

**WYDZIAŁ TELEKOMUNIKACJI, INFORMATYKI
I ELEKTROTECHNIKI**

[Programowanie urządzeń mobilnych \[05-IST-PAM-SP5\]-laboratorium](#)

Lab (Nr_12) Temat: **Aplikacja Kotlin cz 4**



Cel:
Wersja ostateczna zgodna z założeniami.

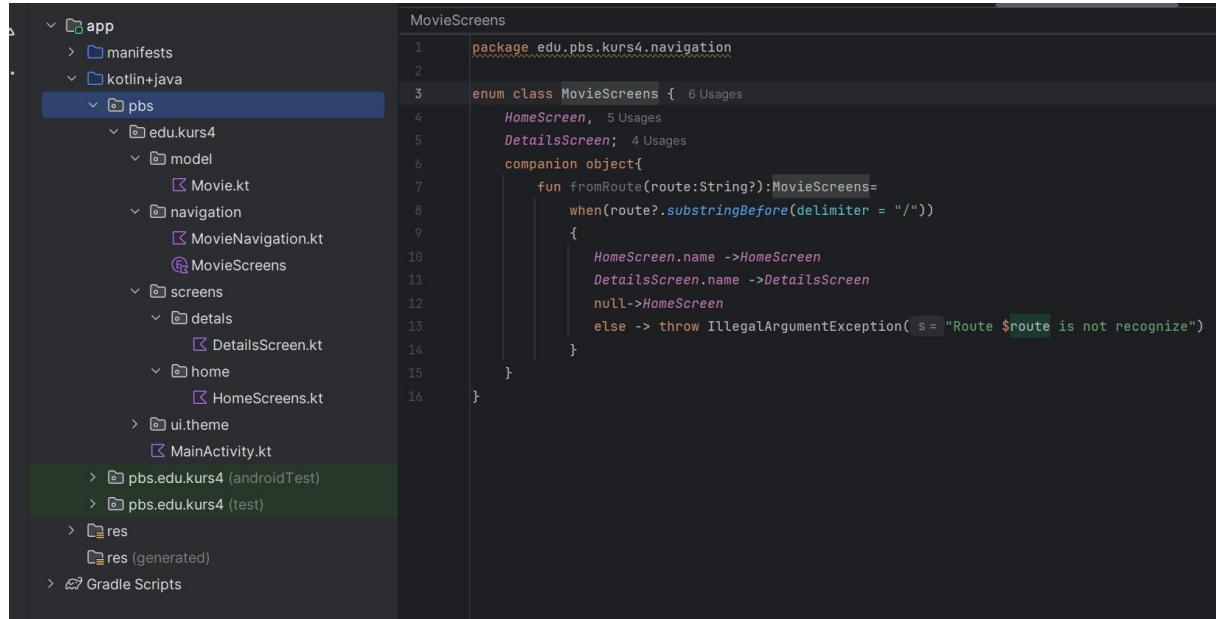
1. Struktura projektu

Aplikacja jest podzielona na logiczne moduły:

Kod

app/

model/	→ dane i modele (Movie, getMovies)
navigation/	→ nawigacja między ekranami
screens/	
home/	→ ekran główny (lista filmów)
details/	→ ekran szczegółów filmu
ui/theme/	→ Material3 (kolory, typografia)
MainActivity.kt	→ punkt startowy aplikacji



2.MainActivity – punkt startowy aplikacji

kotlin

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

Co robi ta funkcja?

- Jest to główna aktywność aplikacji – uruchamiana jako pierwsza.
- setContent { ... } uruchamia Jetpack Compose UI zamiast tradycyjnego XML.
- MyApp { MovieNavigation() } oznacza:
 - otocz UI motywem Material3 (Kurs4Theme)
 - uruchom system nawigacji (MovieNavigation())

3.MyApp – funkcja odpowiedzialna za motyw (theme)

kotlin

```
@Composable
fun MyApp(content: @Composable () -> Unit) {
    Kurs4Theme {
        content()
    }
}
```

```

        }
    }
Co robi?


- Nakłada na aplikację motyw Material3 zdefiniowany w Kurs4Theme.
- Przyjmuje dowolny Composable jako content.
- Dzięki temu wszystkie ekranы dziedziczą:
  - kolory,
  - typografię,
  - styl komponentów Material3.

```

MovieRow – najważniejszy komponent listy filmów
 To jest kluczowy element UI – pojedynczy wiersz reprezentujący film.

◆ 3.1. Sygnatura funkcji

```
kotlin
@Composable
fun MovieRow(
    movie: Movie = getMovies()[0],
    onClick: (String) -> Unit = {}
)
```

Parametry:

Parametr	Typ	Opis
movie	Movie	Dane filmu do wyświetlenia
onClick (String)	-> Unit	Callback wywoływany po kliknięciu

◆ 3.2. Stan lokalny – rozwijanie szczegółów
 kotlin

```
var expanded by remember { mutableStateOf(false) }
val density = LocalDensity.current
```

Co robi?

- expanded – czy sekcja szczegółów jest widoczna.
- remember – Compose zapamiętuje stan między recompozycjami.
- LocalDensity – potrzebne do animacji (dp → px).

◆ 3.3. Główna karta (Card)

```
kotlin
Card(
    modifier = Modifier
        .padding(4.dp)
        .fillMaxWidth()
        .height(180.dp)
        .clickable { onClick(movie.id.toString()) },
    shape = RoundedCornerShape(CornerSize(16.dp)),
    elevation = CardDefaults.cardElevation(defaultElevation = 6.dp)
)
```

Co robi?

- Tworzy kartę Material3 z cieniem i zaokrąglonymi rogami.
- Cała karta jest klikalna → przejście do ekranu szczegółów.
- Ustawia rozmiar i marginesy.

◆ 3.4. Układ poziomy (Row)

```
kotlin
Row(
    verticalAlignment = Alignment.CenterVertically,
    horizontalArrangement = Arrangement.Start
)
```

Zawiera:

1. miniaturę plakatu,
2. kolumnę z informacjami,
3. ikonę rozwijania.

◆ 3.5. Obraz filmu (Coil)

```
kotlin
val painter = rememberAsyncImagePainter(
    model = ImageRequest.Builder(LocalContext.current)
        .data(movie.images[0])
        .size(Size.ORIGINAL)
        .transformations(CircleCropTransformation())
        .crossfade(false)
        .build()
)
```

Co robi?

- Ładuje obraz z internetu.
- Stosuje transformację CircleCrop.
- Zapamiętuje painter, aby nie przeładowywać obrazu.

◆ 3.6. Podstawowe informacje o filmie

```
kotlin
Text(movie.title)
Text("Director: ${movie.director}")
Text("Released: ${movie.year}")
Wyświetla najważniejsze dane filmu.
```

◆ 3.7. Animowane rozwijanie szczegółów

```
kotlin
AnimatedVisibility(
    visible = expanded,
    enter = slideInVertically { with(density) { -40.dp.roundToPx() } }
        + expandVertically(expandFrom = Alignment.CenterVertically)
        + fadeIn(initialAlpha = 0.4f),
    exit = slideOutVertically() + shrinkVertically() + fadeOut()
)
```

Co robi?

- Pokazuje/ukrywa dodatkowe informacje z animacją.
- Animacje wejścia:
 - slideInVertically – wsuwa się z góry,
 - expandVertically – rozwija się,
 - fadeIn – pojawia się.
- Animacje wyjścia:
 - slideOutVertically – wysuwa się,
 - shrinkVertically – zwija się,
 - fadeOut – znika.

Co jest animowane?

- opis fabuły (plot)
- reżyser
- aktorzy

◆ 3.8. Ikona rozwijania

```
kotlin
Icon(
    imageVector = if (expanded) Icons.Filled.KeyboardArrowUp else
Icons.Filled.KeyboardArrowDown,
    modifier = Modifier
        .size(25.dp)
        .clickable { expanded = !expanded },
    tint = Color.LightGray
)
```

Co robi?

- Zmienia ikonę w zależności od stanu.
- Kliknięcie zmienia expanded.
- Wywołuje animację AnimatedVisibility.

4. DIAGRAM ARCHITEKTURY APLIKACJI FILMOWEJ (Jetpack Compose + Material3)

Poniższy diagram przedstawia:

- strukturę modułów,

- przepływ danych,
- zależności między ekranami,
- komponenty UI,
- nawigację.

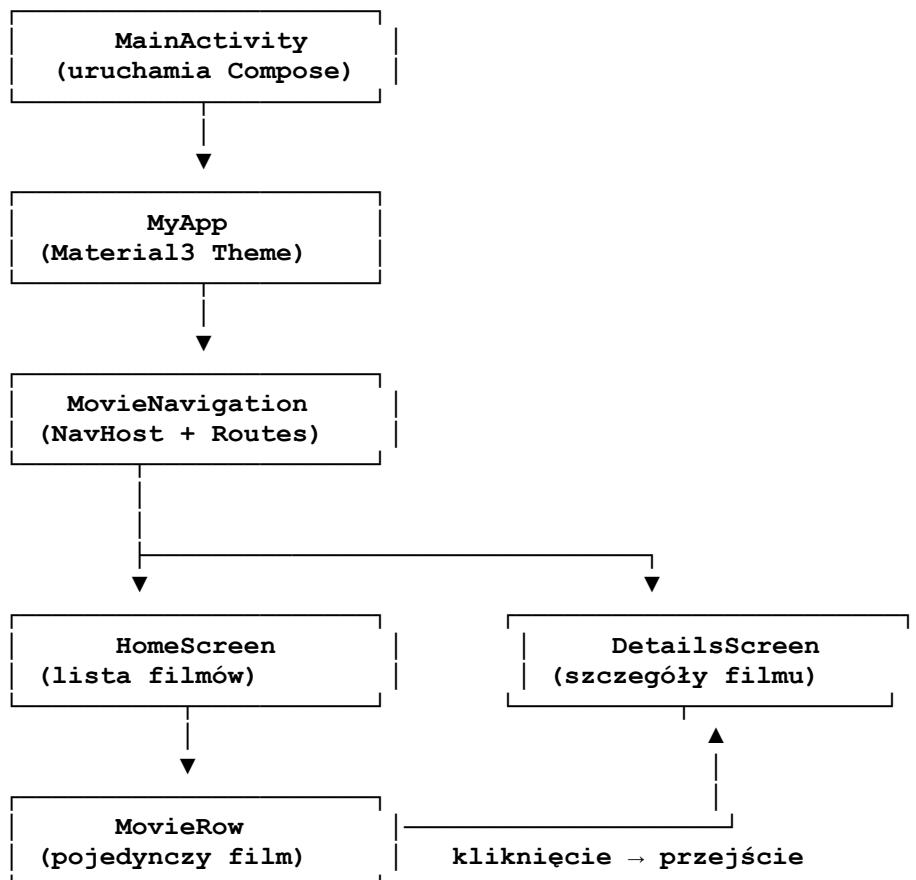
5. Struktura projektu (diagram folderów)

Kod

```
app/
  └── MainActivity.kt
  └── ui/
    └── theme/
      └── Color.kt
      └── Theme.kt
      └── Type.kt
  └── navigation/
    └── MovieNavigation.kt
    └── MovieScreens.kt
  └── screens/
    └── home/
      └── HomeScreen.kt
    └── details/
      └── DetailsScreen.kt
  └── components/
    └── MovieRow.kt
  └── model/
    └── Movie.kt
    └── getMovies.kt
```

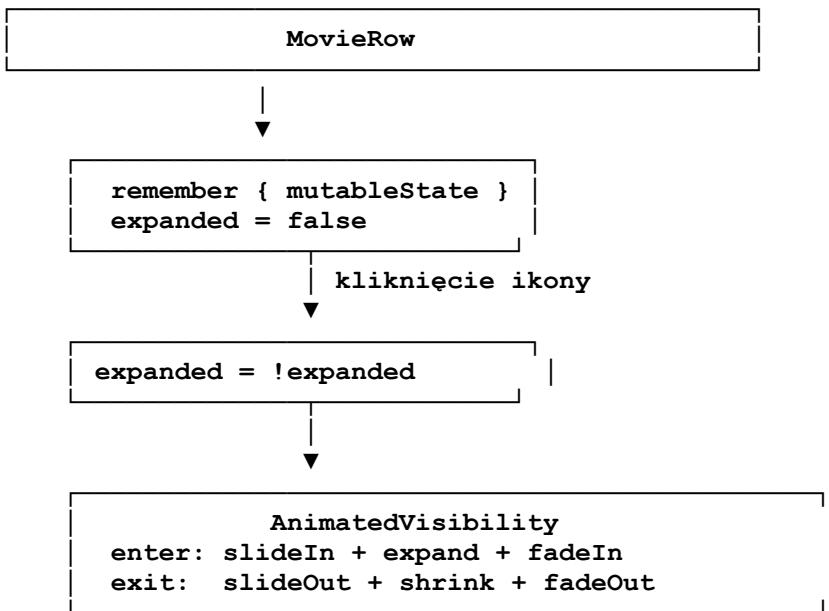
6. Przepływ aplikacji (diagram logiczny)

Kod



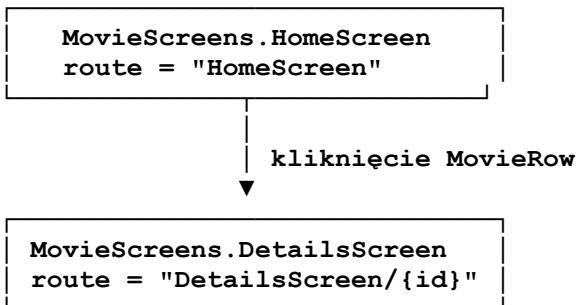
7. Diagram działania MovieRow (stan + animacje)

Kod



8. Diagram nawigacji (routing)

Kod



9. Diagram ładowania obrazu (Coil)

Kod

```

MovieRow
|
rememberAsyncImagePainter
|
ImageRequest.Builder
|
  data(url)
  size(Size.ORIGINAL)
  transformations(CircleCrop)
  crossfade(false)
|
Coil Engine
|
  cache (memory)
  cache (disk)
  network (jeśli brak w cache)
|
Image()
  
```

This diagram details the image loading process using Coil. It starts with a `MovieRow` component, which uses `rememberAsyncImagePainter`. This leads to an `ImageRequest.Builder` object, which is configured with `data(url)`, `size(Size.ORIGINAL)`, `transformations(CircleCrop)`, and `crossfade(false)`. Below this is the `Coil Engine`, which provides options for `cache (memory)`, `cache (disk)`, and `network (jeśli brak w cache)`. The final step is the creation of an `Image()` object.

10. Diagram przepływu danych (od modelu do UI)

Kod

```

getMovies()
  
```

The diagram shows a single step: `getMovies()`, representing the flow of data from the model layer to the UI layer.

```

List<Movie>
|
↓
HomeScreen
|
↓
LazyColumn
|
↓
MovieRow(movie)
|
↓
DetailsScreen(movieId)
|
↓
movie = getMovies().find { it.id == movieId }

```

```

73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120

```

```

    fun MovieRow(
        movie: Movie = getMovies()[0],
        onItemClick: (String) -> Unit = {}
    ) {
        var expanded by remember { mutableStateOf(false) }

        val density = LocalDensity.current
        Card(modifier = Modifier
            .padding(all = 4.dp)
            .fillMaxWidth()
            .height(height = 180.dp)
            .clickable { onItemClick(movie.id.toString()) },
            shape = RoundedCornerShape(size = 16.dp),
            elevation = CardDefaults.cardElevation(defaultElevation = 6.dp)
        ) {
            Row(
                verticalAlignment = Alignment.CenterVertically,
                horizontalArrangement = Arrangement.Start
            ) {
                RowScope {
                    Surface(
                        modifier = Modifier
                            .padding(all = 12.dp)
                            .size(size = 100.dp),
                        shape = RectangleShape,
                        tonalElevation = 0.dp
                    ) {
                        val painter = rememberAsyncImagePainter(
                            model = ImageRequest.Builder(context = LocalContext.current)
                                .data(data = movie.images[0])
                                .size(size.ORIGINAL)
                                .transformations(transformations = CircleCropTransformation())
                                .crossfade(enable = false)
                                .build()
                        )
                        Box(modifier = Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
                            Image(
                                painter = painter,
                                contentDescription = "Movie Poster"
                            )
                        }
                    }
                }
                Column(modifier = Modifier.padding(all = 4.dp)) {
                    Text(text = movie.title, style = MaterialTheme.typography.titleLarge)
                    Text(text = "Director: ${movie.director}", style = MaterialTheme.typography.bodySmall)
                    Text(text = "Released: ${movie.year}", style = MaterialTheme.typography.bodySmall)
                    AnimatedVisibility(
                        visible = expanded,
                        enter = slideInVertically { height = with(LocalDensity.current) { -40.dp.roundToPx() } }
                    )
                }
            }
        }
    }
}

```

Funkcja MovieRow cz1

```
kt kotlin+java
  pbs.edu.kurs4
    model
    navigation
    screens
      details
        DetailsScreen.kt
      home
    ui.theme
      Color.kt
      Theme.kt
      Type.kt
    MainActivity.kt
  pbs.edu.kurs4 (androidTest)
  pbs.edu.kurs4 (test)
res
  res (generated)
Gradle Scripts

kt { this RowScope
  Column(modifier = Modifier.padding(all = 4.dp)) {
    Text(text = movie.title, style = MaterialTheme.typography.titleLarge)
    Text(text = "Director: ${movie.director}", style = MaterialTheme.typography.bodySmall)
    Text(text = "Released: ${movie.year}", style = MaterialTheme.typography.bodySmall)
  }
  AnimatedVisibility(
    visible = expanded,
    enter = slideInVertically(),
    exit = slideOutVertically() + shrinkVertically()
  )
}
Column { this AnimatedVisibilityScope
  Column { this ColumnScope
    Text {
      text = buildAnnotatedString {
        AnnotatedStringBuilder {
          withStyle(SpanStyle(color = Color.DarkGray, fontSize = 13.sp)) {
            append("Plot:")
          }
          withStyle(SpanStyle(
            color = Color.Yellow,
            fontSize = 13.sp,
            fontWeight = FontWeight.ExtraBold
          )) {
            append(movie.plot)
          }
          withStyle(SpanStyle(
            color = Color.Blue,
            fontSize = 13.sp,
            fontWeight = FontWeight.Normal
          )) {
            append("\n Mobile kurs4")
          }
        }
      }
      modifier = Modifier.padding(all = 6.dp)
    }
    HorizontalDivider(modifier = Modifier.padding(all = 3.dp))
    Text {
      text = "Director: ${movie.director}",
      style = MaterialTheme.typography.bodySmall
    }
  }
}
```

Funkcja MovieRow cz2

Funkcja MovieRow cz3

11. Model danych

Dane są przechowywane lokalnie w formie listy obiektów Movie.

Model:

header:

data class Movie(

```
a class Movie{  
    val id: Int,  
    val title: String,  
    val year: String,  
    val runtime: String,  
    val genere: String,
```

```

    val director: String,
    val actors: String,
    val plot: String,
    val images: List<String>
)

```

Źródło danych:

```

kotlin
fun getMovies(): List<Movie>

```

To pozwala studentom skupić się na UI i nawigacji, bez konieczności używania API.

12. Nawigacja (Navigation Compose)

Nawigacja jest zdefiniowana w pliku:

Kod

```

MovieNavigation.kt

```

Wykorzystuje:

- NavHost – kontener na ekranы
- composable() – definicje ekranów
- przekazywanie argumentów (movieId)
- animacje przejść (slide)

The screenshot shows the Android Studio interface. On the left, the project structure is visible under the 'app' module. It includes 'manifests', 'kotlin/java' (containing 'pbs' and 'edu.kurs4' packages), 'navigation' (containing 'MovieNavigation.kt'), 'screens' (containing 'details' and 'home' packages), 'ui.theme' (containing 'MainActivity.kt'), and 'res'. The 'MovieNavigation.kt' file is selected and its code is displayed on the right. The code defines a composable function 'MovieNavigation' that creates a 'NavController' and a 'NavHost' with two routes: 'HomeScreen' and 'DetailsScreen'. The 'DetailsScreen' route is annotated with 'composable(route = "\${MovieScreens.DetailsScreen.name}/{movieId}")' and takes a 'movieData' argument from the navigation stack.

```

MovieNavigation() -> NavHost(...){}
1 package edu.pbs.kurs4.navigation
2 import androidx.compose.runtime.Composable
3 import androidx.compose.ui.tooling.preview.Preview
4 import androidx.navigation.compose.NavHost
5 import androidx.navigation.compose.composable
6 import androidx.navigation.compose.rememberNavController
7 import edu.pbs.kurs4.screens.details.DetailsScreen
8 import pbs.edu.kurs4.screens.home.HomeScreen
9
10 @Preview 3 Usages
11 @Composable
12 fun MovieNavigation() {
13     val navController = rememberNavController()
14     NavHost(
15         navController = navController,
16         startDestination = MovieScreens.HomeScreen.name
17     ) { this: NavGraphBuilder
18         composable(route = MovieScreens.HomeScreen.name) { this: AnimatedContentScope<NavBackStackEntry> ->
19             HomeScreen(navController = navController)
20         }
21         composable(route = "${MovieScreens.DetailsScreen.name}/{movieId}") { this: AnimatedContentScope<NavBackStackEntry> ->
22             val movieData = navBackStackEntry.arguments?.getString(key = "movieId")
23             DetailsScreen(navController = navController, movieId = movieData.toIntOrNull())
24         }
25     }
26 }

```

Funkcja MovieNavigation

13. HomeScreen – ekran główny

HomeScreen wyświetla:

- TopAppBar (Material3)
- BottomNavigationBar
- LazyColumn z listą filmów

Każdy element listy to MovieRow.

Kluczowe elementy:

- LazyColumn – wydajna lista
- MovieRow – pojedynczy element listy
- onItemClick – przejście do szczegółów

14. MovieRow – element listy filmów

MovieRow to komponent, który:

- wyświetla miniaturę plakatu (Coil)
- pokazuje podstawowe informacje o filmie
- ma ikonę rozwijania
- posiada animację rozwijania szczegółów (AnimatedVisibility)
- reaguje na kliknięcie i przechodzi do DetailsScreen

The screenshot shows the Android Studio interface. On the left, the project structure is displayed under the 'app' folder. It includes 'manifests', 'kotlin+java' (containing 'pbs.edu.kurs4' which has 'model', 'navigation', 'screens', 'details', and 'home' packages), 'ui.theme', 'MainActivity.kt', 'pbs.edu.kurs4 (androidTest)', 'pbs.edu.kurs4 (test)', 'res', and 'res (generated)'. Below these are 'Gradle Scripts' and 'Build Variants'. The 'HomeScreens.kt' file is selected and shown in the main code editor. The code is a Kotlin function-based composition for a 'HomeScreen' that sets up a top bar with a title and a bottom bar with a navigation item, and then displays a list of movies using a LazyColumn.

```

1 package pbs.edu.kurs4.screens.home
2
3 import androidx.compose.foundation.layout.*
4 import androidx.compose.foundation.lazy.LazyColumn
5 import androidx.compose.foundation.lazy.items
6 import androidx.compose.material.icons(Icons)
7 import androidx.compose.material.icons.filled.AccountBox
8 import androidx.compose.material3.*
9 import androidx.compose.runtime.Composable
10 import androidx.compose.ui.Modifier
11 import androidx.compose.ui.graphics.Color
12 import androidx.compose.ui.unit.dp
13 import androidx.navigation.NavController
14 import androidx.compose.material3.TopAppBarDefaults
15 import androidx.compose.material3.Text
16 import androidx.compose.material3.NavigationBar
17 import androidx.compose.material3.NavigationBarItem
18 import edu.pbs.kurs4.navigation.MovieScreens
19 import edu.pbs.kurs4.model.Movie
20 import edu.pbs.kurs4.model.getMovies
21 import pbs.edu.kurs4.MovieRow
22 @OptIn(markerClass = ExperimentalMaterial3Api::class) 2 Usages
23 @Composable
24
25 fun HomeScreen(navController: NavController) {
26     Scaffold(topBar = { TopAppBar(
27         title = { Text(text = "Movies TopAppBar Wykład") },
28         colors = TopAppBarDefaults.topAppBarColors(containerColor = Color.Magenta) },
29         bottomBar = { NavigationBar( containerColor = Color.Blue, tonalElevation = 9.dp )
30             { NavigationBarItem(selected = true, onClick = {}, icon = { Icon(imageVector = Icons.Default.AccountBox, contentDescription = null) },
31                 label = { Text(text = "Movies bottomBar Wykład") } ) } }
32         { innerPadding -> Column(modifier = Modifier.padding(paddingValues = innerPadding)) { MainContent(navController = navController) } } }
33     @Composable 1 Usage
34     fun MainContent(
35         navController: NavController,
36         // moviesList: List<String> = listOf("Avatar", "555", "Harry Potter", "Life", "Lolek", "Bolek", "Krecik", "Ida Swieta", "Wyklad")
37         moviesList: List<Movie> = getMovies()
38     ) {
39         LazyColumn(modifier = Modifier.padding(all = 12.dp)) { this.LazyListScope
40             items(items = moviesList) { this.LazyItemScope it ->
41
42                 MovieRow(movie = it){ movie ->navController.navigate(route = MovieScreens.DetailsScreen.name + "/$movie") }
43             }
44         }
45     }
46 }
47

```

Funkcja HomeScreen

15. DetailsScreen – ekran szczegółów

Ekran szczegółów:

- odbiera movieId z nawigacji
- wyszukuje film w liście
- wyświetla duży plakat
- pokazuje MovieRow (bez klikalności)
- prezentuje galerię obrazów w LazyRow

```

1 package edu.pbs.kurs4.screens.details
2 import android.util.Log
3 import androidx.compose.foundation.Image
4 import androidx.compose.foundation.layout.*
5 import androidx.compose.foundation.lazy.LazyRow
6 import androidx.compose.foundation.lazy.items
7 import androidx.compose.foundation.lazy.items
8 import androidx.compose.material.icons(Icons)
9 import androidx.compose.material.icons.automirrored.filled.ArrowBack
10 import androidx.compose.runtime.Composable
11 import androidx.compose.ui.Alignment
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.graphics.Color
14 import androidx.compose.ui.unit.dp
15 import androidx.navigation.NavController
16 import coil.compose.rememberAsyncImagePainter
17 import edu.pbs.kurs4.model.getMovies
18 import pbs.edu.kurs4.MovieRow
19
20 @OptIn(markerClass = ExperimentalMaterial3Api::class) 2 Usages
21 @Composable
22 fun DetailsScreen(navController: NavController, movieId: Int?) {
23     val movie = getMovies().firstOrNull { it.id == movieId }
24
25     Scaffold(
26         topBar = {
27             TopAppBar(
28                 title = { Text(text = "Movies") },
29                 navigationIcon = {
30                     IconButton(onClick = { navController.popBackStack() }) {
31                         Icon(
32                             imageVector = Icons.AutoMirrored.Filled.ArrowBack,
33                             contentDescription = "Back"
34                         )
35                     }
36                 },
37                 colors = TopAppBarDefaults.topAppBarColors(
38                     containerColor = Color.Transparent
39                 )
40             )
41         }
42     ) { innerPadding ->
43
44         Surface(
45             modifier = Modifier
46                 .padding(paddingValues = innerPadding)
47                 .fillMaxSize()
48         ) {
49             Column(
50                 horizontalAlignment = Alignment.CenterHorizontally,
51                 verticalArrangement = Arrangement.Top
52             ) {
53                 Log.d(tag = "DetailsScreen", msg = "movieId = $movieId")
54                 Log.d(tag = "DetailsScreen", msg = "movie = $movie")
55                 Log.d(tag = "DetailsScreen", msg = "Images count = ${movie?.images?.size ?: 0}")
56                 movie?.let { MovieRow(movie = it) }
57                 Spacer(modifier = Modifier.height(height = 8.dp))
58                 HorizontalDivider()
59                 Text(text = "movieId = $movieId")
60                 Text(text = "movie = ${movie?.title ?: "null"}")
61                 Text(text = "Images count = ${movie?.images?.size ?: 0}")
62                 Text(text = "Movie Images")
63                 movie?.let { it: Movie ->
64                     HorizontalScalableImageview(it.images)
65                 }
66             }
67         }
68     }
69 }
70 @Composable 1 Usage
71 private fun HorizontalScalableImageview(images: List<String>) {
72     LazyRow { this: LazyListScope
73         items(items = images) { this: LazyItemScope, image ->
74             Card(modifier = Modifier
75                 .padding(all = 12.dp)
76                 .size(size = 240.dp),
77                 elevation = CardDefaults.cardElevation(defaultElevation = 5.dp)
78             ) {
79                 this: ColumnScope
80                 val painter = rememberAsyncImagePainter(model = image)
81                 Image(
82                     painter = painter,
83                     contentDescription = "Movie Poster",
84                     modifier = Modifier.fillMaxSize()
85                 )
86             }
87         }
88     }
89 }

```

Funkcja DetailsScreen cz 1

```

1 package edu.pbs.kurs4.screens.details
2 import android.util.Log
3 import androidx.compose.foundation.Image
4 import androidx.compose.foundation.layout.*
5 import androidx.compose.foundation.lazy.LazyRow
6 import androidx.compose.foundation.lazy.items
7 import androidx.compose.foundation.lazy.items
8 import androidx.compose.material.icons(Icons)
9 import androidx.compose.material.icons.automirrored.filled.ArrowBack
10 import androidx.compose.runtime.Composable
11 import androidx.compose.ui.Alignment
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.graphics.Color
14 import androidx.compose.ui.unit.dp
15 import androidx.navigation.NavController
16 import coil.compose.rememberAsyncImagePainter
17 import edu.pbs.kurs4.model.getMovies
18 import pbs.edu.kurs4.MovieRow
19
20 @OptIn(markerClass = ExperimentalMaterial3Api::class) 2 Usages
21 @Composable
22 fun DetailsScreen(navController: NavController, movieId: Int?) {
23     val movie = getMovies().firstOrNull { it.id == movieId }
24
25     Scaffold(
26         topBar = {
27             TopAppBar(
28                 title = { Text(text = "Movies") },
29                 navigationIcon = {
30                     IconButton(onClick = { navController.popBackStack() }) {
31                         Icon(
32                             imageVector = Icons.AutoMirrored.Filled.ArrowBack,
33                             contentDescription = "Back"
34                         )
35                     }
36                 },
37                 colors = TopAppBarDefaults.topAppBarColors(
38                     containerColor = Color.Transparent
39                 )
40             )
41         }
42     ) { innerPadding ->
43
44         Surface(
45             modifier = Modifier
46                 .padding(paddingValues = innerPadding)
47                 .fillMaxSize()
48         ) {
49             Column(
50                 horizontalAlignment = Alignment.CenterHorizontally,
51                 verticalArrangement = Arrangement.Top
52             ) {
53                 Log.d(tag = "DetailsScreen", msg = "movieId = $movieId")
54                 Log.d(tag = "DetailsScreen", msg = "movie = $movie")
55                 Log.d(tag = "DetailsScreen", msg = "Images count = ${movie?.images?.size ?: 0}")
56                 movie?.let { MovieRow(movie = it) }
57                 Spacer(modifier = Modifier.height(height = 8.dp))
58                 HorizontalDivider()
59                 Text(text = "movieId = $movieId")
60                 Text(text = "movie = ${movie?.title ?: "null"}")
61                 Text(text = "Images count = ${movie?.images?.size ?: 0}")
62                 Text(text = "Movie Images")
63                 movie?.let { it: Movie ->
64                     HorizontalScalableImageview(it.images)
65                 }
66             }
67         }
68     }
69 }
70 @Composable 1 Usage
71 private fun HorizontalScalableImageview(images: List<String>) {
72     LazyRow { this: LazyListScope
73         items(items = images) { this: LazyItemScope, image ->
74             Card(modifier = Modifier
75                 .padding(all = 12.dp)
76                 .size(size = 240.dp),
77                 elevation = CardDefaults.cardElevation(defaultElevation = 5.dp)
78             ) {
79                 this: ColumnScope
80                 val painter = rememberAsyncImagePainter(model = image)
81                 Image(
82                     painter = painter,
83                     contentDescription = "Movie Poster",
84                     modifier = Modifier.fillMaxSize()
85                 )
86             }
87         }
88     }
89 }

```

Funkcja DetailsScreen cz 2

Plik gradle

OPIS PLIKU build.gradle.kts DLA APLIKACJI ANDROID (Compose + Material3)
Plik build.gradle.kts jest sercem konfiguracji projektu Android. Określa:

- jakie pluginy są używane,
- jak kompilować aplikację,
- jakie biblioteki są potrzebne,
- jakie wersje SDK obowiązują,
- jak działa Compose.

Poniżej znajdziesz szczegółowe omówienie każdej sekcji.

✖ 16. Sekcja plugins

```
kotlin
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose)
}
```

Co robi?

- android.application – plugin Androida, który pozwala budować aplikację (APK).
- kotlin.android – dodaje wsparcie dla języka Kotlin.
- kotlin.compose – włącza wsparcie dla Jetpack Compose.

Dlaczego alias?

Projekt używa Version Catalog (libs.versions.toml), który:

- centralizuje wersje pluginów i bibliotek,
- ułatwia aktualizacje,
- eliminuje duplikaty.

17. Sekcja android { ... }

To główna konfiguracja aplikacji.

◆ 17.1. Namespace

```
kotlin
namespace = "pbs.edu.kurs4"
To unikalna przestrzeń nazw aplikacji – odpowiednik pakietu.
```

◆ 17.2. compileSdk

```
kotlin
compileSdk {
    version = release(36)
}


- Określa na jakiej wersji Android SDK komplujemy projekt.
- Tu: Android 14 (API 36).

```

◆ 17.3. defaultConfig

```
kotlin
defaultConfig {
    applicationId = "pbs.edu.kurs4"
    minSdk = 31
    targetSdk = 36
    versionCode = 1
    versionName = "1.0"
    testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
}
```

Co oznacza?

Pole	Znaczenie
applicationId	unikalny identyfikator aplikacji
minSdk = 31	minimalna wersja Androida → Android 12
targetSdk = 36	aplikacja zoptymalizowana pod Android 14
versionCode	numer wersji dla Google Play
versionName	wersja widoczna dla użytkownika

Pole	Znaczenie
<code>testInstrumentationRunner runner do testów UI</code>	
◆ 17.4. <code>buildTypes</code>	
<code>kotlin</code>	
<code>buildTypes {</code>	
<code>release {</code>	
<code>isMinifyEnabled = false</code>	
<code>proguardFiles(...)</code>	
<code>}</code>	
<code>}</code>	
Co robi?	
• Definiuje konfigurację dla wersji release.	
• <code>isMinifyEnabled = false</code> – wyłączone zaciemnianie kodu.	
• <code>proguardFiles</code> – pliki reguł ProGuard/R8.	
◆ 17.5. <code>compileOptions</code>	
<code>kotlin</code>	
<code>compileOptions {</code>	
<code>sourceCompatibility = JavaVersion.VERSION_11</code>	
<code>targetCompatibility = JavaVersion.VERSION_11</code>	
<code>}</code>	
Aplikacja używa Java 11 jako wersji języka.	
◆ 17.6. <code>kotlinOptions</code>	
<code>kotlin</code>	
<code>kotlinOptions {</code>	
<code>jvmTarget = "11"</code>	
<code>}</code>	
Kompilator Kotlin generuje kod zgodny z JVM 11.	
◆ 17.7. <code>buildFeatures</code>	
<code>kotlin</code>	
<code>buildFeatures {</code>	
<code>compose = true</code>	
<code>}</code>	
Włącza Jetpack Compose.	
Bez tego Compose nie działa.	
18. Sekcja dependencies	
To lista bibliotek używanych w projekcie.	
◆ 18.1. Podstawowe biblioteki Androida	
<code>kotlin</code>	
<code>implementation("androidx.core:core-ktx:1.12.0")</code>	
<code>implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.6.2")</code>	
<code>implementation("androidx.activity:activity-compose:1.8.0")</code>	
• <code>core-ktx</code> – rozszerzenia Kotlin dla Androida.	
• <code>lifecycle-runtime-ktx</code> – cykl życia + coroutines.	
• <code>activity-compose</code> – integracja Activity z Compose.	
◆ 18.2. Compose BOM	
<code>kotlin</code>	
<code>implementation(platform("androidx.compose:compose-bom:2023.10.01"))</code>	
Co to jest BOM?	
Bill of Materials – zarządza wersjami wszystkich modułów Compose.	
Dzięki temu możesz pisać:	
<code>kotlin</code>	
<code>implementation("androidx.compose.ui:ui")</code>	
bez podawania wersji.	
◆ 18.3. Moduły Compose	
<code>kotlin</code>	
<code>implementation("androidx.compose.ui:ui")</code>	
<code>implementation("androidx.compose.ui:ui-graphic")</code>	
<code>implementation("androidx.compose.ui:ui-tooling-preview")</code>	

```
implementation("androidx.compose.material3:material3:1.2.0")
    • ui - podstawowe komponenty Compose.
    • ui-graphics - grafika, kolory, rysowanie.
    • ui-tooling-preview - podgląd w Android Studio.
    • material3 - komponenty Material Design 3.
    ◆ 18.4. Nawigacja Compose
kotlin
implementation("androidx.navigation:navigation-compose:2.7.5")
Umożliwia:
    • NavHost
    • composable()
    • przekazywanie argumentów
    ◆ 18.5. Coil - ładowanie obrazów
kotlin
implementation(libs.coil.compose)
Ładuje obrazy z internetu w Compose.
    ◆ 18.6. Testy
kotlin
testImplementation("junit:junit:4.13.2")
androidTestImplementation("androidx.test.ext:junit:1.1.5")
androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
androidTestImplementation("androidx.compose.ui:ui-test-junit4")
debugImplementation("androidx.compose.ui:ui-tooling")
debugImplementation("androidx.compose.ui:ui-test-manifest")
    • testy jednostkowe (JUnit)
    • testy UI (Espresso + Compose Test)
    • narzędzia debugowania Compose
```

app

```

1  plugins {
2     alias(libs.plugins.android.application)
3     alias(libs.plugins.kotlin.android)
4     alias(libs.plugins.kotlin.compose)
5   }
6
7   android {
8     namespace = "pbs.edu.kurs4"
9     compileSdk { this: CompileSdkSpec
10        version = release( version = 36 )
11    }
12
13    defaultConfig { this: ApplicationDefaultConfig
14        applicationId = "pbs.edu.kurs4"
15        minSdk = 31
16        targetSdk = 36
17        versionCode = 1
18        versionName = "1.0"
19        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
20    }
21
22    buildTypes { this: NamedDomainObjectContainer<ApplicationBuildType>
23        release { this: ApplicationBuildType
24            isMinifyEnabled = false
25            proguardFiles(
26                ...files = getDefaultProguardFile( name = "proguard-android-optimize.txt"),
27                "proguard-rules.pro"
28            )
29        }
30    }
31
32    compileOptions { this: CompileOptions
33        sourceCompatibility = JavaVersion.VERSION_11
34        targetCompatibility = JavaVersion.VERSION_11
35    }
36
37    kotlinOptions {
38        jvmTarget = "11"
39    }
40
41    buildFeatures { this: ApplicationBuildFeatures
42        compose = true
43    }
44
45    dependencies {
46        implementation("androidx.core:core-ktx:1.12.0")
47        implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.6.2")
48        implementation("com.google.android.material:material:1.8.0")
49
50        implementation(platform( notation = "androidx.compose:compose-bom:2023.10.01"))
51        implementation("androidx.compose.ui:ui")
52    }
53
54    proguardRules.pro (ProGuard Rules for ":app")
55    gradle.properties (Project Properties)
56    gradle-wrapper.properties (Gradle Version)
57    libs.versions.toml (Version Catalog "libs")
58    local.properties (SDK Location)
59
60

```

Plik.gradle cz1

app

```

1  android {
2     buildTypes { this: NamedDomainObjectContainer<ApplicationBuildType>
3         release { this: ApplicationBuildType
4             isMinifyEnabled = true
5             proguardFiles(
6                 ...files = getDefaultProguardFile( name = "proguard-android-optimize.txt"),
7                 "proguard-rules.pro"
8             )
9         }
10    }
11
12    compileOptions { this: CompileOptions
13        sourceCompatibility = JavaVersion.VERSION_11
14        targetCompatibility = JavaVersion.VERSION_11
15    }
16
17    kotlinOptions {
18        jvmTarget = "11"
19    }
20
21    buildFeatures { this: ApplicationBuildFeatures
22        compose = true
23    }
24
25    dependencies {
26        implementation("androidx.core:core-ktx:1.12.0")
27        implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.6.2")
28        implementation("com.google.android.material:material:1.8.0")
29
30        implementation(platform( notation = "androidx.compose:compose-bom:2023.10.01"))
31        implementation("androidx.compose.ui:ui")
32        implementation("androidx.compose.ui:ui-graphics")
33        implementation("androidx.compose.ui:ui-tooling-preview")
34        implementation("androidx.compose.material3:material3:1.2.0")
35        implementation("androidx.navigation:navigation-compose:2.7.5")
36
37        testImplementation("junit:junit:4.13.2")
38        androidTestImplementation("androidx.test.ext:junit:1.1.5")
39        androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
40        androidTestImplementation("androidx.compose.ui:ui-test-junit4")
41        debugImplementation("androidx.compose.ui:ui-tooling")
42        debugImplementation("androidx.compose.ui:ui-test-manifest")
43
44        implementation(libs.coil.compose)
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

```

Plik gradle cz2