LAB 7

MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Điều hướng giữa các màn hình với Navigation
- ✓ Quản lý trạng thái giao diện với State
- ✓ MVVM, ViewModel

NỘI DUNG

BÀI 1: XÂY DỰNG CHỨC NĂNG LẤY DANH SÁCH PHIM SỬ DỤNG VIEWMODEL

Hướng dẫn:

Sử dụng lại giao diện hiển thị danh sách phim ở bài 02 Lab 06

Bước 1: Import thư viện

```
implementation("androidx.compose.runtime:runtime-livedata:1.6.0")
```

Bước 2: Tạo class **MainViewModel** kế thừa **ViewModel()**

```
class MainViewModel : ViewModel() {
    private val _movies = MutableLiveData<List<Movie>>()
    val movies: LiveData<List<Movie>> = _movies

    init {
        _movies.value = Movie.getSampleMovies()
    }
}
```

Bước 3: Trong MainActivity trong **setContent** khai báo **MainViewModel** vừa tạo và tạo state đặt tên là **moviesState**

```
val mainViewModel: MainViewModel = viewModel()
val moviesState =
        mainViewModel.movies.observeAsState(initial = emptyList())
MovieScreen(moviesState.value)
```

BÀI 2: THỰC HIỆN CHUYỂN MÀN HÌNH GIỮA CÁC SCREEN SỬ DỤNG NAVIGATION

Hướng dẫn

Bước 1: Tạo file **Screen** khai báo và quản lý route các màn hình có trong ứng dụng

```
enum class Screen(val route: String) {
   SCREEN1("Screen1"),
   SCREEN2("Screen2"),
   SCREEN3("Screen3"),
}
```

Bước 2: Tạo file **ScreenNavigation** cấu hình và quản lý các screen

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

   NavHost(
      navController = navController,
      startDestination = Screen.SCREEN1.route,
   ) {
      composable(Screen.SCREEN1.route) { Screen1(navController) }
      composable(Screen.SCREEN2.route) { Screen2(navController) }
      composable(Screen.SCREEN3.route) { Screen3(navController) }

   }
}
```

Tạo một số màn hình mẫu để di chuyển qua lại

```
@Composable
fun Screen1(navController: NavController) {
    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(Color.Red)
            .clickable { navController.navigate(Screen.SCREEN2.route) },
        contentAlignment = Alignment.Center
    ) {
        Text("Screen 1", color = Color.White, fontSize = 20.sp)
    }
}

@Composable
fun Screen2(navController: NavController) {
    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(Color.Yellow)
            .clickable { navController.navigate(Screen.SCREEN3.route) },
        contentAlignment = Alignment.Center
    ) {
        Text("Screen 2", color = Color.White, fontSize = 20.sp)
    }
}

@Composable
fun Screen3(navController: NavController) {
    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(Color.Green)
            .clickable { navController.navigate(Screen.SCREEN1.route) },
        contentAlignment = Alignment.Center
    ) {
        Text("Screen 3", color = Color.White, fontSize = 20.sp)
    }
}
```

Chạy thử và ta được kết quả có thể chuyển đổi giữa các screen với nhau

Trong trường hợp nếu muốn chuyển hướng qua một màn hình và clear màn hình đã chuyển hướng trước đó ta thêm thuộc tính popUpTo.

Ví dụ: Từ màn hình 1, chuyển hướng sang màn hình 2, sau khi chuyển hướng qua màn hình 2, clear màn hình 1 không có người dùng quay lại

```
@Composable
fun Screen1(navController: NavController) {
    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(Color.Red)
            .clickable {
                navController.navigate(Screen.SCREEN2.route) {
                    popUpTo(Screen.SCREEN1.route) { inclusive = true }
                }
            },
        contentAlignment = Alignment.Center
    ) {
        Text("Screen 1", color = Color.White, fontSize = 20.sp)
    }
}
```

Bước 3: Thay đổi trong MainActivity

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ScreenNavigation()
        }
    }
}
```

BÀI 3: THỰC HIỆN CHỨC NĂNG ĐĂNG NHẬP CHO ỨNG DỤNG

Bước 1: Tạo class LoginViewModel kế thừa ViewModel

```
class LoginViewModel : ViewModel() {

    private val _username = MutableLiveData<String>()
    val username: LiveData<String> = _username

    private val _password = MutableLiveData<String>()
    val password: LiveData<String> = _password

    private val _rememberMe = MutableLiveData<Boolean>()
    val rememberMe: LiveData<Boolean> = _rememberMe

    private val _isAuthenticated = MutableLiveData<Boolean?>()
    val isAuthenticated: LiveData<Boolean?> = _isAuthenticated


    fun login(username: String, password: String, rememberMe: Boolean) {
        if (username.equals("admin") && password.equals("123")) {
            _isAuthenticated.value = true
        } else {
            _isAuthenticated.value = false
        }

    }

    fun resetAuthenticationState() {
        _isAuthenticated.value = null
    }
}
```

Bước 2: Xây dựng màn hình LoginScreen

- Sử dụng lại layout Lab05

```
@Composable
fun LoginScreen(navController: NavController) {
    val loginViewModel: LoginViewModel = viewModel()
    LoginCard(navController, loginViewModel)
}

@Composable
fun LoginCard(navController: NavController, loginViewModel: LoginViewModel) {
    val snackbarHostState = remember { SnackbarHostState() }
    HandleLoginState(snackbarHostState, loginViewModel, navController)

    Scaffold(
        snackbarHost = { SnackbarHost(snackbarHostState) }
    ) { paddingValues ->
        LoginForm(loginViewModel, paddingValues)
    }
}

@Composable
fun HandleLoginState(
    snackbarHostState: SnackbarHostState,
    loginViewModel: LoginViewModel,
    navController: NavController
) {
    val isAuthenticated by loginViewModel.isAuthenticated.observeAsState()

    LaunchedEffect(key1 = isAuthenticated) {
        when (isAuthenticated) {
            true -> {
                navController.navigate(Screen.MOVIE_SCREEN.route) {
                    popUpTo(Screen.LOGIN.route) { inclusive = true }
                }
            }

            false -> {
                snackbarHostState.showSnackbar(
                    message = "Invalid username or password.",
                    duration = SnackbarDuration.Short
                )
                loginViewModel.resetAuthenticationState()
            }

            null -> {}
        }
    }
```

```
}

@Composable
fun LoginForm(
    loginViewModel: LoginViewModel,
    paddingValues: PaddingValues
) {
    val usernameState by loginViewModel.username.observeAsState("")
    val rememberMeState by loginViewModel.rememberMe.observeAsState(false)
    var username by remember { mutableStateOf(usernameState) }
    var password by remember { mutableStateOf("") }
    var rememberMe by remember { mutableStateOf(rememberMeState) }
    val isLoginEnabled = username.isNotBlank() && password.isNotBlank()

    LaunchedEffect(usernameState, rememberMeState) {
        username = usernameState
        rememberMe = rememberMeState
        Log.d("PAM", "LoginForm: username $usernameState rememberMeState $rememberMeState")
    }

    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(Color.LightGray),
        contentAlignment = Alignment.Center,
    ) {
        Card(
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 32.dp)
                .padding(paddingValues),
            colors = CardDefaults.cardColors(containerColor = Color.White),
            elevation = CardDefaults.cardElevation(defaultElevation = 4.dp)
        ) {
            Column(
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(36.dp, 24.dp).verticalScroll(rememberScrollState()),
                horizontalAlignment = Alignment.CenterHorizontally,
                verticalArrangement = Arrangement.Center
            ) {
                Image(
                    painter = painterResource(id = R.drawable.ic_launcher_foreground),
                    contentDescription = "Logo",
                )
                Spacer(modifier = Modifier.height(20.dp))
                UsernameField(username, onUsernameChange = { username = it })
                PasswordField(password, onPasswordChange = { password = it })
                RememberMeSwitch(rememberMe) { isChecked -> rememberMe = isChecked }
                Spacer(modifier = Modifier.height(16.dp))
```

```
            LoginButton(isLoginEnabled) {
                loginViewModel.login(
                    username,
                    password,
                    rememberMe
                )
            }
        }
    }
}

@Composable
fun UsernameField(username: String, onUsernameChange: (String) -> Unit) {
    OutlinedTextField(
        modifier = Modifier.fillMaxWidth(),
        value = username,
        onValueChange = onUsernameChange,
        label = { Text("Username") },
    )
}

@Composable
fun PasswordField(password: String, onPasswordChange: (String) -> Unit) {
    OutlinedTextField(
        modifier = Modifier.fillMaxWidth(),
        value = password,
        onValueChange = onPasswordChange,
        label = { Text("Password") },
        visualTransformation = PasswordVisualTransformation()
    )
}

@Composable
fun LoginButton(isEnabled: Boolean, onLoginClick: () -> Unit) {
    Button(
        onClick = onLoginClick,
        enabled = isEnabled,
        colors = ButtonDefaults.buttonColors(
            containerColor = if (isEnabled) Color.DarkGray else Color.LightGray,
            contentColor = Color.White
        ),

        modifier = Modifier.fillMaxWidth().defaultMinSize(40.dp)
    ) {
        Text("Login", fontWeight = FontWeight.Bold)
    }
}

@Composable
```

```
fun RememberMeSwitch(rememberMe: Boolean, onCheckedChange: (Boolean) -> Unit) {
    var isChecked by remember { mutableStateOf(rememberMe) }

    Row(
        Modifier.fillMaxWidth(),
        verticalAlignment = Alignment.CenterVertically,
    ) {
        Switch(
            checked = isChecked,
            onCheckedChange = {
                isChecked = it
                onCheckedChange(it)
            },
            modifier = Modifier
                .scale(0.75f)
                .padding(0.dp),
        )
        Text("Remember Me?", modifier = Modifier.padding(start = 12.dp))
    }
}
```

Bước 3: Trong Screen khai báo thêm có route

```
enum class Screen(val route: String) {
    LOGIN("Login"),
    MOVIE_SCREEN("MovieScreen"),
    SCREEN1("Screen1"),
    SCREEN2("Screen2"),
    SCREEN3("Screen3"),
}
```

Bước 4: Trong ScreenNavigation, khai báo MainViewModel, movieState và các screen

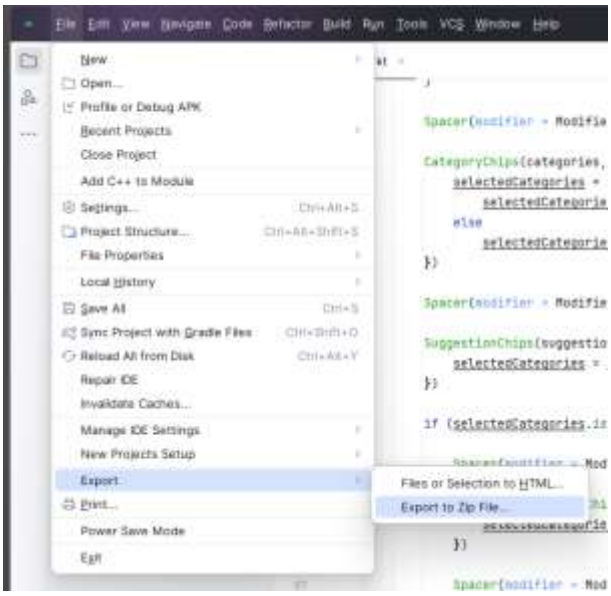
```
@Composable
fun ScreenNavigation() {
    val navController = rememberNavController()

    val mainViewModel: MainViewModel = viewModel()
    val moviesState = mainViewModel.movies.observeAsState(initial = emptyList())


    NavHost(
        navController = navController,
        startDestination = Screen.LOGIN.route,
    ) {
        composable(Screen.LOGIN.route) { LoginScreen(navController) }
        composable(Screen.MOVIE_SCREEN.route) { MovieScreen(moviesState.value) }
        composable(Screen.SCREEN1.route) { Screen1(navController) }
        composable(Screen.SCREEN2.route) { Screen2(navController) }
        composable(Screen.SCREEN3.route) { Screen3(navController) }

    }
}
```

Chạy thử ứng dụng.

BÀI 4: GV CHO THÊM

*** YÊU CẦU NỘP BÀI:

Sv nén file bao gồm các yêu cầu đã thực hiện trên, nộp lms đúng thời gian quy định của giảng viên. Không nộp bài coi như không có điểm.
Hướng dẫn nén project: **File** > **Export** > **Export to Zip File**.



--- Hết ---