



# PROGRAMOWANIE URZĄDZEŃ MOBILNYCH 1

## WYKŁAD 11

- Nawigacja w aplikacji
- Compose Navigation

Centralny komponent zarządzający **stanem nawigacji**, śledzący **stos ekranów** (back stack) i aktualną pozycję na grafie.

```
val navController = rememberNavController()

NavHost(
    navController = navController,
    startDestination = "screenA"

    composable(route = "screenA"){
        ScreenA(navController)
    }

    composable(route = "screenB") {
        ScreenB(navController)
    }
}
```

Centralny komponent zarządzający **stanem nawigacji**, śledzący **stos ekranów** (back stack) i aktualną pozycję na grafie.

Kontener wyświetlający aktualną pozycję na grafie. Łączy **NavController** z grafem nawigacyjnym (**NavGraph**).

```
val navController = rememberNavController()

NavHost(
    navController = navController,
    startDestination = "screenA"

    composable(route = "screenA"){
        ScreenA(navController)
    }

    composable(route = "screenB") {
        ScreenB(navController)
    }
}
```

Centralny komponent zarządzający **stanem nawigacji**, śledzący **stos ekranów** (back stack) i aktualną pozycję na grafie.

Kontener wyświetlający aktualną pozycję na grafie. Łączy **NavController** z grafem nawigacyjnym (**NavGraph**).

```
val navController = rememberNavController()

NavHost(
    navController = navController,
    startDestination = "screenA"

    composable(route = "screenA"){
        ScreenA(navController)
    }

    composable(route = "screenB") {
        ScreenB(navController)
    }
}
```

Każdy ekran jest identyfikowany przez route (np. "home", "profile/{userId}").

# Nawigacja – Podstawowe Elementy

Centralny komponent zarządzający stanem nawigacji, śledzący stos ekranów (back stack) i aktualną pozycję na grafie.

Kontener wyświetlający aktualną pozycję na grafie. Łączy **NavController** z grafem nawigacyjnym (**NavGraph**).

Graf definiowany wewnątrz **NavHost**, składający się z celu i powiązań między nimi.

```
val navController = rememberNavController()

NavHost(
    navController = navController,
    startDestination = "screenA"
) {
    composable(route = "screenA"){
        ScreenA(navController)
    }
    composable(route = "screenB") {
        ScreenB(navController)
    }
}
```

Każdy ekran jest identyfikowany przez route (np. "home", "profile/{userId}").

Przekazywanie danych przez **route** (argumenty w URL)

```
@Composable
```

```
fun HomeScreen(navController: NavController) {
```

```
    Column(
```

```
        modifier = Modifier.fillMaxSize(),
```

```
        verticalArrangement = Arrangement.Center,
```

```
        horizontalAlignment = Alignment.CenterHorizontally
```

```
    ) {
```

```
        Button(onClick = {
```

```
            navController.navigate(route: "profile/123/Kamil")
```

```
        }) {
```

```
            Text(text: "Przejdź do profilu")
```

```
        }
```

```
    }
```

```
}
```

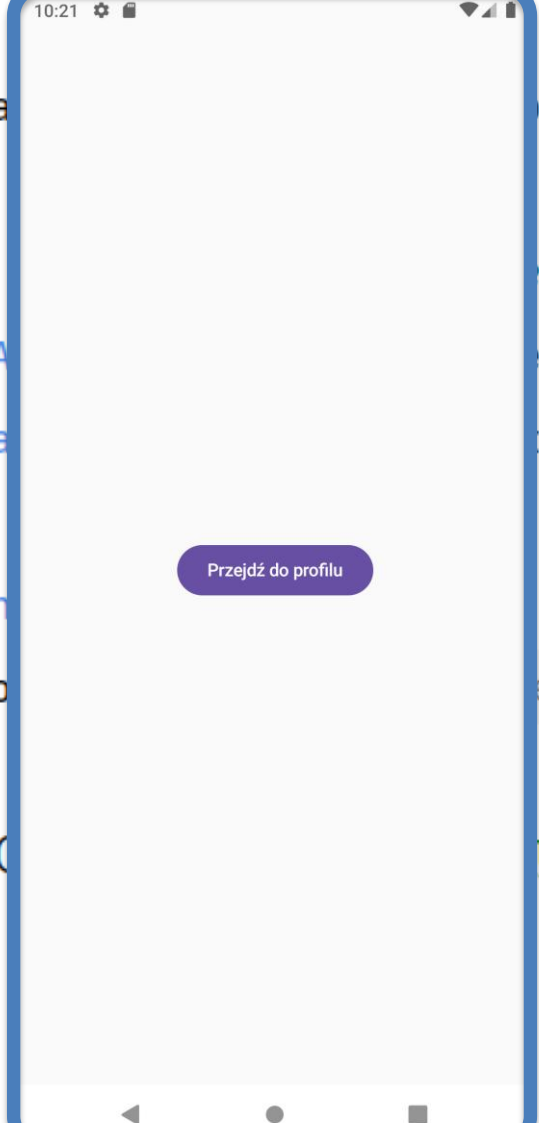
Nazwa ekranu

dane

dane

Przekazywanie danych przez route (argumenty w URL)

```
@Composable
fun HomeScreen(navController: NavController) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalAlignment = Alignment.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Button(
            onClick = {
                navController.navigate("profile/123/Kamil")
            }
        ) {
            Text("Przejdź do profilu")
        }
    }
}
```



Przekazywanie danych przez **route** (argumenty w URL)

Funkcja przyjmuje dane jako parametry

```
@Composable
fun ProfileScreen(userId: String, UserName: String) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text: "ID użytkownika: $userId")
        Text(text: "Nazwa użytkownika: $userName")
    }
}
```



Przekazywanie danych przez URL

```
@Composable
```

```
fun ProfileScreen
```

```
Column(
```

```
    modifier =
```

```
    verticalAlignment =
```

```
    horizontalAlignment =
```

```
) {
```

```
    Text(text =
```

```
    Text(text =
```

```
}
```

```
}
```

ID użytkownika: 123  
Nazwa użytkownika: Kamil

```
userName: String) {
```

```
    .size(100.dp),
```

```
    alignment = Alignment.Center,
```

```
    horizontalAlignment = Alignment.CenterHorizontally
```

```
    text = "$userId"
```

```
    text = "$userName"
```

## Przekazywanie danych przez **route** (argumenty w URL)

@Composable

```
fun NavigationApp() {
```

```
    val navController = rememberNavController()
```

```
    NavHost(navController = navController, startDestination = "home") {
```

```
        composable(route: "home") {
```

```
            HomeScreen(navController)
```

```
        }
```

```
        composable(route: "profile/{userId}/{userName}") { backStackEntry ->
```

```
            val userId = backStackEntry.arguments?.getString(key: "userId") ?: ""
```

```
            val userName = backStackEntry.arguments?.getString(key: "userName") ?: ""
```

```
            ProfileScreen(userId = userId, userName = userName)
```

```
        }
```

```
    }
```

```
}
```

Nazwa ekranu

Parametr 1

Parametr 2

## Przekazywanie danych przez **query parameters**

```
@Composable
```

```
fun HomeScreen(navController: NavController) {
```

```
    Column(
```

```
        modifier = Modifier.fillMaxSize(),
```

```
        verticalArrangement = Arrangement.Center,
```

```
        horizontalAlignment = Alignment.CenterHorizontally
```

```
    ) {
```

```
        Button(onClick = {
```

```
            navController.navigate(route: "search?query=Jetpack Compose&sort=desc")
```

```
        }) {
```

```
            Text(text: "Wyszukaj")
```

```
        }
```

```
    }
```

```
}
```

Nazwa ekranu

dane

dane

## Przekazywanie danych przez **query parameters**

@Composable

```
fun SearchScreen(query: String, sort: String) {  
    Column(  
        modifier = Modifier.fillMaxSize(),  
        verticalArrangement = Arrangement.Center,  
        horizontalAlignment = Alignment.CenterHorizontally  
    ) {  
        Text(text = "Wyniki wyszukiwania dla: '$query'", fontSize = 16.sp)  
        Text(text = "Sortowanie: $sort", fontSize = 24.sp)  
    }  
}
```

## Przekazywanie danych przez **query parameters**

```
NavHost(navController = navController, startDestination = "home") {  
    composable(route: "home") {  
        HomeScreen(navController)  
    }  
    composable(  
        route: "search?query={query}&sort={sort}",  
        arguments = listOf(  
            navArgument(name: "query") {  
                type = NavType.StringType  
                defaultValue = ""  
            },  
            navArgument(name: "sort") {  
                type = NavType.StringType  
                defaultValue = "asc"  
            }  
        )  
    ) { backStackEntry ->  
        val query = backStackEntry.arguments?.getString(key: "query") ?: ""  
        val sort = backStackEntry.arguments?.getString(key: "sort") ?: "asc"  
        SearchScreen(query = query, sort = sort)  
    }  
}
```

Nazwa ekranu

argumenty

Rozpakowanie  
argumentów

## Przekazywanie złożonych danych (obiektów) przez **SavedStateHandle**

```
data class User(  
    val id: String,  
    val name: String,  
    val email: String  
)
```

```
@Composable  
fun HomeScreen(navController: NavController) {  
    Column(  
        modifier = Modifier.fillMaxSize(),  
        verticalArrangement = Arrangement.Center,  
        horizontalAlignment = Alignment.CenterHorizontally  
    ) {  
        Button(onClick = {  
            val user = User(  
                id = "123",  
                name = "Jan Kowalski",  
                email = "jan@example.com"  
            )  
            val userJson = Uri.encode(Gson().toJson(user))  
            navController.navigate(route: "details/$userJson")  
        }) {  
            Text(text: "Przejdź do szczegółów")  
        }  
    }  
}
```

Konwerter



Przekazywanie złożonych danych (obiektów) przez **SavedStateHandle**

```
@Composable
✓ fun DetailsScreen(user: User) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text: "Szczegóły użytkownika")
        Spacer(modifier = Modifier.height(16.dp))
        Text(text: "ID: ${user.id}")
        Text(text: "Imię: ${user.name}")
        Text(text: "Email: ${user.email}")
    }
}

data class User(
    val id: String,
    val name: String,
    val email: String
)
```

Przekazywanie złożonych danych (obiektów) przez **SavedStateHandle**

@Composable

```
fun NavigationApp() {
```

```
    val navController = rememberNavController()
```

```
    NavHost(navController = navController, startDestination = "home") {
```

```
        composable(route: "home") {
```

```
            HomeScreen(navController)
```

```
        }
```

```
        composable(route: "details/{userJson}") { backStackEntry ->
```

```
            val userJson = backStackEntry.arguments?.getString(key: "userJson") ?: ""
```

```
            val user = try {
```

```
                Gson().fromJson(userJson, User::class.java)
```

```
            } catch (e: Exception) {
```

```
                User(id: "", name: "", email: "")
```

```
            }
```

```
            DetailsScreen(user = user)
```

```
    }
```

```
data class User(
```

```
    val id: String,
```

```
    val name: String,
```

```
    val email: String
```

```
)
```



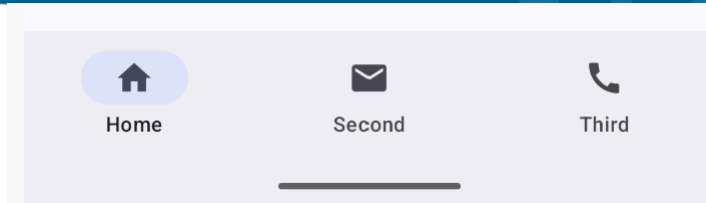
```
✓ sealed class Screens(val route: String) {  
    data object MainScreen : Screens(route: "main_screen")  
    data object SecondScreen : Screens(route: "second_screen")  
}
```

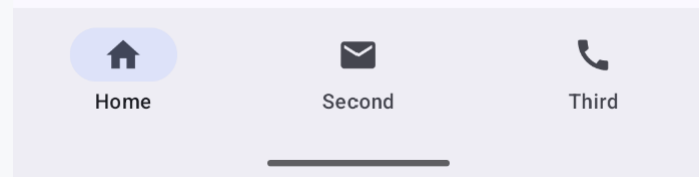
```
@Composable
fun MainScreen(onSecondScreen: () -> Unit) {
    Column (
        Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ){
        Text( text: "Home Screen")
        Spacer(modifier = Modifier.height(8.dp))
        Button(onClick = onSecondScreen) {
            Text( text: "Go to Second Screen")
        }
    }
}

@Composable
fun SecondScreen(onHome: () -> Unit) {
    Column (
        Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ){
        Text( text: "Home Screen")
        Spacer(modifier = Modifier.height(8.dp))
        Button(onClick = onHome) {
            Text( text: "Go back to Home Screen")
        }
    }
}
```

```
✓ sealed class Screens(val route: String) {  
    data object MainScreen : Screens( route: "main_screen")  
    data object SecondScreen : Screens( route: "second_screen")  
}
```

```
@Composable  
✓ fun Navigation() {  
    val navController = rememberNavController()  
    ✓ NavHost(navController = navController, startDestination = Screens.MainScreen.route) {  
    ✓     composable(Screens.MainScreen.route) {  
        MainScreen(onSecondScreen = { navController.navigate(Screens.SecondScreen.route) })  
    }  
    ✓     composable(Screens.SecondScreen.route) {  
        SecondScreen(onHome = { navController.navigate(Screens.MainScreen.route) })  
    }  
    }  
}
```

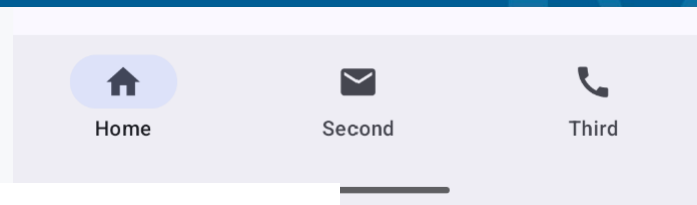




```
sealed class Screens(val route: String) {  
    data object MainScreen : Screens(route: "main_screen")  
    data object SecondScreen : Screens(route: "second_screen")  
    data object ThirdScreen : Screens(route: "third_screen")  
}  
  
sealed class BottomBarScreens(  
    val route: String,  
    val title: String,  
    val icon: ImageVector  
) {  
    data object Home : BottomBarScreens(Screens.MainScreen.route,  
        title: "Home", Icons.Default.Home)  
    data object Second : BottomBarScreens(Screens.SecondScreen.route,  
        title: "Second", Icons.Default.Email)  
    data object Third : BottomBarScreens(Screens.ThirdScreen.route,  
        title: "Third", Icons.Default.Call)  
}
```

Ekrany aplikacji

Ekrany na które  
można przejść  
z dolnej belki

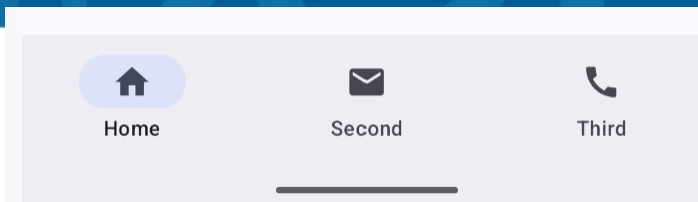


```
@Composable
fun MainScreen() {
    Column(
        Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text( text: "Home Screen")
    }
}
```

```
@Composable
fun SecondScreen() {
    Column(
        Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizont
        verticalArrangement = Arrangement.Center
    ) {
        Text( text: "Second Screen")
        Spacer(modifier = Modifier.height(8.dp))
    }
}
```

```
@Composable
fun ThirdScreen() {
    Column(
        Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text( text: "Third Screen")
    }
}
```

# Nawigacja – Bottom Navigation



@Composable

```
fun BottomNavGraph(navHostController: NavHostController) {  
    NavHost(navController = navHostController, startDestination = Screens.MainScreen.route) {  
        composable(Screens.MainScreen.route) { MainScreen() }  
        composable(Screens.SecondScreen.route) { SecondScreen() }  
        composable(Screens.ThirdScreen.route) { ThirdScreen() }  
    }  
}
```

Lista ekranów

Graf nawigacji  
powiązany z  
dolną belką

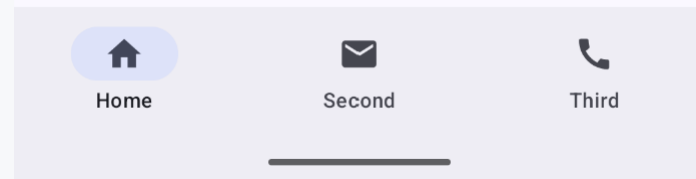
@Composable

```
fun BottomNavigationBar(navController: NavHostController) {  
    val screens = listOf(BottomBarScreens.Home, BottomBarScreens.Second, BottomBarScreens.Third)  
    val navBackStackEntry by navController.currentBackStackEntryAsState()  
    val currentDestination = navBackStackEntry?.destination  
  
    NavigationBar {  
        screens.forEach { screen ->  
            NavigationBarItem(  
                label = { Text(text = screen.title) },  
                selected = currentDestination?.hierarchy?.any { it.route == screen.route } == true,  
                onClick = { navController.navigate(screen.route) },  
                icon = { Icon(screen.icon, contentDescription = screen.title) }  
            )  
        }  
    }  
}
```

Pobiera aktualną  
pozycję nawigacji

Pobiera aktualny wpis na  
stosie nawigacji (back  
stack) jako stan kompozycji

określa, czy dana zakładka w  
pasku nawigacji powinna być  
oznaczona jako wybrana



```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@Composable
fun Navigation() {
    val navController = rememberNavController()
    Scaffold(
        bottomBar = {
            BottomNavigationBar(navController)
        }
    ) { BottomNavGraph(navController) }
}
```