licznikiem, w którym użytkownik wpisuje imię.

4.2 Powitanie nad licznikiem zmienia się dynamicznie wraz ze wpisywanym imieniem: "Cześć, [imię]! Twój wynik: [count]"

4.3 Obsłuż pustą wartość TextField - gdy pole jest puste, wyświetl "Cześć, nieznajomy!" zamiast "Cześć, !".

4.4 Użyj pamiętanego stanu: var name by remember { mutableStateOf("") }.

Wskazówka do Zadania 4: TextField: Poniżej przykład użycia OutlinedTextField w Compose.

Kotlin - hint do Zadania 4

```
setContent {
  HelloMobileTheme {
    Surface(
      modifier = Modifier.fillMaxSize(),
      color = MaterialTheme.colorScheme.background
    ) {
      CounterScreen() // ← zamieniono Greeting na CounterScreen
    }
  }
}var name by remember { mutableStateOf("") }

OutlinedTextField(
  value = name,
  onValueChange = { name = it }, // nowa wartość po każdym znaku
  label = { Text("Twoje imię") },
  singleLine = true,
  modifier = Modifier
    .fillMaxWidth()
    .padding(bottom = 16.dp)
)

// Dynamiczny tekst powitania
val greeting = if (name.isBlank()) {
  "Cześć, nieznajomy!"
} else {
  "Cześć, $name!"
}

Text(
  text = greeting,
  style = MaterialTheme.typography.titleMedium
)
```

7. Najczęstsze błędy i ich rozwiązania

Gradle sync failed	Sprawdź połączenie z Internetem. Kliknij **File → Sync Project with Gradle Files**. Jeśli błąd nadal występuje: **File → Invalidate Caches / Restart → Invalidate and Restart**.
Emulator nie startuje	Sprawdź: **BIOS → Intel VT-x** lub **AMD-V** włączone. Windows: **Task Manager → Performance → Virtualization: Enabled**. Alternatywa: połącz fizyczny telefon kablem USB i włącz **USB Debugging** w opcjach dewelopera.
Cannot resolve symbol 'Composable'	Import nie został dodany automatycznie. Naciśnij Alt+Enter na podkreślonej anotacji → **Add import.** Lub dodaj ręcznie: **import androidx.compose.runtime.Composable**
Build error: Unresolved reference	Sprawdź, czy wszystkie importy są na górze pliku. Sprawdź, czy piszesz w zakresie @Composable (nie poza funkcją). Kliknij **Build → Clean Project**, potem **Rebuild Project**.
Emulator bardzo wolny	Zwiększ RAM w AVD Manager (edytuj AVD → Show Advanced Settings → RAM: 4096 MB). Windows: upewnij się, że HAXM jest zainstalowany (SDK Manager → SDK Tools). Alternatywa: użyj fizycznego urządzenia Android z USB Debugging.
Live Edit nie działa	Upewnij się, że aplikacja jest uruchomiona (nie zatrzymana). Live Edit działa tylko dla @Composable - zmiany w logice wymagają pełnego buildu. Sprawdź: **Settings → Editor → Live Edit → Apply Code Changes on the Fly**.

9. Materiały dodatkowe

Oficjalna dokumentacja

Jetpack Compose	Oficjalny przewodnik: https://developer.android.com/compose
Codelabs Android	Interaktywne tutoriale Google: https://developer.android.com/codelabs
Kotlin Koans	Ćwiczenia z języka Kotlin online: https://kotlinlang.org/docs/koans.html
Material Design 3	Specyfikacja i komponenty: https://m3.material.io
Android API Reference	Pełna dokumentacja klas: https://developer.android.com/reference
Compose Layout Basics	Codelab: https://developer.android.com/codelabs/jetpack-compose-layouts

Polecane kursy wideo

Android Basics with Compose (Google) - *oficjalny kurs Google, darmowy, ~50 godzin*
youtube.com/@PhilippLackner - *bardzo aktualny, polecany przez społeczność*
youtube.com/@stevdza-san - *animacje, nawigacja, architecture*

10. Sprawozdanie

Sprawozdanie nie jest wymagane dla tego ćwiczenia.

Repozytorium GitHub: *prześlij link do repozytorium na Moodle.*

#	Akcja	Instrukcja wysłania repozytorium
1	**Utwórz repo**	Utwórz prywatne repozytorium na github.com o nazwie pam-lab1-<twoje_inicjały, albo index> (np. pam-lab1-ajk).
2	**Inicjuj Git**	W Android Studio: VCS → Enable Version Control Integration → Git.Lub w terminalu: cd /ścieżka/do/projektu && git init && git remote add origin <url>
3	**.gitignore**	Android Studio automatycznie generuje .gitignore dla projektu Gradle. Sprawdź czy zawiera: .gradle/, build/, *.apk, .idea/workspace.xml
4	**Pierwszy commit**	git add . && git commit -m 'feat: lab1 initial implementation' (Conventional Commits!)
5	**Push i link**	git push -u origin main. Dodaj prowadzącego jako collaboratora (Settings → Collaborators). Wklej URL repo na Moodle.

Instrukcja Kotlin 1: Konfiguracja środowiska deweloperskiego i pierwsza aplikacja Android
Programowanie Aplikacji Mobilnych | Katedra Informatyki i Automatyki
MobileHub
Następne ćwiczenie: Kotlin 2 - Nawigacja między ekranami, ViewModel i wzorzec MVVM