Студенческая
ИТ-лаборатория

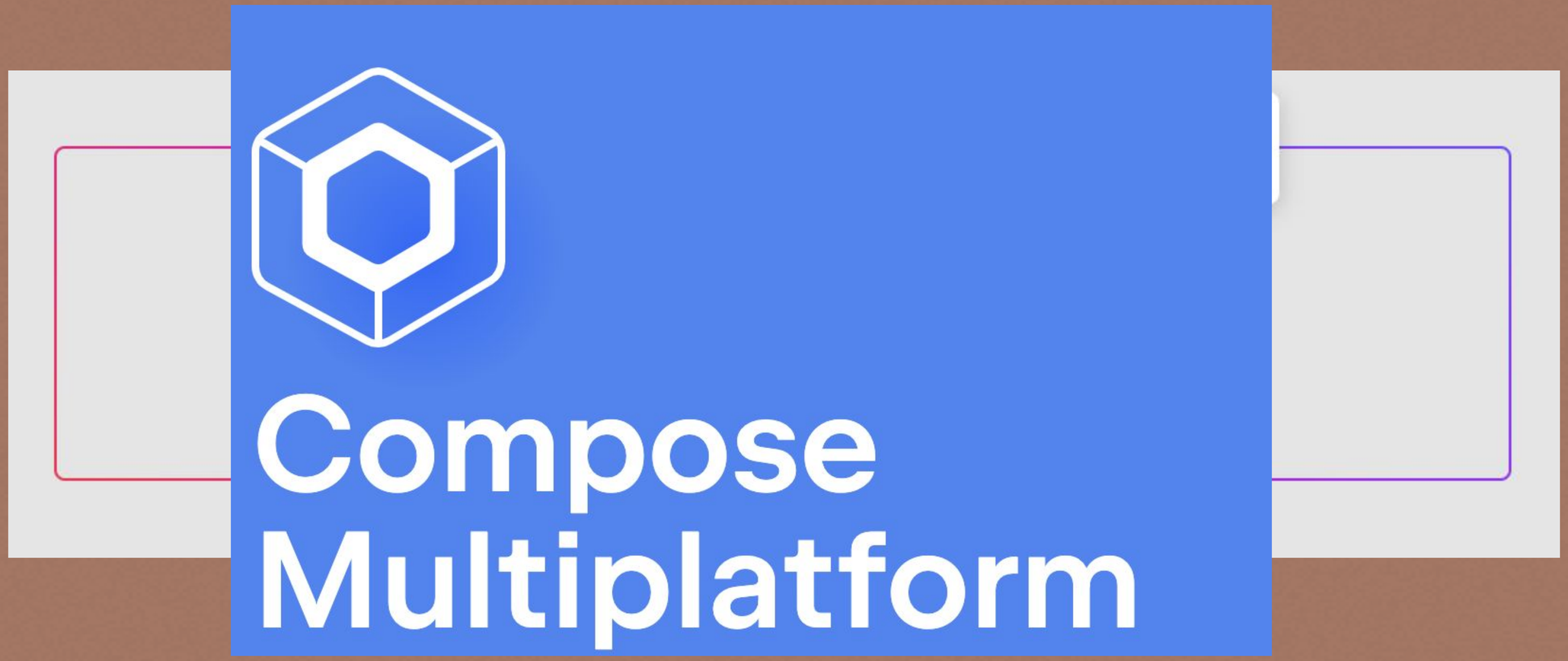
Основы разработки Android-приложений. UI и Навигация

Александр Григорьев
Android-разработчик в компании Effective

Почему Android?

Kotlin Multiplatform

effectiveband



План

effectiveband

- Архитектура
- Подходы к реализации UI
- Compose
- Реализация UI
- Навигация

Студенческая
ИТ-лаборатория

effectiveband

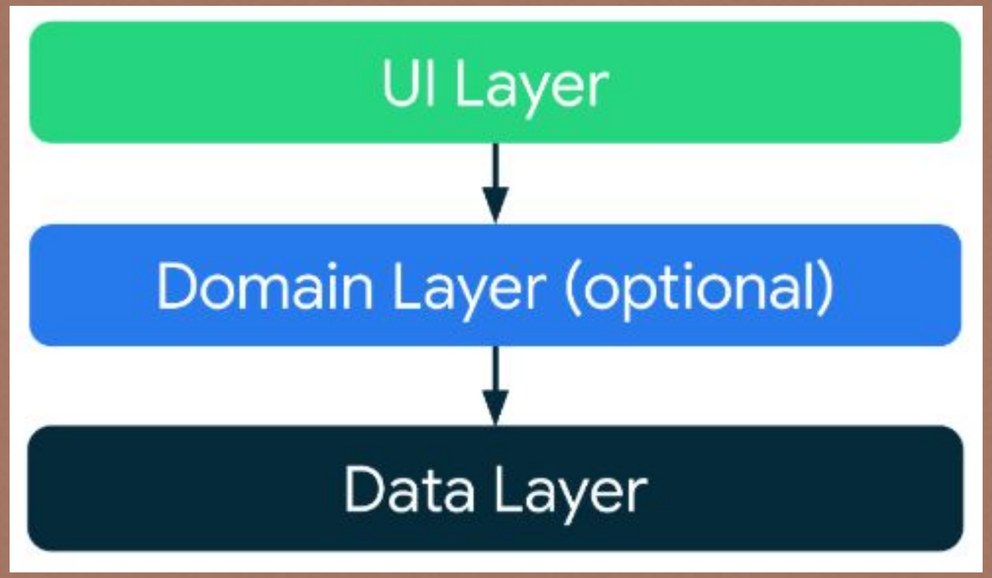
Архитектура и зачем она нужна?

Чистая архитектура

effectiveband

Масштабируемость
Тестирование
Понимание кода

Мы экономим время!



Студенческая
ИТ-лаборатория

effectiveband

UI-слой в Android

Подходы

effectiveband

- Activity - каждый экран. Открытие экрана - Intent
- Переход на фрагменты
- Переход на Compose

Студенческая
ИТ-лаборатория

effectiveband

Compose или XML?

Студенческая
ИТ-лаборатория

effectiveband

Compose

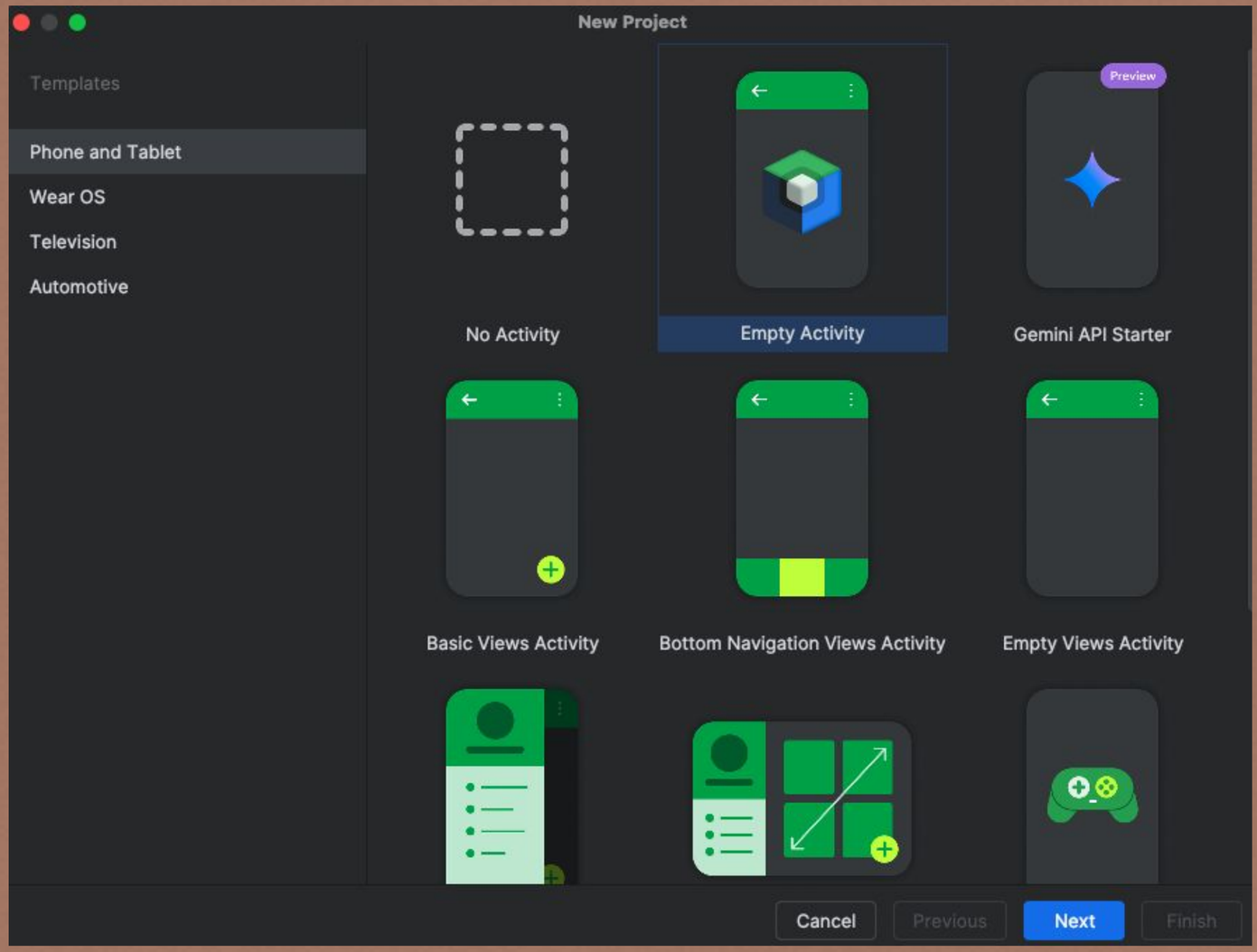
Студенческая
ИТ-лаборатория

effectiveband

Время делать приложение

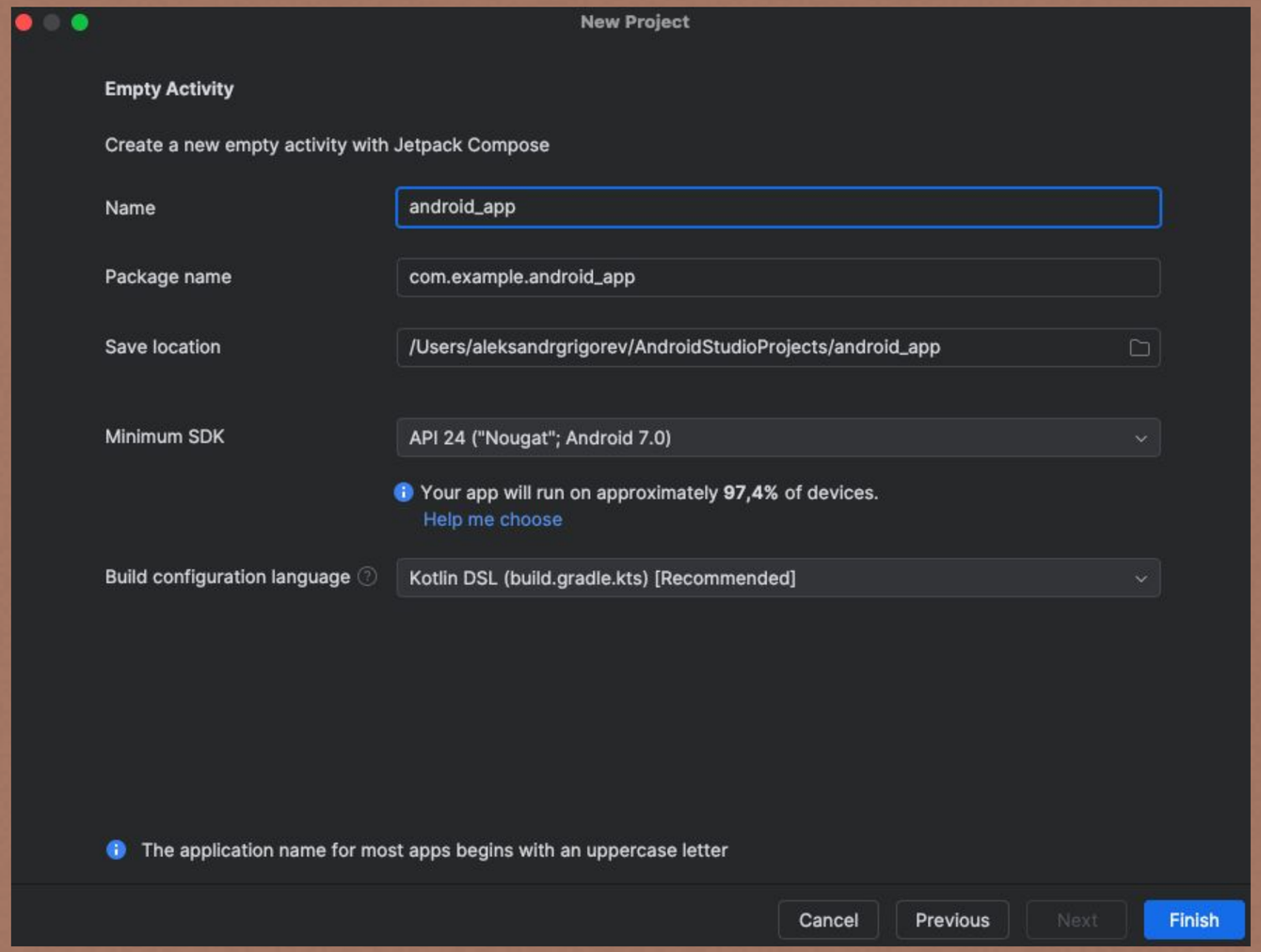
Создаем проект

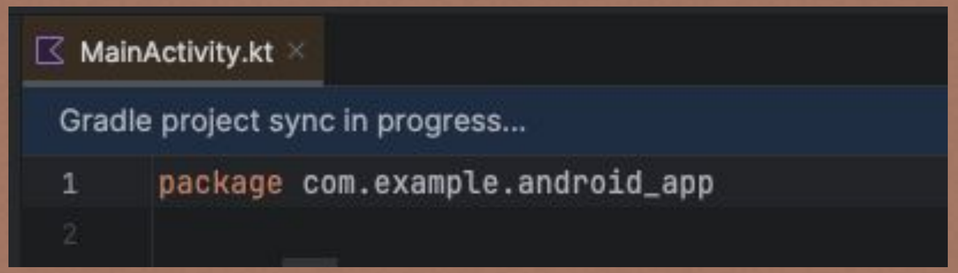
effectiveband



Создаем проект

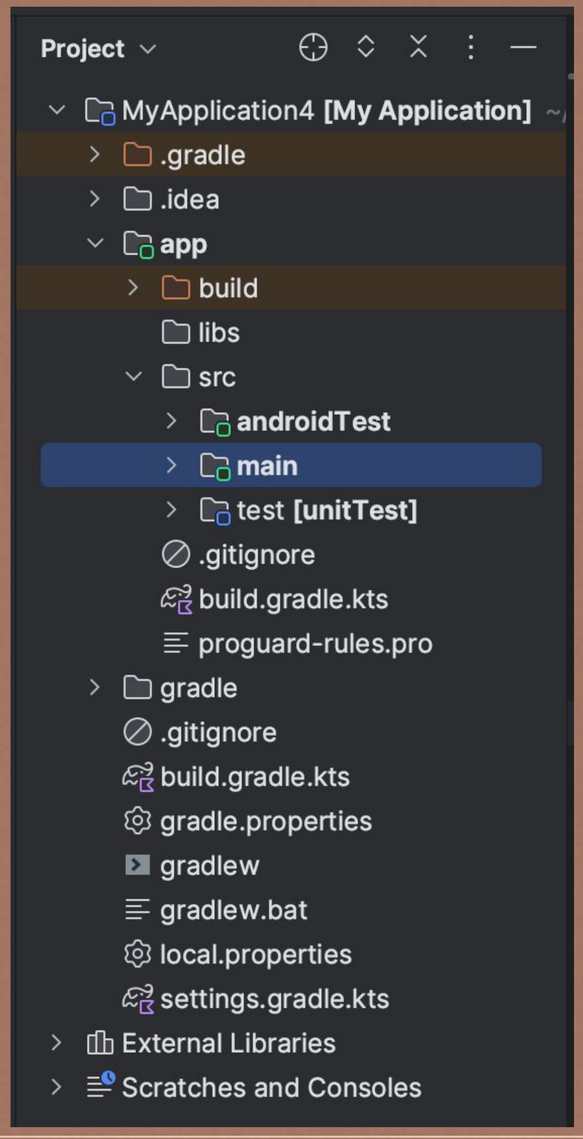
effectiveband

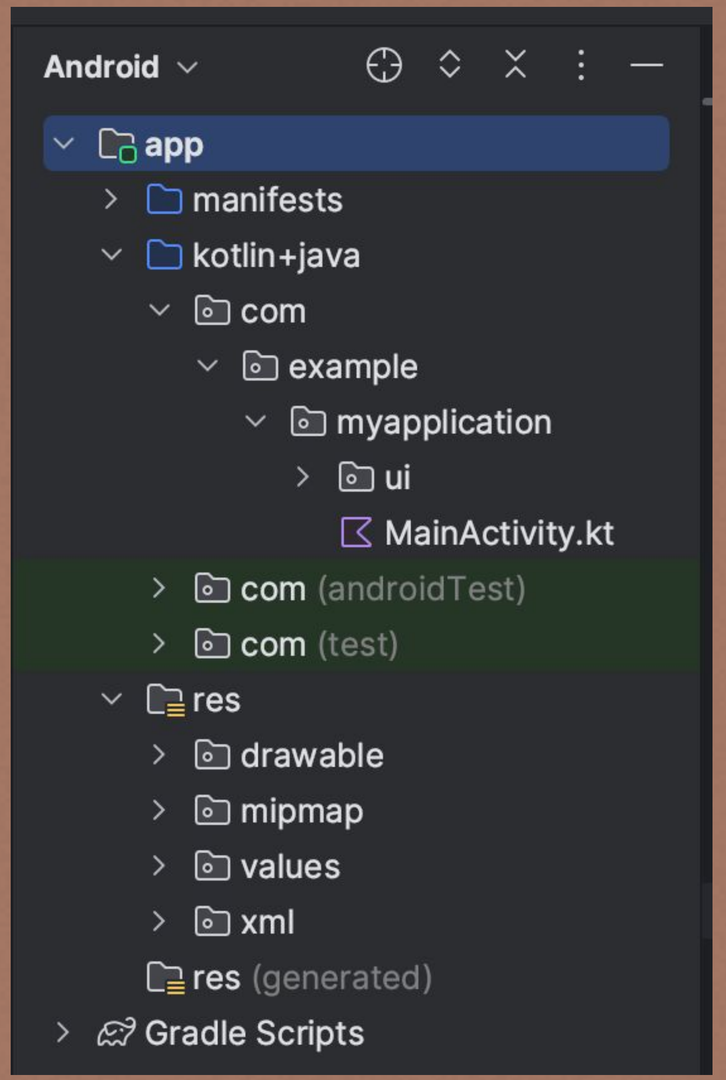




Структора проекта

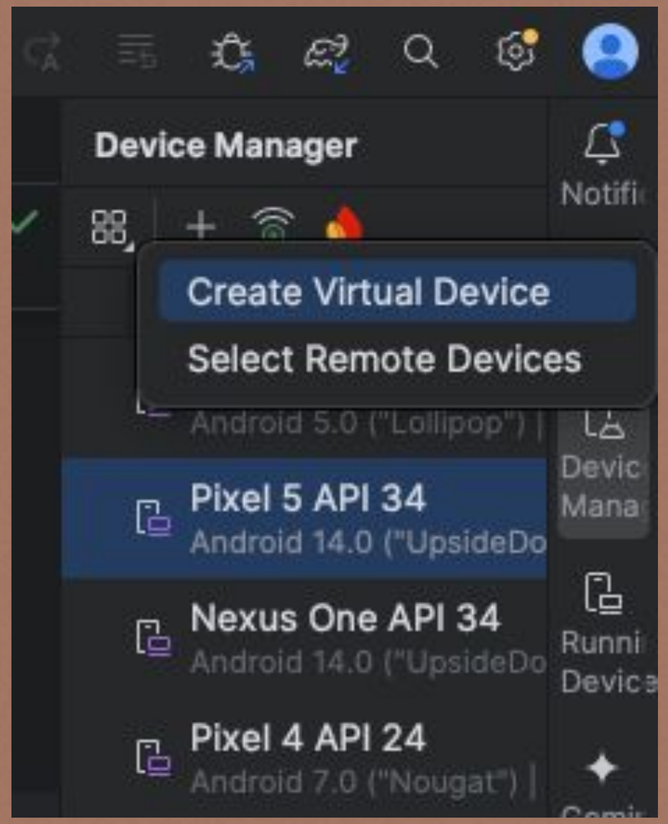
effectiveband

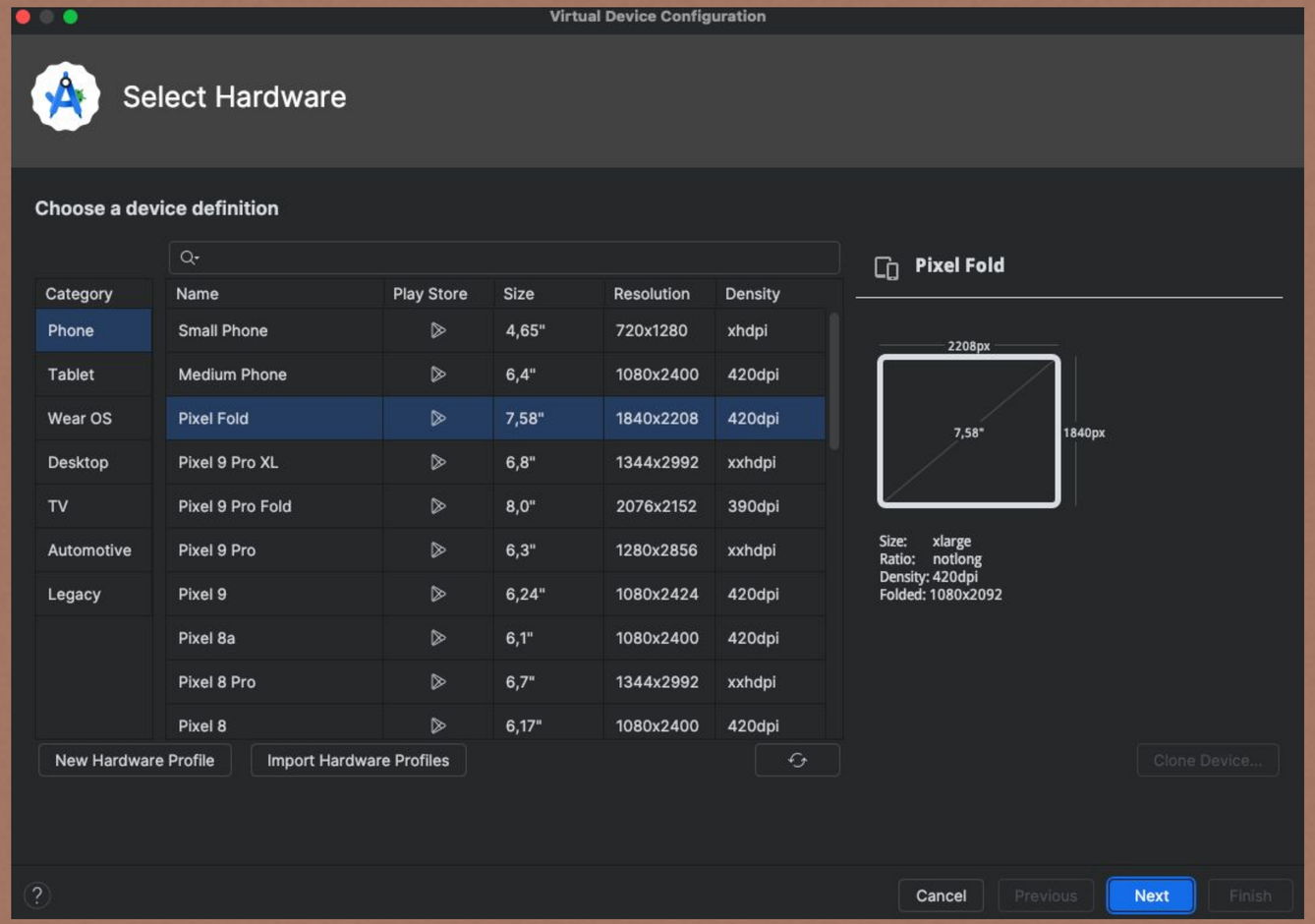




Подключаем эмулятор

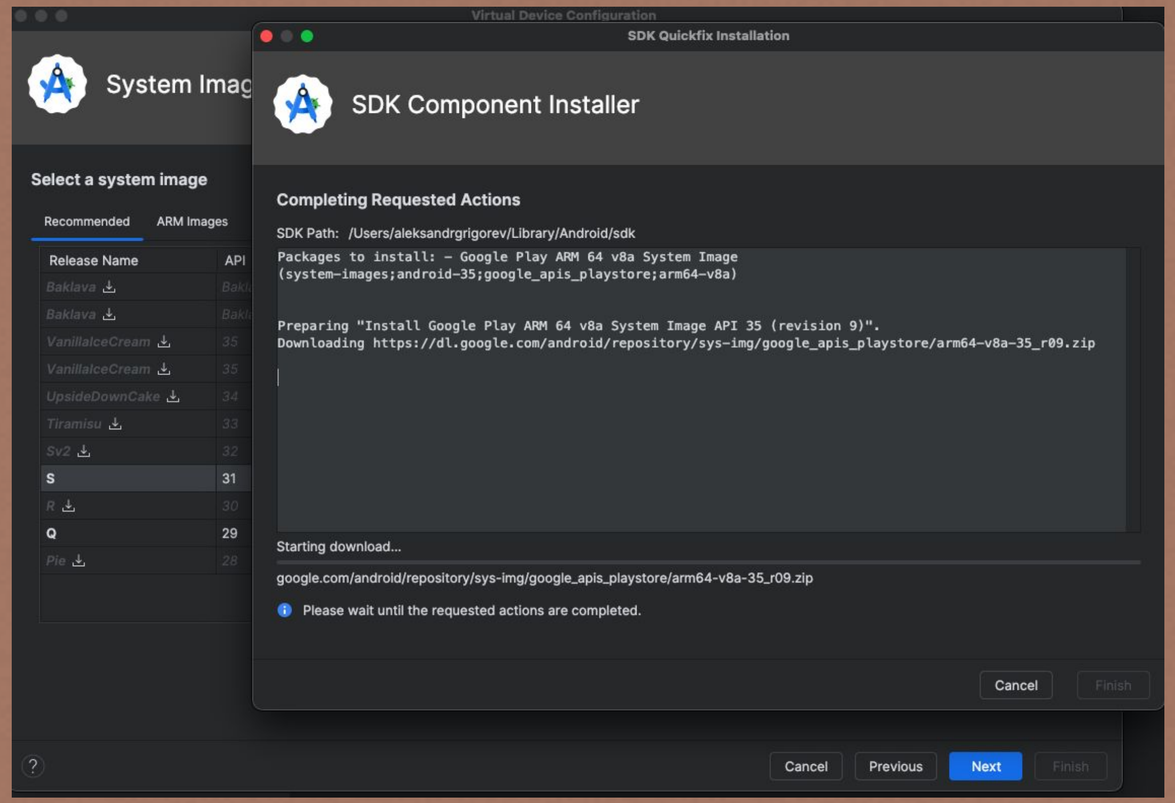
effectiveband

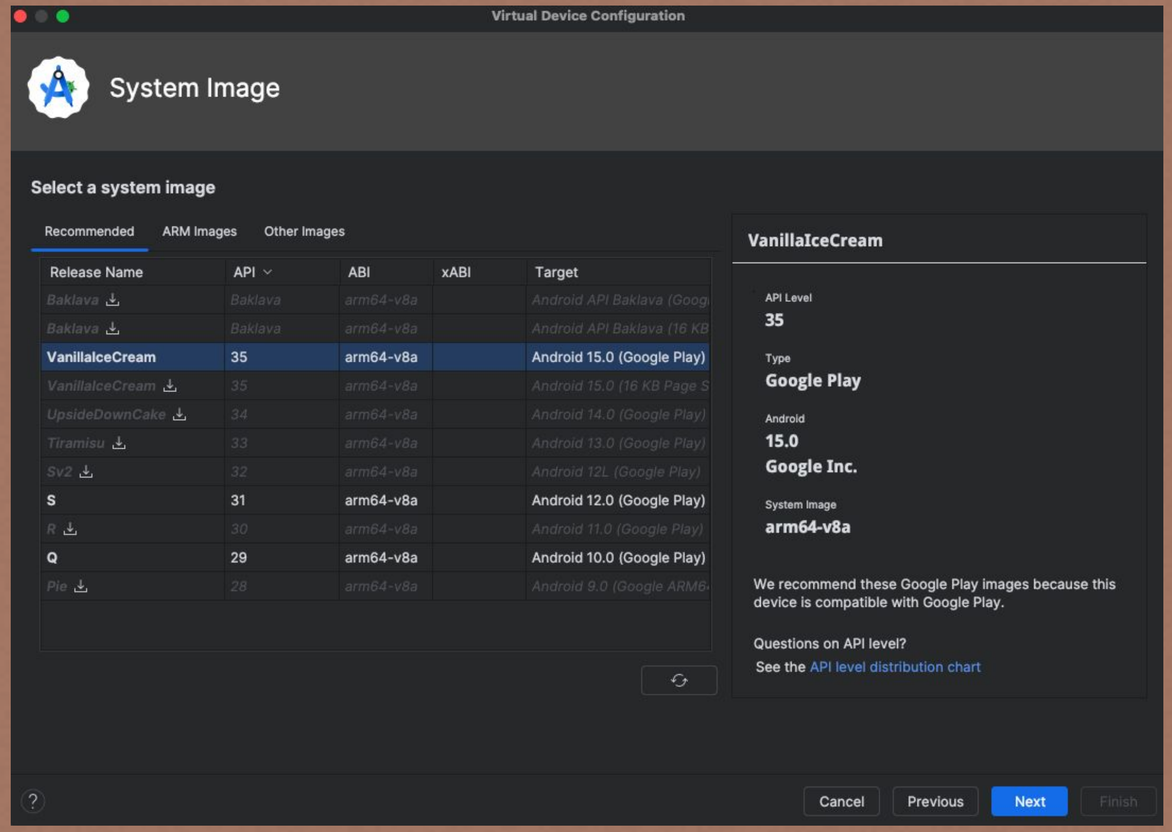




Подключаем эмулятор

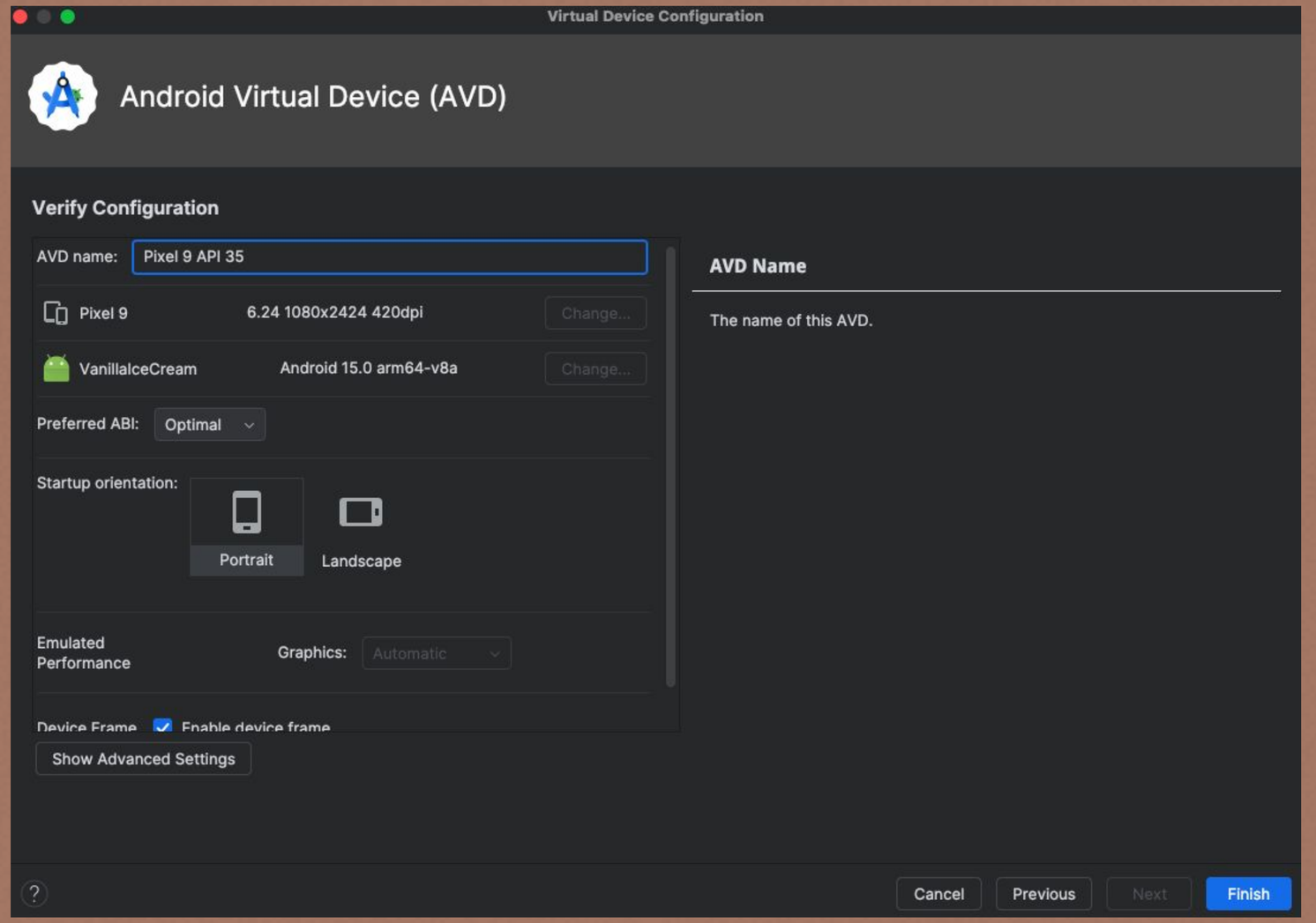
effectiveband





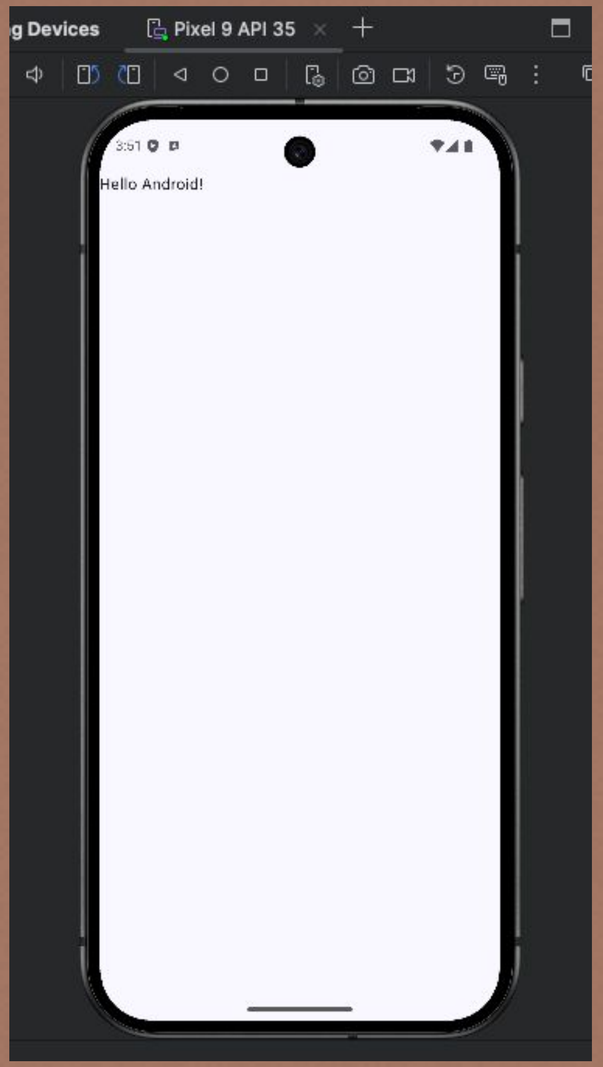
Подключаем эмулятор

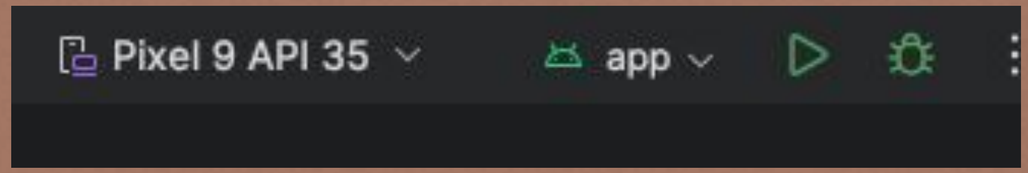
effectiveband



Подключаем эмулятор

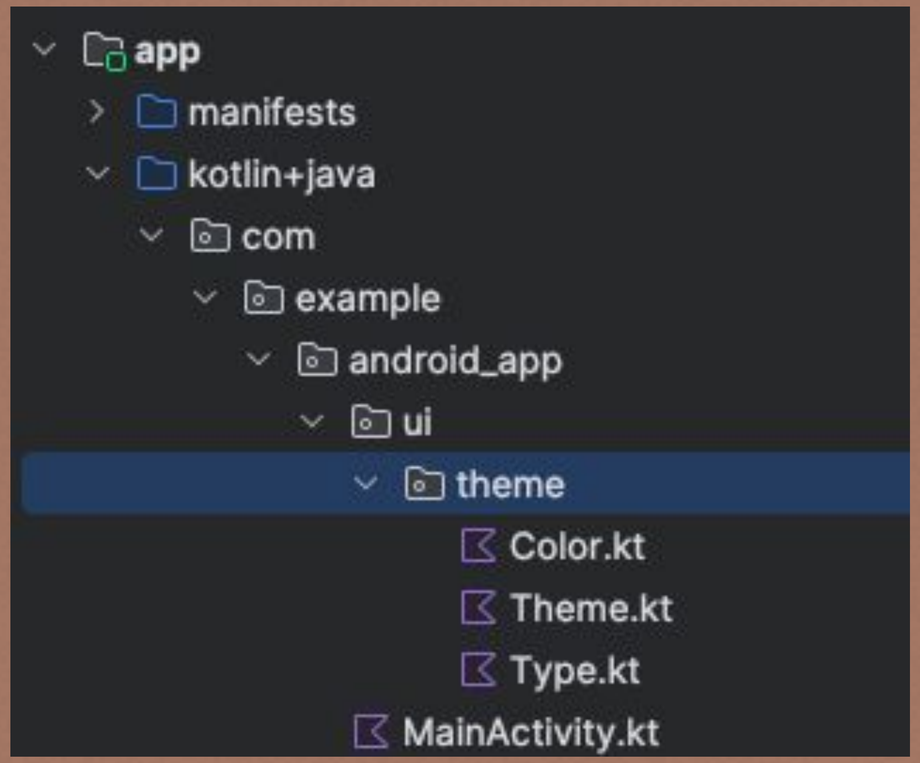
effectiveband





Что имеем?

effectiveband




```
private val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)

private val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = PurpleGrey40,
    tertiary = Pink40
```

```
4
5  val Purple80 = Color( color: 0xFFD0BCFF)
6  val PurpleGrey80 = Color( color: 0xFFCCC2DC)
7  val Pink80 = Color( color: 0xFFEFB8C8)
8
9  val Purple40 = Color( color: 0xFF6650a4)
10 val PurpleGrey40 = Color( color: 0xFF625b71)
11 val Pink40 = Color( color: 0xFF7D5260)
```

Что имеем?

effectiveband

- Стартуем в Activity
- Точка входа для compose - setContent
- Compose-функции помечаются аннотацией @Compose
- Аннотация @Preview для отладки верстки (отображение)

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Android_appTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        name = "Android",
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    Android_appTheme {
        Greeting( name: "Android")
    }
}
```

Что имеем?

effectiveband

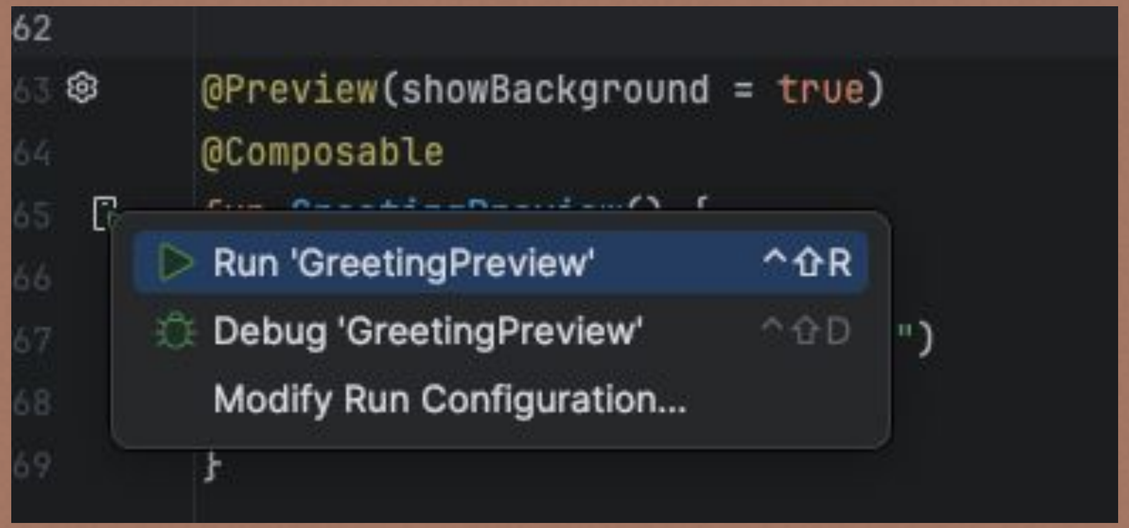
```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
}
```

```
@Composable
fun Text(
    text: String,
    modifier: Modifier = Modifier,
    color: Color = Color.Unspecified,
    fontSize: TextUnit = TextUnit.Unspecified,
    fontStyle: FontStyle? = null,
    fontWeight: FontWeight? = null,
    fontFamily: FontFamily? = null,
    letterSpacing: TextUnit = TextUnit.Unspecified,
    textDecoration: TextDecoration? = null,
    textAlign: TextAlign? = null,
    lineHeight: TextUnit = TextUnit.Unspecified,
    overflow: TextOverflow = TextOverflow.Clip,
    softWrap: Boolean = true,
    maxLines: Int = Int.MAX_VALUE,
    minLines: Int = 1,
    onTextLayout: ((TextLayoutResult) -> Unit)? = null,
    style: TextStyle = LocalTextStyle.current
) {
```

Играемся с версткой

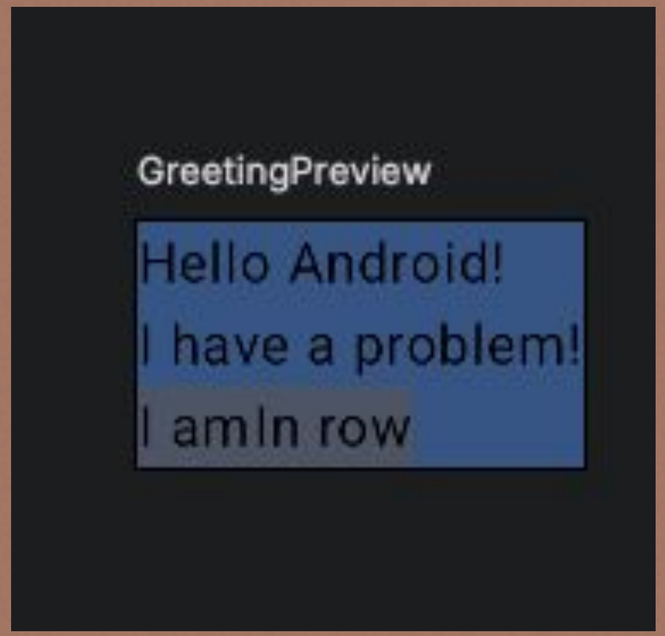
effectiveband

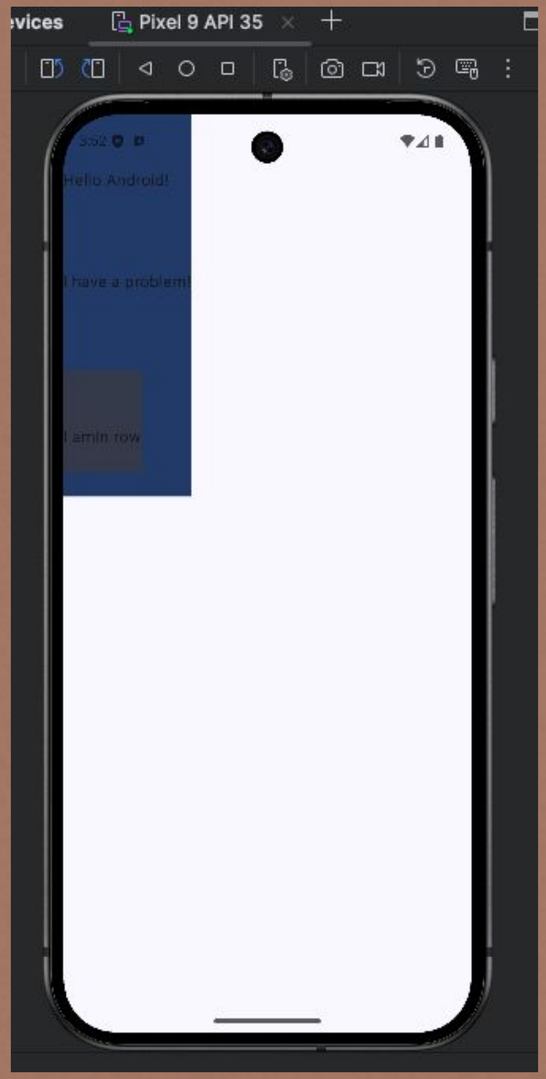
```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Column(
        modifier = Modifier.background(MaterialTheme.colorScheme.primary)
    ) { this: ColumnScope
        Text(
            text = "Hello $name!",
            modifier = modifier
        )
        Text(
            text = "I have a problem!",
            modifier = modifier
        )
        Row(modifier = modifier.background(MaterialTheme.colorScheme.secondary)) {
            Text(
                text = "I am",
                modifier = modifier
            )
            Text(
                text = "In row",
                modifier = modifier
            )
        }
    }
}
```



Играемся с версткой

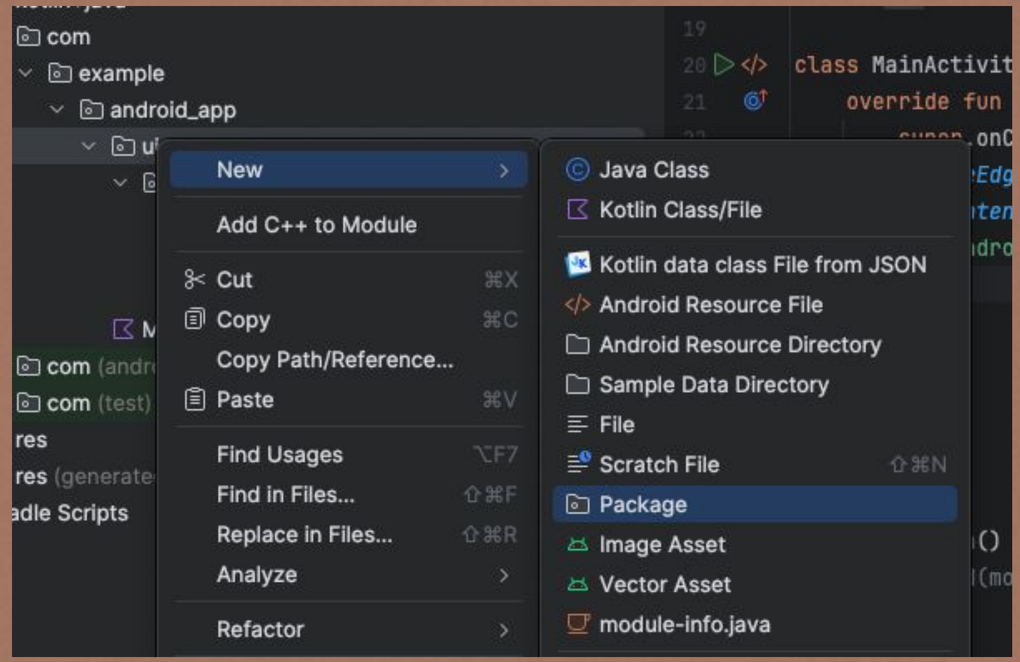
effectiveband

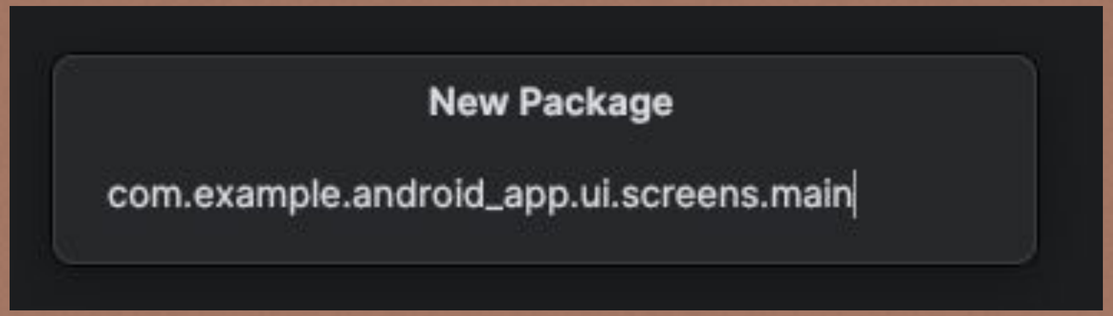


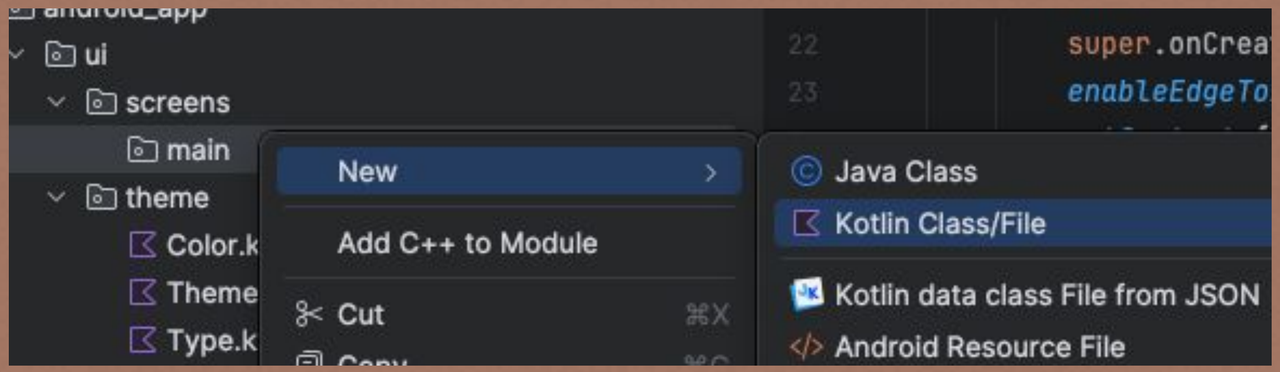


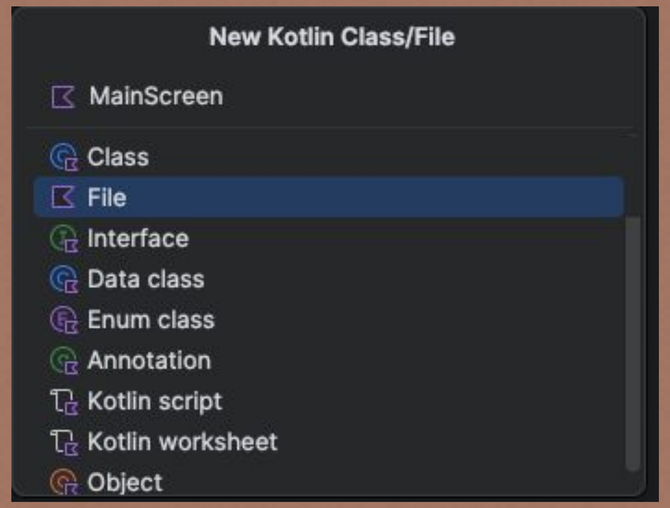
Создаем экран

effectiveband









Создаем экран

effectiveband

```
@Composable
fun MainScreen() {
    Scaffold(
        topBar = {},
        bottomBar = {},
    ) { innerPadding ->

    }
}
```

```
@Composable
public fun Scaffold(
    modifier: Modifier = Modifier,
    topBar: @Composable () -> Unit = {},
    bottomBar: @Composable () -> Unit = {},
    snackbarHost: @Composable () -> Unit = {},
    floatingActionButton: @Composable () -> Unit = {},
    floatingActionButtonPosition: FabPosition = FabPosition.End,
    containerColor: Color = MaterialTheme.colorScheme.background,
    contentColor: Color = contentColorFor(containerColor),
    contentWindowInsets: WindowInsets = ScaffoldDefaults.contentWindowInsets,
    content: @Composable (PaddingValues) -> Unit
): Unit
```

Создаем экран

effectiveband

```
@Composable
fun MainScreen() {
    Scaffold(
        topBar = {},
        bottomBar = {},
    ) { innerPadding ->
        Column(
            modifier = Modifier.padding(innerPadding)
        ) { this: ColumnScope


        }
    }
}
```

```
@Composable
fun TopBar() {
    Row() { this: RowScope
        IconButton() {  }
        Text(text = stringResource(R.string.title))
    }
}
```

```
@Composable
public fun IconButton(
    onClick: () -> Unit,
    modifier: Modifier = Modifier,
    enabled: Boolean = true,
    colors: IconButtonColors = IconButtonDefaults.iconButtonColors(),
    interactionSource: MutableInteractionSource = remember { MutableInteractionSource() },
    content: @Composable () -> Unit
): Unit
```

Доделываем top bar

effectiveband

```
@Composable
fun TopBar(
    title: String,
    onBackClick: () -> Unit
) {
    Row(
        modifier = Modifier.padding(16.dp).fillMaxWidth(),
        horizontalArrangement = Arrangement.SpaceBetween
    ) { this: RowScope

        IconButton(onClick = onBackClick) {
            Icon(Icons.AutoMirrored.Filled.ArrowBack, contentDescription: "Back")
        }
        Text(title)
    }
}
```

Добавляем кнопку

effectiveband

```
@Composable
fun PrimaryButton(
    modifier: Modifier = Modifier,
    text: String,
    onClick: () -> Unit) {
    Button(
        modifier = modifier.padding(8.dp).height(60.dp),
        onClick = onClick
    ) { this: RowScope
        Text(text = text)
    }
}
```

Финалим экран

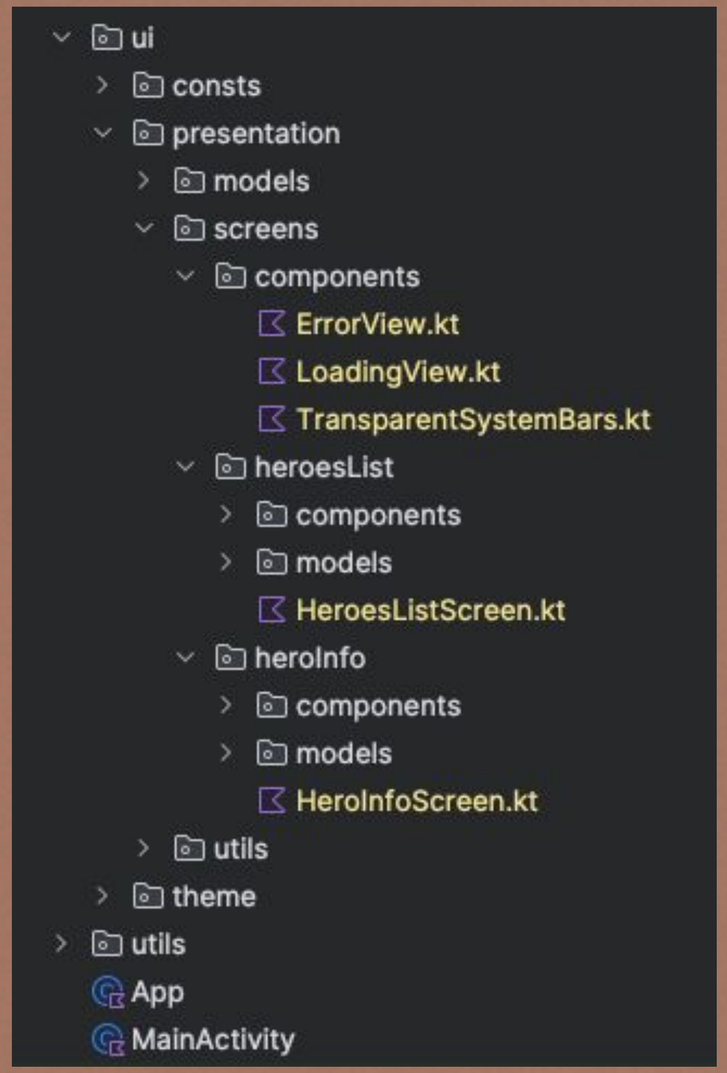
effectiveband

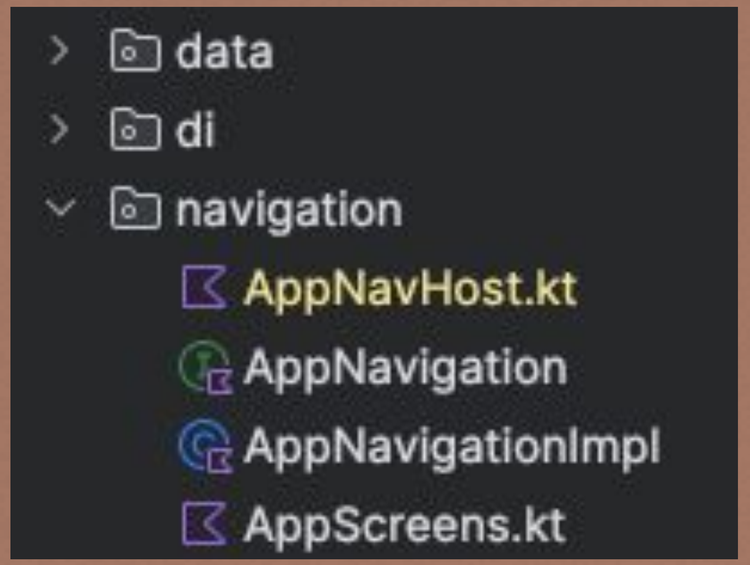
```
@Composable
fun MainScreen() {
    Scaffold(
        topBar = {
            TopBar(title = stringResource(R.string.title)) {}
        },
        bottomBar = {
            PrimaryButton(
                modifier = Modifier.fillMaxWidth(),
                text = stringResource(R.string.button_text),
                onClick = {}
            ) },
    ) { innerPadding ->
        Column(
            modifier = Modifier.padding(innerPadding)
        ) { this: ColumnScope


        }
    }
}
```

Организация файлов в проекте

effectiveband





А что делать дальше?

effectiveband

- Проблема сохранения состояния
- Как подвязать бизнес-логику к UI?

ViewModel

effectiveband

- Содержит бизнес-логику и отвечает за состояние экрана

- Имеет жизненный цикл

```
class HeroInfoViewModel @Inject constructor(
    private val navigation: AppNavigation,
    private val repository: Repository,
) : ViewModel() {

    private val _uiState = MutableStateFlow(HeroInfoUiState.Empty)
    val uiState: StateFlow<HeroInfoUiState> = _uiState.asStateFlow()

    fun sendEvent(event: HeroInfoEvent) {
        when (event) {
            is HeroInfoEvent.LoadHeroInfo -> getHeroInfo(event.value)
            HeroInfoEvent.PopBack -> popBack()
        }
    }
```

Студенческая
ИТ-лаборатория

effectiveband

Паттерны

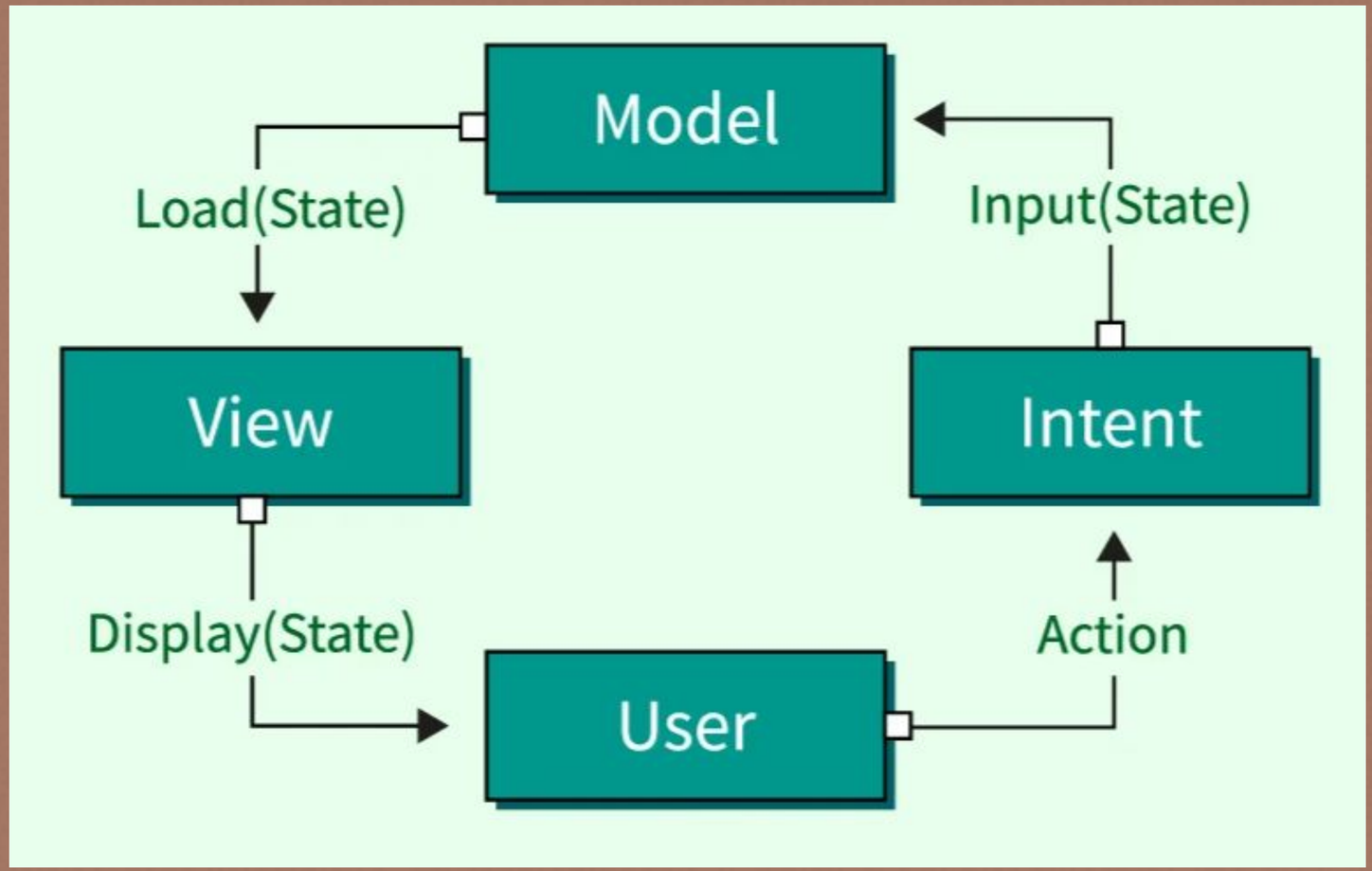
Паттерны

effectiveband

- MVC
- MVP
- MVVM
- MVI

MVI

effectiveband



Состояние экрана

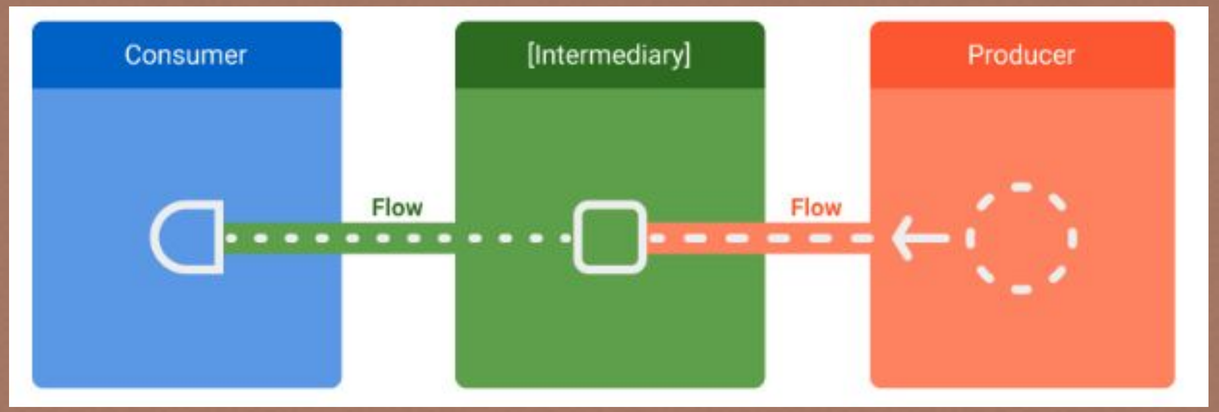
effectiveband

- Что-то на древнем
- LiveData - не используем
- Flow (StateFlow и SharedFlow)

Flow

effectiveband

- Поток данных на который подписываются
- Преимущество: постоянный поток данных



Состояние экрана

effectiveband

State

```
data class HeroesListUiState(
    val heroesList: List<HeroDataUi>,
    val stateUi: StateUi,
    val isRefreshing: Boolean,
) {
    companion object {
        val Empty = HeroesListUiState(
            stateUi = StateUi.Loading,
            isRefreshing = false,
            heroesList = emptyList()
        )
    }
}
```

ViewModel

```
private val _uiState = MutableStateFlow(HeroesListUiState.Empty)
val uiState : StateFlow<HeroesListUiState> = _uiState.asStateFlow()
```

Экран

```
val uiState = viewModel.uiState.collectAsState()
```

Отправка и получение эвентов

effectiveband

Экран

```
@Composable
fun HeroInfoScreen(viewModel: HeroInfoViewModel = viewModel(), id: Int) {

    val uiState = viewModel.uiState.collectAsState()

    LaunchedEffect(key1 = Unit, block = { this: CoroutineScope
        viewModel.sendEvent(HeroInfoEvent.LoadHeroInfo(id))
    })

    when (uiState.value.stateUi) {
        StateUi.Error -> ErrorView()
        StateUi.Loading -> LoadingView()
        StateUi.Success -> HeroInfoView(heroInfo = uiState.value.heroInfo!!)
    }

    HeroScreenTopBar {
        viewModel.sendEvent(HeroInfoEvent.PopBack)
    }
}
```

Эвенты

```
sealed class HeroInfoEvent {
    object PopBack : HeroInfoEvent()
    data class LoadHeroInfo(val value: Int) : HeroInfoEvent()
}
```

ViewModel

```
fun sendEvent(event: HeroInfoEvent) {
    when (event) {
        is HeroInfoEvent.LoadHeroInfo -> getHeroInfo(event.value)
        HeroInfoEvent.PopBack -> popBack()
    }
}
```

Студенческая
ИТ-лаборатория

effectiveband

Навигация Compose

Навигация

effectiveband

- Навигация представлена в виде графа
- Хранение в экранов стеке
- NavHost и NavController

NavHost

effectiveband

```
@Serializable
object Profile
@Serializable
object FriendsList

@Composable
fun MyAppNavHost(
    modifier: Modifier = Modifier,
    navController: NavHostController = rememberNavController(),

) {
    NavHost(
        modifier = modifier,
        navController = navController,
        startDestination = Profile
    ) {
        composable<Profile> {
            ProfileScreen(
                onNavigateToFriends = { navController.navigate(route = FriendsList) },
                /*...*/
            )
        }
        composable<FriendsList> { FriendsListScreen(/*...*/) }
    }
}

@Composable
fun ProfileScreen(
    onNavigateToFriends: () -> Unit,
    /*...*/
) {
    /*...*/
    Button(onClick = onNavigateToFriends) {
        Text(text = "See friends list")
    }
}
```

Сущности для экранов

effectiveband

```
const val DETAIL_ARGUMENT_KEY = "id"

sealed class AppScreens(
    val route: String
) {
    object HeroesListScreen : AppScreens( route: "HeroesListScreen")
    object HeroInfoScreen : AppScreens( route: "HeroInfoScreen")
}
```

А где создаем?

effectiveband

```
@AndroidEntryPoint
class MainActivity : ComponentActivity() {
    @Inject
    lateinit var navigation: AppNavigation

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        WindowCompat.setDecorFitsSystemWindows(window, decorFitsSystemWindows: false)
        setContent {
            val navController = rememberNavController()
            navigation.navHostController = navController
            EffectiveLabsTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.secondary
                ) {
                    AppNavHost(navController = navController)
                }
            }
        }
    }
}
```

Красивый NavHost

effectiveband

```
@Composable
fun AppNavHost(navController: NavHostController) {

    NavHost(
        navController = navController,
        startDestination = AppScreens.HeroesListScreen.route
    ) { this: NavGraphBuilder
        composable(route = AppScreens.HeroesListScreen.route) { it: NavBackStackEntry
            val viewModel = hiltViewModel<HeroesListViewModel>()
            HeroesListScreen(viewModel)
        }
        composable(
            route = "${AppScreens.HeroInfoScreen.route}/{$DETAIL_ARGUMENT_KEY}",
            arguments = listOf(navArgument(DETAIL_ARGUMENT_KEY) { type = NavType.IntType }),
            deepLinks = listOf(
                navDeepLink { uriPattern = "$uri/{$DETAIL_ARGUMENT_KEY}" }
            )
        ) { it: NavBackStackEntry
            val id = it.arguments?.getInt(DETAIL_ARGUMENT_KEY)!!.toInt()
            val viewModel = hiltViewModel<HeroInfoViewModel>()
            HeroInfoScreen(viewModel, id = id)
        }
    }
}
```

Студенческая
ИТ-лаборатория

effectiveband

Пример проекта

Студенческая
ИТ-лаборатория

effectiveband

еще примеров

XML

Compose

Студенческая
ИТ-лаборатория

effectiveband

Возникают трудности в создании?
А как сделать это, а как то?

Студенческая
ИТ-лаборатория

effectiveband

Гуглим:)

Студенческая
ИТ-лаборатория

effectiveband

Лабы от Google - лучший старт

Пробежались по

effectiveband

- Чистая архитектура в Android
- Актуальный подход к созданию UI
- Создание приложение
- Актуальные подходы
- Реализация навигации

Студенческая
ИТ-лаборатория

effectiveband

Спасибо за внимание

Студенческая
ИТ-лаборатория

effectiveband

Спасибо за внимание