МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное бюджетное образовательное учреждение высшего образования  
**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет информационных систем и технологий  
Кафедра «Информационные системы»  
Дисциплина «Программирование мобильных устройств»

Лабораторная работа №2

Выполнил:  
студент гр. ПИбд-32  
Бондаренко М. С.  
Проверил:  
доцент кафедры  
Филиппов А.А.

Ульяновск, 2023 г

**Задание**

Разработка экранных форм приложения и настройка навигации.

Необходимо:

1. На основе макетов, полученных в рамках ЛР № 1, разработать экранные формы приложения. При разработке экранных форм можно использовать фрагменты и/или композицию (compose).

2. Настроить навигацию между экранными формами приложения.

3. Разработать необходимые классы-сущности и создать на их основе тестовые наборы данных.

4. Использовать тестовые данные для демонстрации работы приложения.

**Выполнение**

1. Были разработаны экранные формы приложения на языке Kotlin с использованием композиции (compose). Пример простейшей экранной формы:

@Composable  
fun SettingsScreen(navController: NavHostController){  
 Column(  
 modifier = Modifier  
 .*fillMaxSize*(),  
 verticalArrangement = Arrangement.Top,  
 horizontalAlignment = Alignment.CenterHorizontally  
 ) **{** Image(  
 painter = painterResource(id = R.drawable.*settingsplaceholder*),  
 contentDescription = "settings",  
 contentScale = ContentScale.Crop,  
 modifier = Modifier  
 .*size*(320.*dp*)  
 .*padding*(8.*dp*))  
 IconButton(iconLeft = Icons.Default.*AccountCircle*, label = "Учётная запись",  
 backgroundColor = *ButtonColor2*, textColor = Color.White, onClickAction = **{** navController.navigate("edituser")  
 **}**)  
 IconButton(iconLeft = Icons.Default.*Face*, label = "Внешний вид",  
 backgroundColor = *ButtonColor1*, textColor = Color.Black, onClickAction = **{  
  
 }**)  
 IconButton(iconLeft = Icons.Default.*Share*, label = "Контакты",  
 backgroundColor = *ButtonColor1*, textColor = Color.Black, onClickAction = **{  
  
 }**)  
 IconButton(iconLeft = Icons.Default.*Info*, label = "О приложении",  
 backgroundColor = *ButtonColor1*, textColor = Color.Black, onClickAction = **{  
  
 }**)  
 IconButton(iconLeft = Icons.Default.*ExitToApp*, label = "Выйти",  
 backgroundColor = Color.Red, textColor = Color.White, onClickAction = **{** GlobalUser.getInstance().setUser(null)  
 navController.navigate("authorization")  
 **}**)  
 **}**}

2. Была настроена навигация между экранными формами приложения с помощью файла Navbar.

@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun NavBar(navController: NavHostController) {  
 val topBarState = rememberSaveable **{** (*mutableStateOf*(false)) **}** val bottomBarState = rememberSaveable **{** (*mutableStateOf*(false)) **}** Scaffold(  
 topBar = **{** AnimatedVisibility(  
 visible = topBarState.value,  
 enter = *slideInVertically*(initialOffsetY = **{ it }**),  
 exit = *slideOutVertically*(targetOffsetY = **{ it }**),  
 content = **{** TopAppBar(  
 title = **{** Text(  
 text = "Storyteller!",  
 textAlign = TextAlign.Center,  
 fontFamily = *FontFamily*(  
 *Font*(  
 R.font.*roboto\_regular*, FontWeight.Bold  
 )  
 )  
 )  
 **}** )  
 **}** )  
 **}**,  
 bottomBar = **{** AnimatedVisibility(  
 visible = bottomBarState.value,  
 enter = *slideInVertically*(initialOffsetY = **{ it }**),  
 exit = *slideOutVertically*(targetOffsetY = **{ it }**),  
 content = **{** NavigationBar **{** *navBarItems*.*forEach***{**item **->** NavigationBarItem(  
 icon = **{** Image(  
 painter = painterResource(item.icon),  
 contentDescription = null,  
 modifier = Modifier  
 .*size*(40.*dp*)  
 .*padding*(bottom = 6.*dp*)  
 )  
 **}**,  
 label = **{** Text(  
 text = item.label  
 )  
 **}**,  
 onClick = **{** navController.navigate(item.route)  
 **}**,  
 selected = false,  
 modifier = Modifier.*fillMaxSize*()  
 )  
 **}  
 }  
 }** )  
 **}**,  
 modifier = Modifier.*background*(Color.White)  
 ) **{**innerPaddings **->** NavHost(  
 navController = navController,  
 startDestination = "authorization",  
 modifier = Modifier.*padding*(innerPaddings)  
 ) **{** *composable*("authorization")**{** topBarState.value = true  
 bottomBarState.value = false  
 Authorization(navController = navController)  
 **}** *composable*("registration")**{** topBarState.value = true  
 bottomBarState.value = false  
 Registration(navController = navController)  
 **}** *composable*("main")**{** topBarState.value = false  
 bottomBarState.value = true  
 MainScreen(navController = navController)  
 **}** *composable*("story")**{** topBarState.value = false  
 bottomBarState.value = true  
 ListStoryScreen(navController = navController)  
 **}** *composable*("mail")**{** topBarState.value = false  
 bottomBarState.value = true  
 ListMailScreen(navController = navController)  
 **}** *composable*("settings")**{** topBarState.value = true  
 bottomBarState.value = true  
 SettingsScreen(navController = navController)  
 **}** *composable*("editstory")**{** // Без аргумента  
 topBarState.value = false  
 bottomBarState.value = false  
 EditStoryScreen(navController = navController)  
 **}** *composable*(  
 "editstory/{id}",  
 arguments = *listOf*(*navArgument*("id") **{** type = NavType.IntType **}**) //С аргументом  
 ) **{** backStackEntry **->** backStackEntry.arguments?.*let* **{** topBarState.value = false  
 bottomBarState.value = false  
 EditStoryScreen(navController = navController, storyId = **it**.getInt("id"))  
 **}  
 }** *composable*("editmail")**{** // Без аргумента  
 topBarState.value = false  
 bottomBarState.value = false  
 EditMailScreen(navController = navController)  
 **}** *composable*("edituser")**{** topBarState.value = false  
 bottomBarState.value = false  
 EditUserScreen(navController = navController)  
 **}** *composable*(  
 "viewstory/{id}",  
 arguments = *listOf*(*navArgument*("id") **{** type = NavType.IntType **}**) //С аргументом  
 ) **{** backStackEntry **->** backStackEntry.arguments?.*let* **{** topBarState.value = false  
 bottomBarState.value = false  
 StoryViewScreen(navController = navController, storyId = **it**.getInt("id"))  
 **}  
 }** *composable*(  
 "viewmail/{id}",  
 arguments = *listOf*(*navArgument*("id") **{** type = NavType.IntType **}**) //С аргументом  
 ) **{** backStackEntry **->** backStackEntry.arguments?.*let* **{** topBarState.value = false  
 bottomBarState.value = false  
 MailViewScreen(navController = navController, mailId = **it**.getInt("id"))  
 **}  
 }  
 }  
 }**}

3. Был разработан классы-сущности User.kt, Story.kt, Mail.kt, содержащий в себе поля: название, автор и содержание.

data class User(  
 val email: String,  
 val login: String,  
 val password: String  
)

data class Story(  
 val id: Int,  
 val title: String,  
 val description: String,  
 val cover: Int  
)

data class Mail(  
 val id: Int,  
 val userId: Int,  
 val username: String,  
 val message: String  
)

4. Демонстрация работы приложения. При запуске открывается главная форма, с которой можно перейти на форму регистрации или форму входа. После успешной регистрации пользователя перебрасывает на форму входа. После успешного входа в свой аккаунт пользователь попадает на главную страницу с историями, внизу появляется навигационная панель, по которой можно также перейти в настройки, свои истории или почту.

