

Проектирование структуры мобильного приложения на Android с использованием Jetpack Compose

Андриенко Иван Сергеевич

Приамурский государственный университет имени Шолом-Алейхема
Студент

Пащенко Екатерина Александровна

Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация

В статье рассматриваются особенности проектирования пользовательского интерфейса мобильного приложения для платформы Android с использованием декларативного подхода Jetpack Compose. Описывается организация экранов приложения, реализация навигации между вкладками, построение общего контейнера интерфейса и проектирование интерактивного экрана с использованием WebView и модального нижнего листа. Показано, что применение Jetpack Compose позволяет упростить архитектуру пользовательского интерфейса, повысить читаемость кода и обеспечить гибкость при проектировании сложных экранов.

Ключевые слова: мобильная разработка, Android, Jetpack Compose, UI/UX, навигация, WebView, Material Design, проектирование интерфейсов.

Designing the structure of an Android mobile application using Jetpack Compose

Andrienko Ivan Sergeevich

Sholom-Aleichem Priamursky State University
Student

Paschenko Ekaterina Alexandrovna

Sholom-Aleichem Priamursky State University
Student

Abstract

The article discusses the features of designing the user interface of a mobile application for the Android platform using the declarative Jetpack Compose approach. It describes the organization of application screens, the implementation of navigation between tabs, the construction of a common interface container, and the design of an interactive screen using WebView and a modal bottom sheet. It is shown that the use of Jetpack Compose makes it possible to simplify the architecture

of the user interface, increase code readability, and provide flexibility in designing complex screens.

Keywords: mobile development, Android, Jetpack Compose, UI/UX, navigation, WebView, Material Design, interface design.

1 Введение

1.1 Актуальность

Актуальность исследования обусловлена активным переходом мобильной разработки на декларативные подходы к построению пользовательского интерфейса. Jetpack Compose является современным инструментом разработки Android-приложений, позволяющим отказаться от XML-разметки и описывать интерфейс средствами языка Kotlin. В условиях роста сложности мобильных приложений особое значение приобретает проектирование масштабируемой структуры экранов, навигации и повторно используемых компонентов интерфейса.

1.2 Обзор исследований

В работе Е.Е. Старушенковой и Р.Р. Палютина рассматривается применение архитектурного паттерна MVVM в Android-приложениях на базе Jetpack Compose, где подчёркивается его роль в разделении логики и интерфейса, а также в повышении читаемости и сопровождаемости кода [1]. Исследование О.С. Флоринского и А.А. Барышникова посвящено особенностям построения графического интерфейса с использованием Jetpack Compose, акцентируя внимание на декларативной модели UI и её отличиях от традиционного XML-подхода в Android-разработке [2]. В работе А.В.Дындына и П.С. Новикова анализируется использование Kotlin Multiplatform и Jetpack Compose Multiplatform, что демонстрирует перспективы повторного использования UI-кода и унификации архитектурных решений в мобильной разработке [3]. Н.Е. Алексеев и А.Н.Алпатов исследуют механизмы управления состоянием в Jetpack Compose, подчёркивая важность корректной работы со state и side-effects для обеспечения предсказуемого поведения интерфейса [4]. В зарубежном исследовании С. Диантони, О. Комарудин и А. Ризал рассматривается применение архитектуры MVVM в сочетании с Jetpack Compose, где подтверждается эффективность данного подхода при проектировании масштабируемых и расширяемых Android-приложений [5].

1.3 Цель исследования

Цель исследования – разработать и описать архитектуру пользовательского интерфейса мобильного приложения на платформе Android с использованием Jetpack Compose, обеспечивающую удобную навигацию между экранами и реализацию интерактивных элементов интерфейса.

2 Материалы и методы

В качестве инструментальной основы использовалась среда разработки Android Studio и язык программирования Kotlin. Пользовательский интерфейс реализован с применением библиотеки Jetpack Compose и компонентов Material Design.

3 Результаты и обсуждения

В ходе проектирования пользовательского интерфейса был выбран декларативный подход Jetpack Compose, при котором каждый экран и его элементы описываются в виде функций с аннотацией `@Composable`. Такой подход позволил отказаться от иерархии XML-разметок и реализовать экранную структуру приложения в виде набора логически связанных компонентов. В качестве основного контейнера интерфейса был реализован компонент `MainScreen`, который объединяет навигацию между вкладками и общие элементы управления. Это решение позволило централизовать управление экранной логикой и обеспечить единообразие интерфейса во всём приложении (рис. 1).

```
20 import com.modeg.gymbook.ui.theme.GymBookTheme
21
22 1 Usage
23 </> class MainActivity : ComponentActivity() {
24   @Override fun onCreate(savedInstanceState: Bundle?) {
25     super.onCreate(savedInstanceState)
26     enableEdgeToEdge(
27       statusBarStyle = SystemBarStyle.dark( scrim = Color.TRANSPARENT),
28       navigationBarStyle = SystemBarStyle.dark( scrim = Color.TRANSPARENT)
29     )
30     setContent {
31       GymBookTheme {
32         val navController = rememberNavController()
33         GymBookNavHost(
34           navController = navController,
35           modifier = Modifier.fillMaxSize()
36         )
37       }
38     }
39   }
40 }
```

Рис. 1. Общая структура экранов приложения в Jetpack Compose

Навигация между основными экранами приложения реализована с использованием библиотеки Navigation Compose. Внутри `MainScreen` создаётся навигационный контроллер, а переключение вкладок осуществляется через `NavHost` с набором `composable`-маршрутов. Такое решение позволяет изолировать логику навигации от конкретных экранов и упростить добавление новых вкладок в будущем. Каждый экран реализован в виде отдельной `composable`-функции, что повышает модульность и читаемость кода (рис. 2).

```

1 Usage
@Composable
private fun GymBookNavHost(
    navController: NavHostController,
    modifier: Modifier = Modifier
) {
    val auth = FirebaseAuth.getInstance()
    NavHost(
        navController = navController,
        startDestination = Destinations.Splash.route,
        modifier = modifier
    ) {
        composable(Destinations.Splash.route) {
            SplashScreen(
                onTimeout = {
                    val destination = if (auth.currentUser == null) {
                        Destinations.Auth.route
                    } else {
                        Destinations.Main.route
                    }
                    navController.navigate(route = destination) {
                        popUpTo(Destinations.Splash.route) { inclusive = true }
                    }
                }
            )
        }
        composable(Destinations.Auth.route) {
            AuthScreen(
                onAuthSuccess = {
                    navController.navigate(Destinations.Main.route) {
                        popUpTo(Destinations.Auth.route) { inclusive = true }
                    }
                }
            )
        }
        composable(Destinations.Main.route) {
            MainScreen(
                onSignedOut = {
                    navController.navigate(Destinations.Auth.route) {
                        popUpTo(Destinations.Main.route) { inclusive = true }
                    }
                }
            )
        }
    }
}

```

Рис. 2. Реализация навигации между вкладками с использованием NavHost

Для реализации общих действий навигации создана отдельная composable-функция TopControls, отображающая кнопки возврата и перехода к настройкам. Панель размещается поверх основного контента и фиксируется в верхней части экрана с учётом системных отступов.

Нижнее меню приложения реализовано в виде кастомного composable-компонента GymBottomBar. В отличие от стандартного BottomNavigation, был реализован собственный дизайн с анимацией активной вкладки. При выборе активной вкладки иконка плавно смещается вверх, а под ней отображается текстовая подпись. Анимация реализована с использованием animateDpAsState, что позволяет обеспечить визуальную обратную связь пользователю и повысить выразительность интерфейса (рис. 3).

```
148         if (showBottomBar) {
149             GymBottomBar(
150                 modifier = Modifier
151                     .align(Alignment.BottomCenter)
152                     .navigationBarsPadding()
153                     .padding(start = 16.dp, end = 16.dp, bottom = 2.dp),
154                 destinations = destinations,
155                 currentRoute = currentRoute,
156                 accent = accent,
157                 surfaceColor = surfaceColor,
158                 onDestinationClick = { route ->
159                     navController.navigate(route) {
160                         popUpTo( route = MainRoutes.Anatomy) { saveState = true }
161                         launchSingleTop = true
162                         restoreState = true
163                     }
164                 }
165             )
166         }
167     }
168 }
169
170 1 Usage
171 @Composable
172 private fun TopControls(
173     modifier: Modifier,
174     canNavigateBack: Boolean,
175     onBack: () -> Unit,
176     onSettings: () -> Unit
177 ) {
```

Рис. 3. Реализация нижнего меню

Экран отображения анатомии реализован с использованием контейнера `Box`, который позволяет накладывать элементы интерфейса друг на друга. Это дало возможность совместить фон, интерактивный контент и модальный нижний лист в рамках одного экрана. Для отображения интерактивного изображения тела человека использован стандартный Android `WebView`, встроенный в `Compose` с помощью компонента `AndroidView`. Это позволило использовать SVG-графику и JavaScript-логику без переписывания существующего кода под `Compose`.

Взаимодействие пользователя с `WebView` передаётся в Kotlin через `JavascriptInterface`, что обеспечивает двустороннюю связь между веб-контентом и `Compose`-интерфейсом (рис. 4).

```

147 @OptIn( ...markerClass = ExperimentalMaterial3Api::class)
148 @Composable
149 fun AnatomyScreen() {
150     val context = LocalContext.current
151     var selectedMuscleId by rememberSaveable { mutableStateOf<String?>(value = null) }
152     val sheetState = rememberModalBottomSheetState(skipPartiallyExpanded = true)
153     val exercises = remember( key1 = selectedMuscleId) { AnatomyData.exercisesFor(selectedMuscleId) }
154
155     Box(modifier = Modifier.fillMaxSize()) {
156         Image(
157             painter = painterResource(id = R.drawable.background2),
158             contentDescription = null,
159             modifier = Modifier.fillMaxSize(),
160             contentScale = ContentScale.Crop
161         )
162
163         Box(
164             modifier = Modifier
165                 .fillMaxSize()
166                 .background(
167                     Brush.verticalGradient(
168                         colors = listOf(Color( color = 0x55000000), Color( color = 0x33000000), Color( color = 0x55000000))
169                     )
170                 )
171         )
172
173         BodyMapWebView(
174             modifier = Modifier
175                 .fillMaxSize()
176                 .padding(bottom = 60.dp),
177             onClick = { selectedMuscleId = it }
178         )
179
180         if (selectedMuscleId != null) {
181             val labelRes = MuscleLabels[selectedMuscleId]
182             val title = if (labelRes != null) {
183                 stringResource(id = labelRes)
184             } else {
185                 selectedMuscleId ?: ""
186             }
187             ModalBottomSheet(

```

Рис. 4. Реализация интерактивной модели человека

Реализованная архитектура интерфейса показала, что Jetpack Compose позволяет эффективно проектировать сложные экраны с несколькими уровнями взаимодействия. Компонентный подход упрощает поддержку и расширение приложения, а интеграция классических Android-компонентов (WebView) делает возможным постепенный переход на декларативный UI без полного отказа от существующих решений. Итоговый вид экрана (рис. 5).

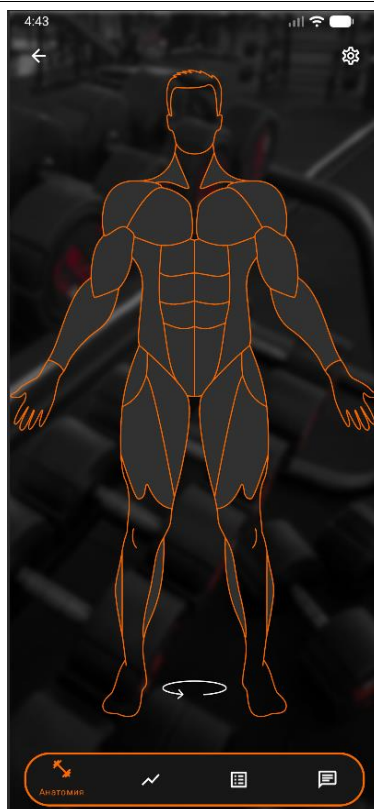


Рис. 5. Итоговый вид экрана

4 Выводы

В ходе работы была разработана структура пользовательского интерфейса мобильного приложения для платформы Android с использованием Jetpack Compose. Реализованы навигация между экранами, пользовательские элементы управления, интерактивный экран с WebView и динамическое отображение данных в модальном нижнем листе. Использование декларативного подхода позволило повысить модульность интерфейса, сократить объём связанного кода и упростить управление состоянием. Полученные результаты подтверждают целесообразность применения Jetpack Compose при проектировании современных мобильных приложений со сложной навигацией и интерактивными элементами.

Библиографический список

1. Старушенкова Е.Е., Палютин Р.Р. Паттерн проектирования MVVM, как один из способов написания «чистого» кода в Android-приложении на Jetpack Compose // E-Scio. 2023. № 4 (79). С. 440–445.
2. Флоринский О.С., Барышников А.А. Графический интерфейс Jetpack Compose // Вестник Ессентукского института управления, бизнеса и права. 2023. № 20. С. 139–145.
3. Дындин А.В., Новиков П.С. Исследование применения Kotlin Multiplatform и Jetpack Compose Multiplatform в мобильной разработке // Вестник науки. 2024. Т. 3. № 4 (73). С. 410–421.
4. Алексеев Н.Е., Алпатов А.Н. Обработка состояния Jetpack Compose в

-
- Android-разработке // Научный аспект. 2024. Т. 16. № 5. С. 2098–2104.
5. Diantoni C., Komarudin O., Rizal A. Arsitektur MVVM dan framework Jetpack Compose pada pengembangan aplikasi Android // JATI (Jurnal Mahasiswa Teknik Informatika). 2024. Т. 8. № 3. С. 3216–3224.