



Студенческая
ИТ-лаборатория

Основы разработки Android-приложений. UI и Навигация

Александр Григорьев

Android-разработчик в компании Effective



Почему Android?

Kotlin Multiplatform



**Compose
Multiplatform**

План



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

Архитектура и зачем она нужна?

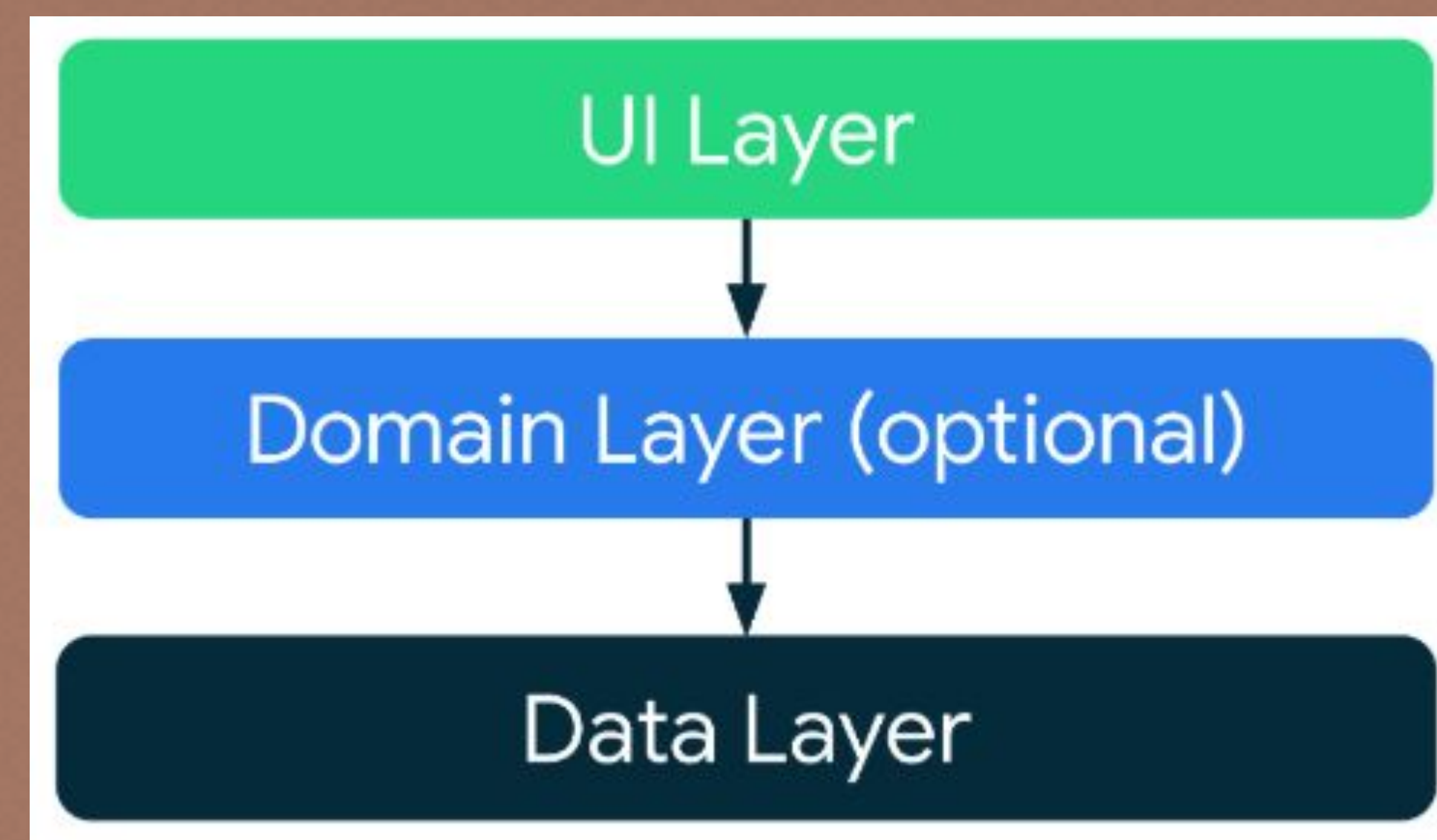
Чистая архитектура

Масштабируемость

Тестирование

Понимание кода

Мы экономим время!



UI-слой в Android

Подходы

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

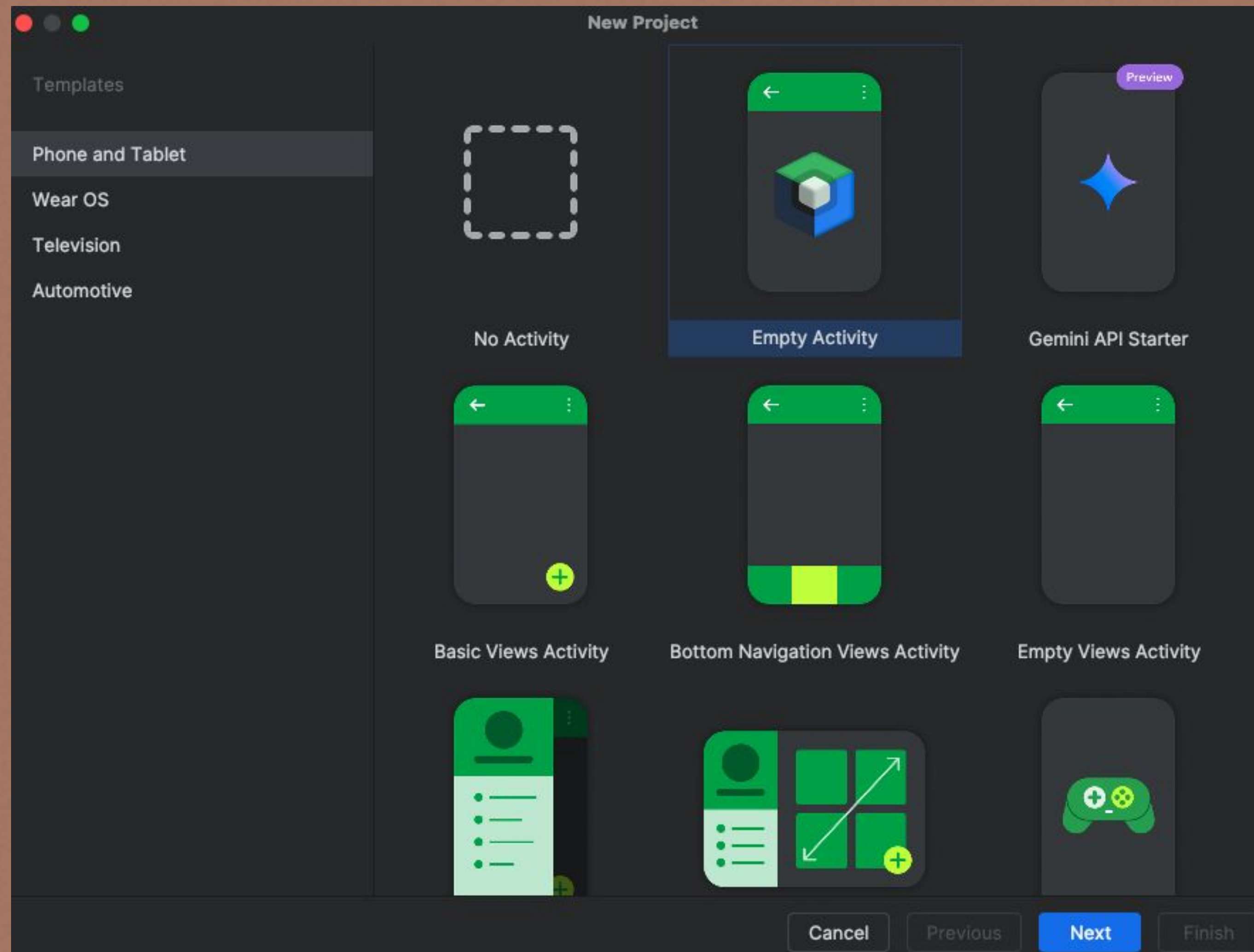
Compose или XML?

Compose



Время делать приложение

Создаем проект



Создаем проект

New Project

Empty Activity

Create a new empty activity with Jetpack Compose

Name

android_app

Package name

com.example.android_app

Save location

/Users/aleksandrgrigorev/AndroidStudioProjects/android_app

Minimum SDK

API 24 ("Nougat"; Android 7.0)

Your app will run on approximately 97,4% of devices.

Help me choose

Build configuration language ?

Kotlin DSL (build.gradle.kts) [Recommended]

The application name for most apps begins with an uppercase letter

Cancel

Previous

Next

Finish

MainActivity.kt x

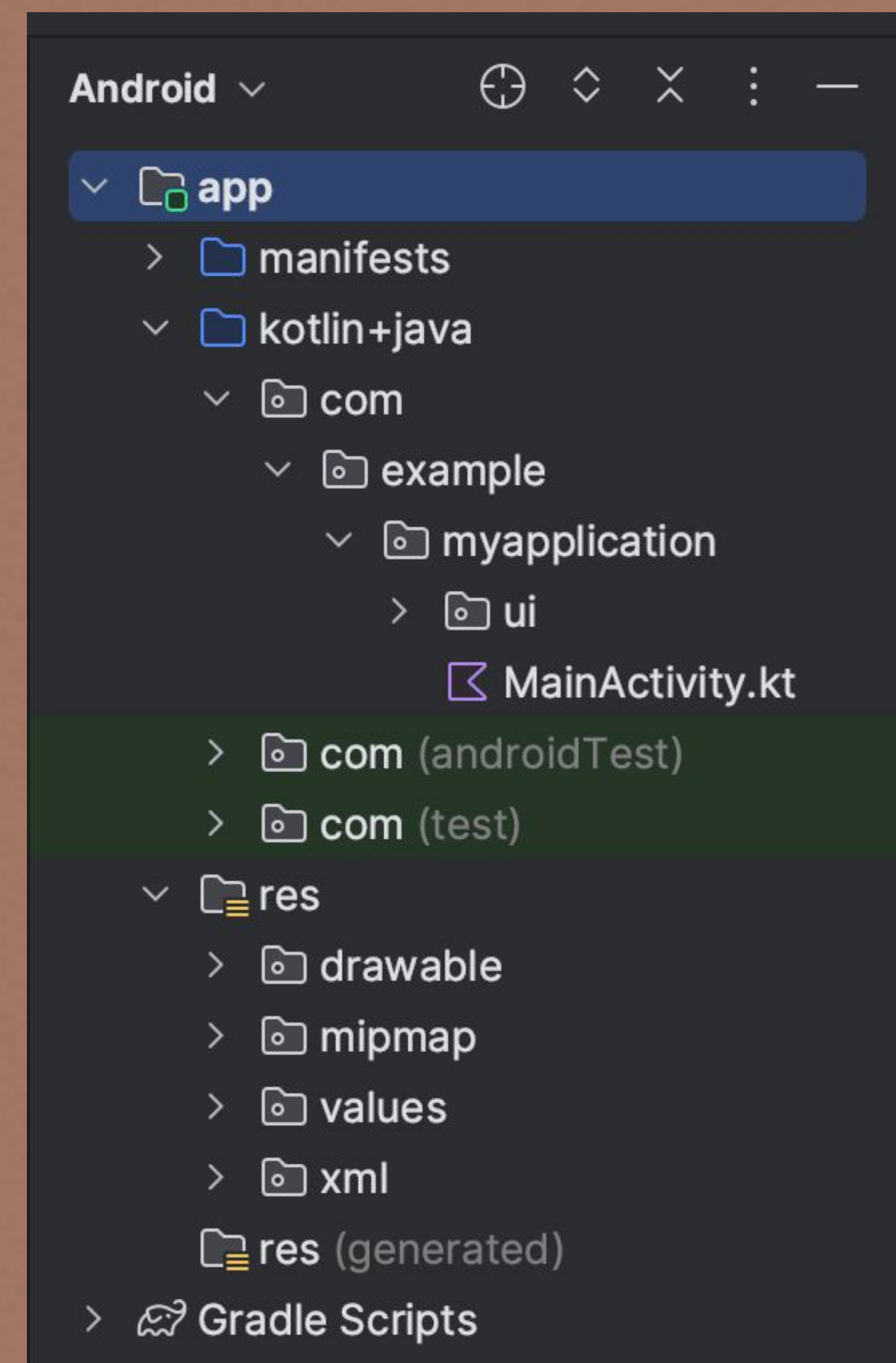
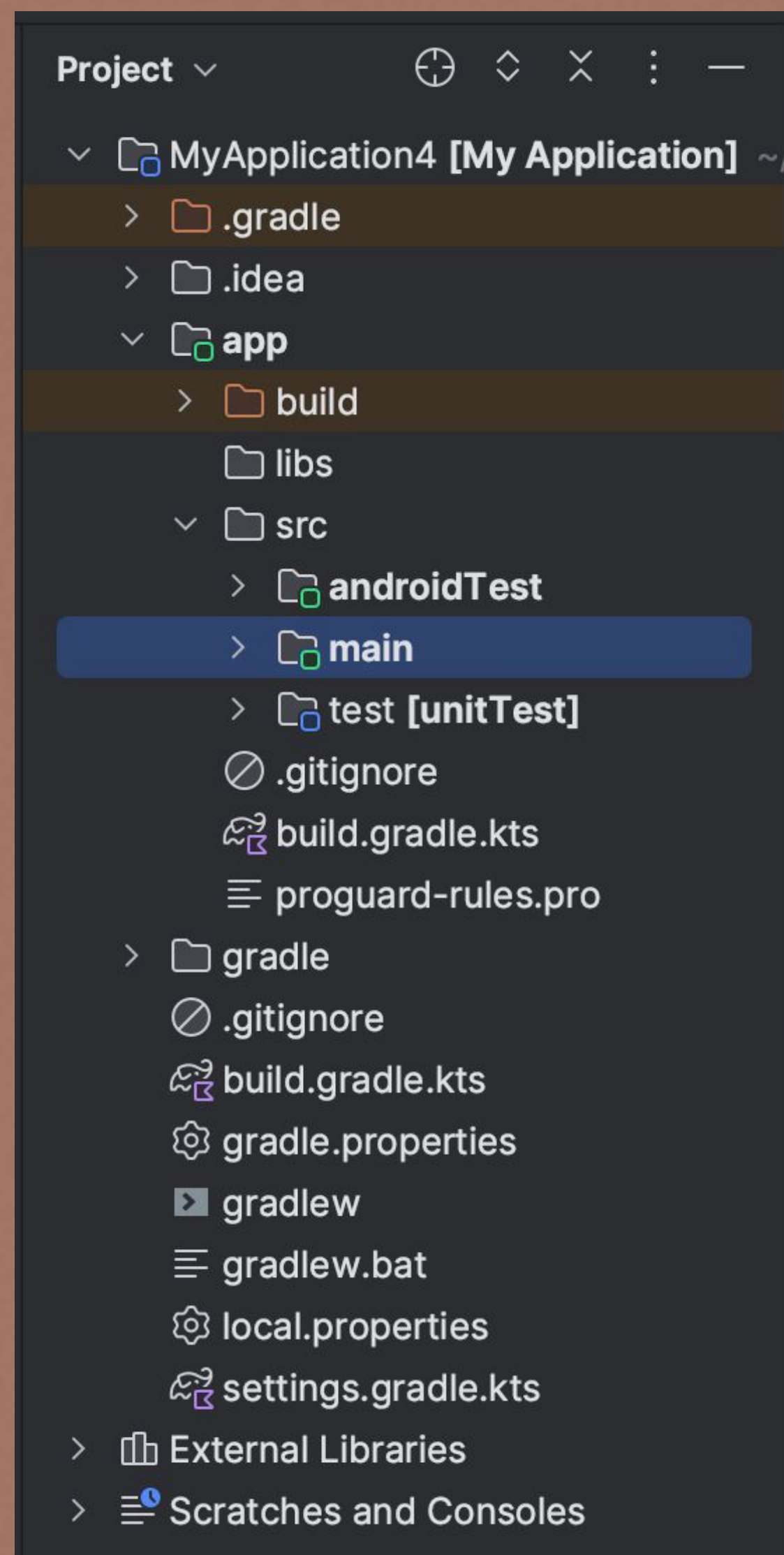
Gradle project sync in progress...

1

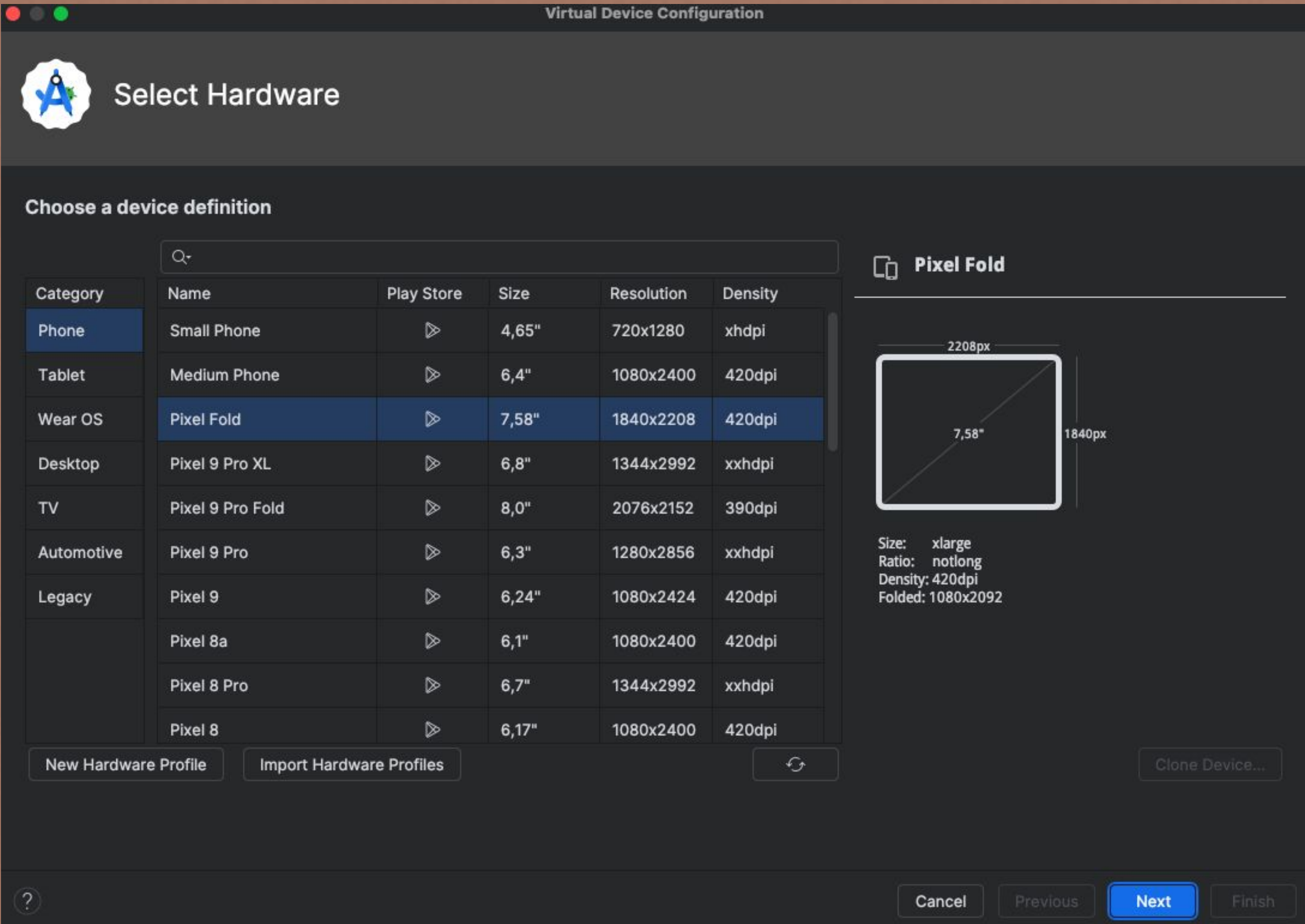
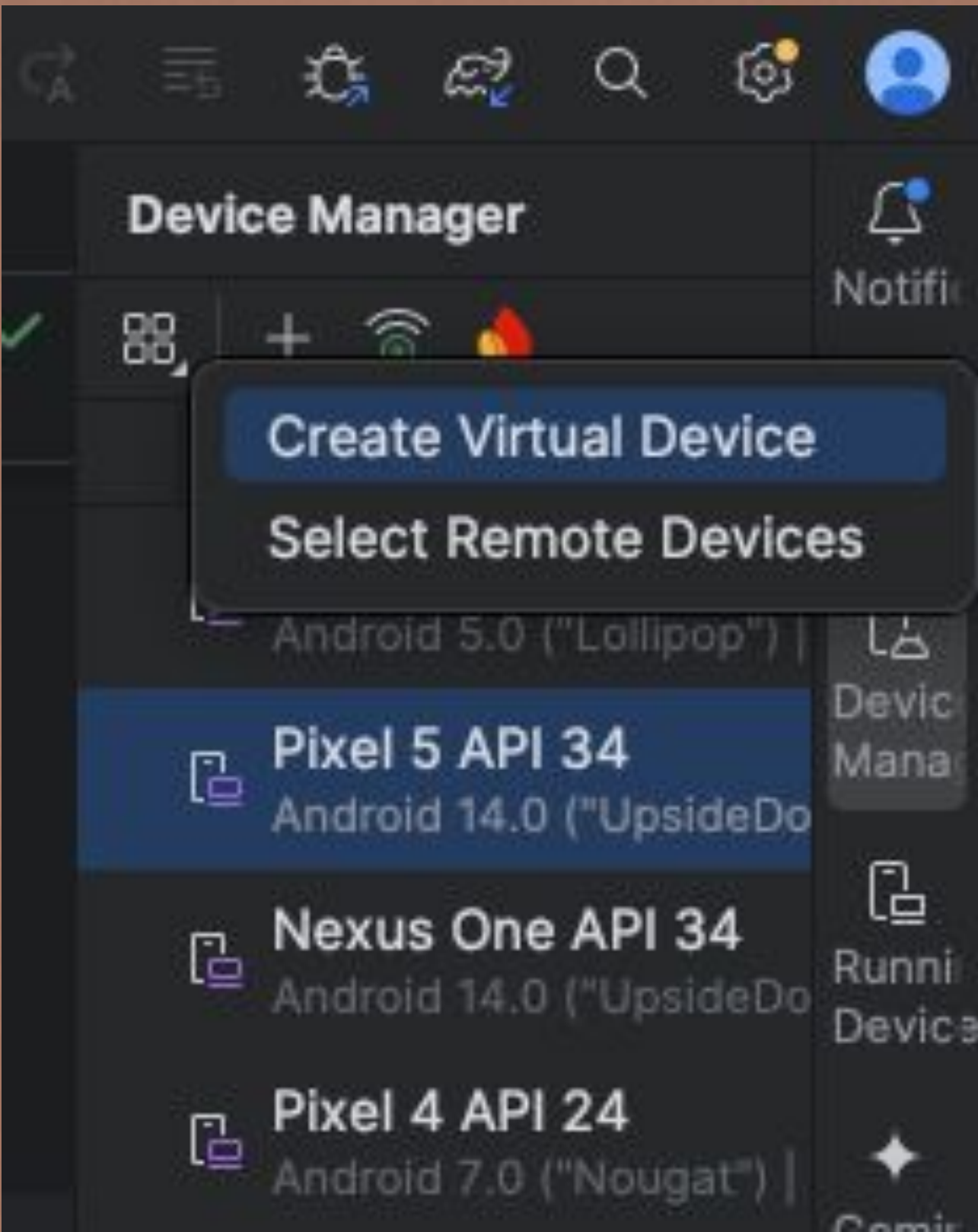
package com.example.android_app

2

Структура проекта



Подключаем эмулятор



Подключаем эмулятор



System Image

Select a system image

RecommendedARM Images

| Release Name | API |
|-----------------|-----------|
| Baklava | Baklava |
| Baklava | Baklava |
| VanillaIceCream | 35 |
| VanillaIceCream | 35 |
| UpsideDownCake | 34 |
| Tiramisu | 33 |
| Sv2 | 32 |
| S | 31 |
| R | 30 |
| Q | 29 |
| Pie | 28 |

SDK Quickfix Installation

SDK Component Installer

Completing Requested Actions

SDK Path: /Users/aleksandrgrigorev/Library/Android/sdk

Packages to install: – Google Play ARM 64 v8a System Image (system-images;android-35;google_apis_playstore;arm64-v8a)

Preparing "Install Google Play ARM 64 v8a System Image API 35 (revision 9)".

Downloading https://dl.google.com/android/repository/sys-img/google_apis_playstore/arm64-v8a-35_r09.zip

Starting download...

google.com/android/repository/sys-img/google_apis_playstore/arm64-v8a-35_r09.zip

Please wait until the requested actions are completed.

Cancel

Finish

?

Cancel

Previous

Next

Finish

System Image

Select a system image

RecommendedARM ImagesOther Images

| Release Name | API | ABI | xABI | Target |
|------------------------|-----------|------------------|------|---------------------------------------|
| Baklava | Baklava | arm64-v8a | | Android API Baklava (Google Play) |
| Baklava | Baklava | arm64-v8a | | Android API Baklava (16 KB Page Size) |
| VanillaIceCream | 35 | arm64-v8a | | Android 15.0 (Google Play) |
| VanillaIceCream | 35 | arm64-v8a | | Android 15.0 (16 KB Page Size) |
| UpsideDownCake | 34 | arm64-v8a | | Android 14.0 (Google Play) |
| Tiramisu | 33 | arm64-v8a | | Android 13.0 (Google Play) |
| Sv2 | 32 | arm64-v8a | | Android 12L (Google Play) |
| S | 31 | arm64-v8a | | Android 12.0 (Google Play) |
| R | 30 | arm64-v8a | | Android 11.0 (Google Play) |
| Q | 29 | arm64-v8a | | Android 10.0 (Google Play) |
| Pie | 28 | arm64-v8a | | Android 9.0 (Google ARM64-v8a) |

VanillaIceCream

API Level

35

Type

Google Play

Android

15.0

Google Inc.

System Image

arm64-v8a

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?

See the [API level distribution chart](#)

Refresh

?


Cancel

Previous

Next


Finish

Подключаем эмулятор

 Android Virtual Device (AVD)


Verify Configuration

AVD name:

 Pixel 9

6.24 1080x2424 420dpi

Change...


 VanillaIceCream


Android 15.0 arm64-v8a

Change...

Preferred ABI: Optimal

Startup orientation:

 Portrait

 Landscape

Emulated Performance

Graphics: Automatic

Device Frame ☒ Enable device frame

Show Advanced Settings

AVD Name

The name of this AVD.

?

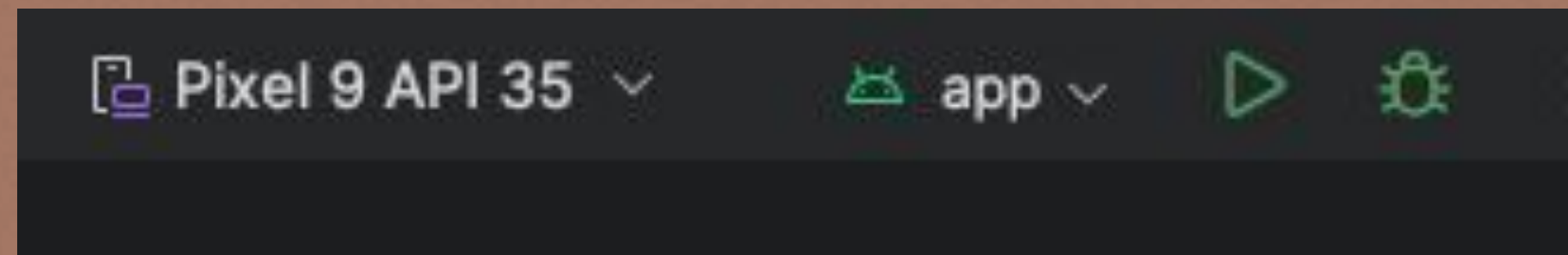
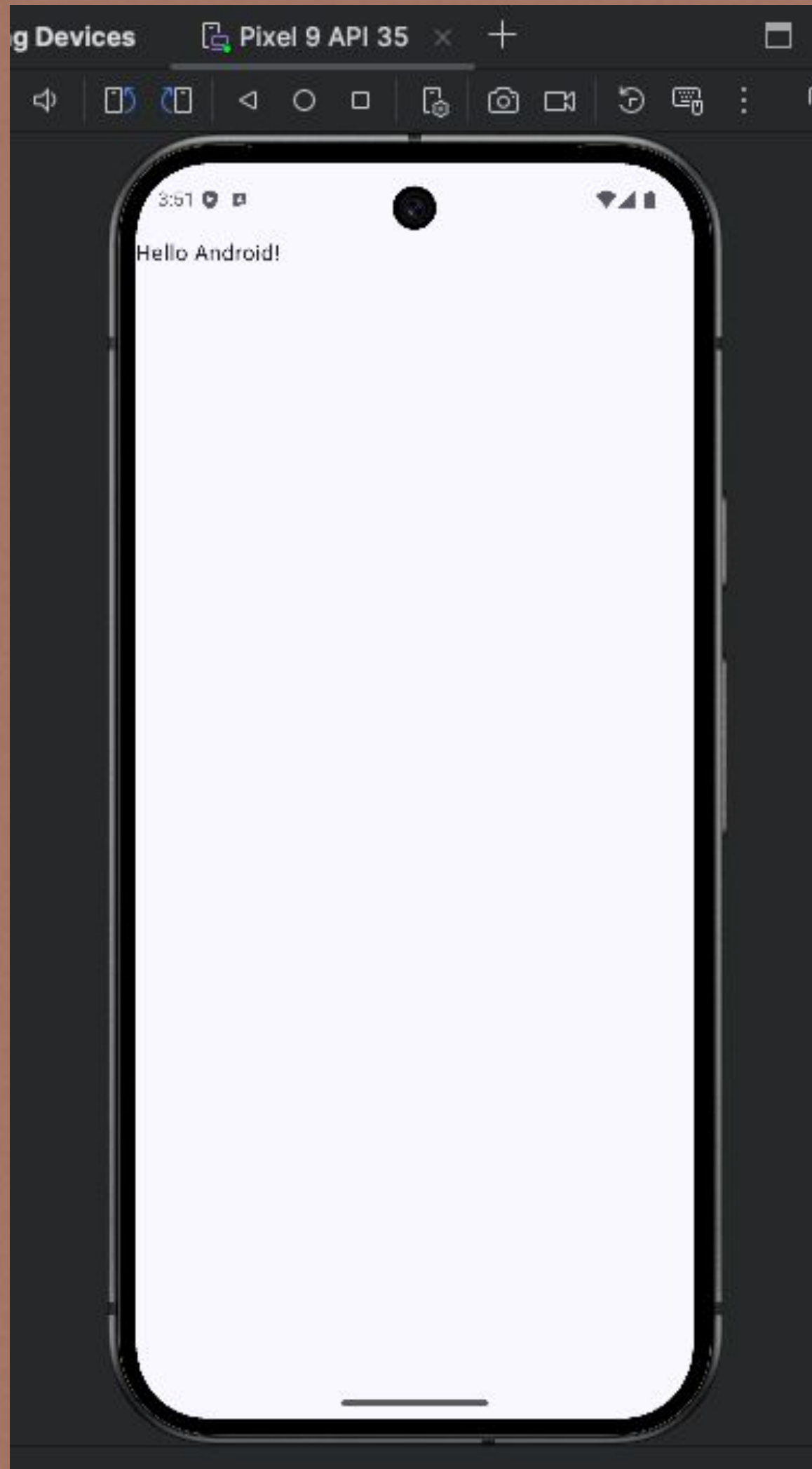
Cancel

Previous

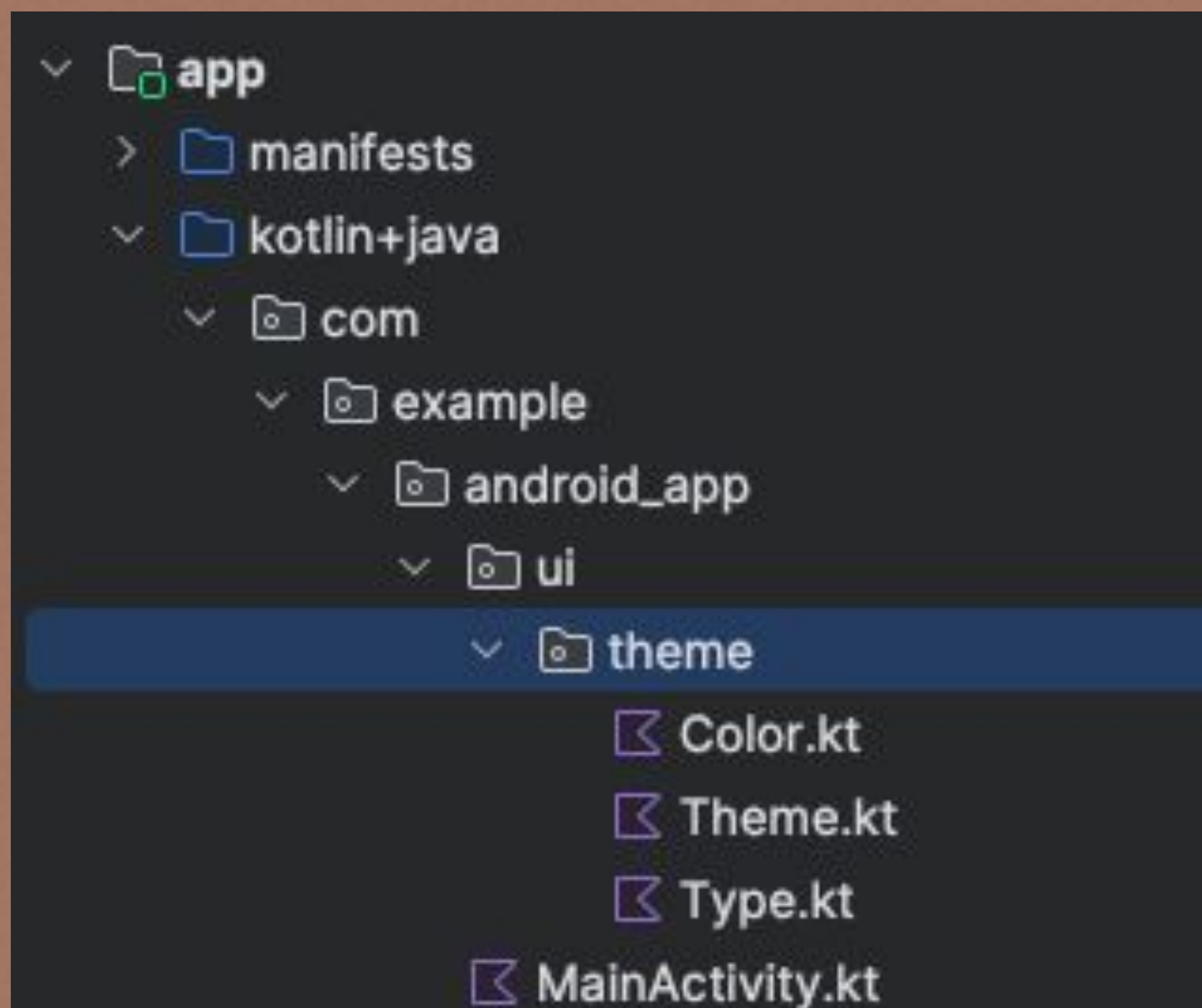
Next

Finish

Подключаем эмулятор



Что имеем?



```
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?

- 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?

```
@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

```
@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
            )
        }
    }
}
```

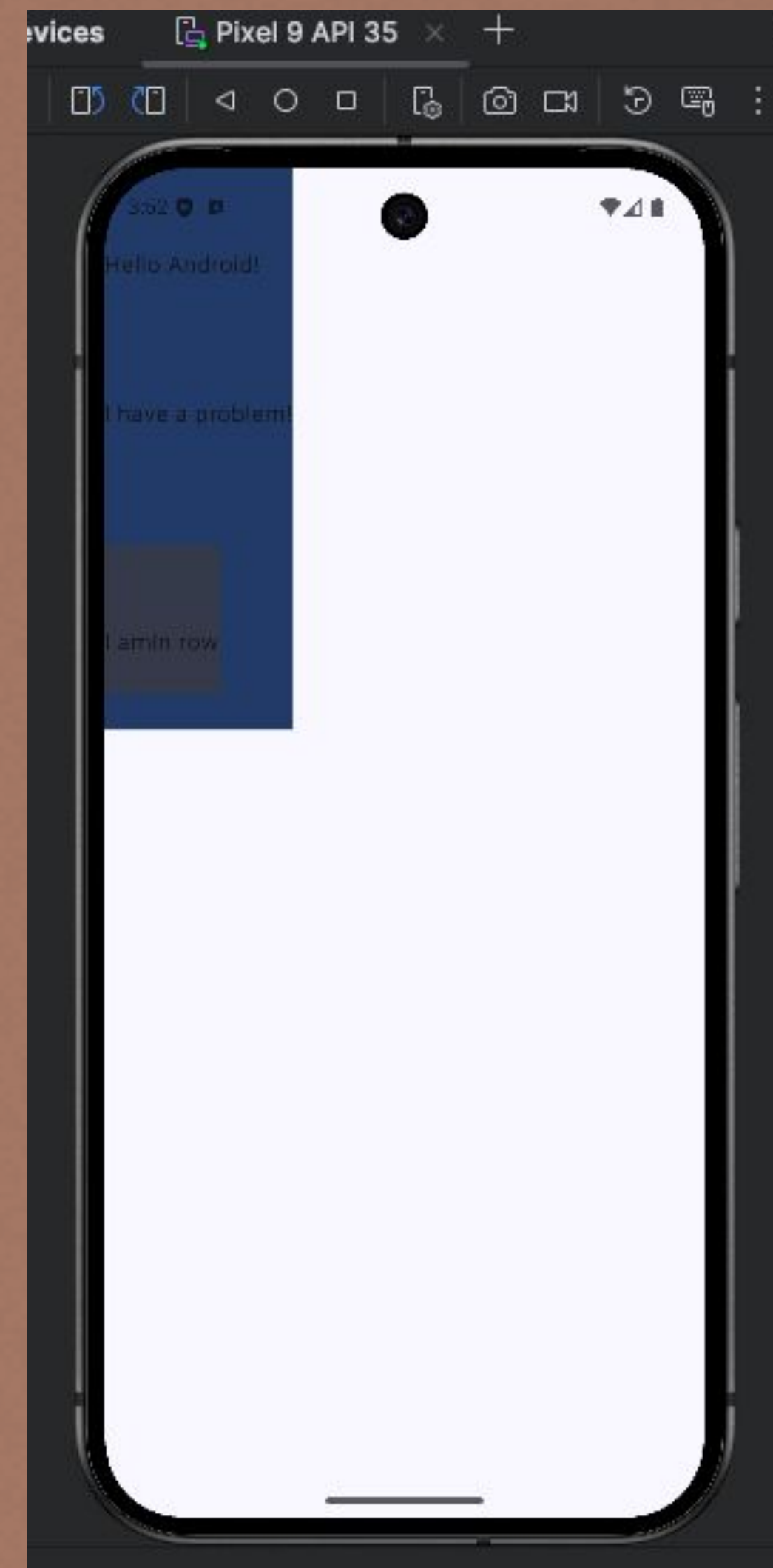
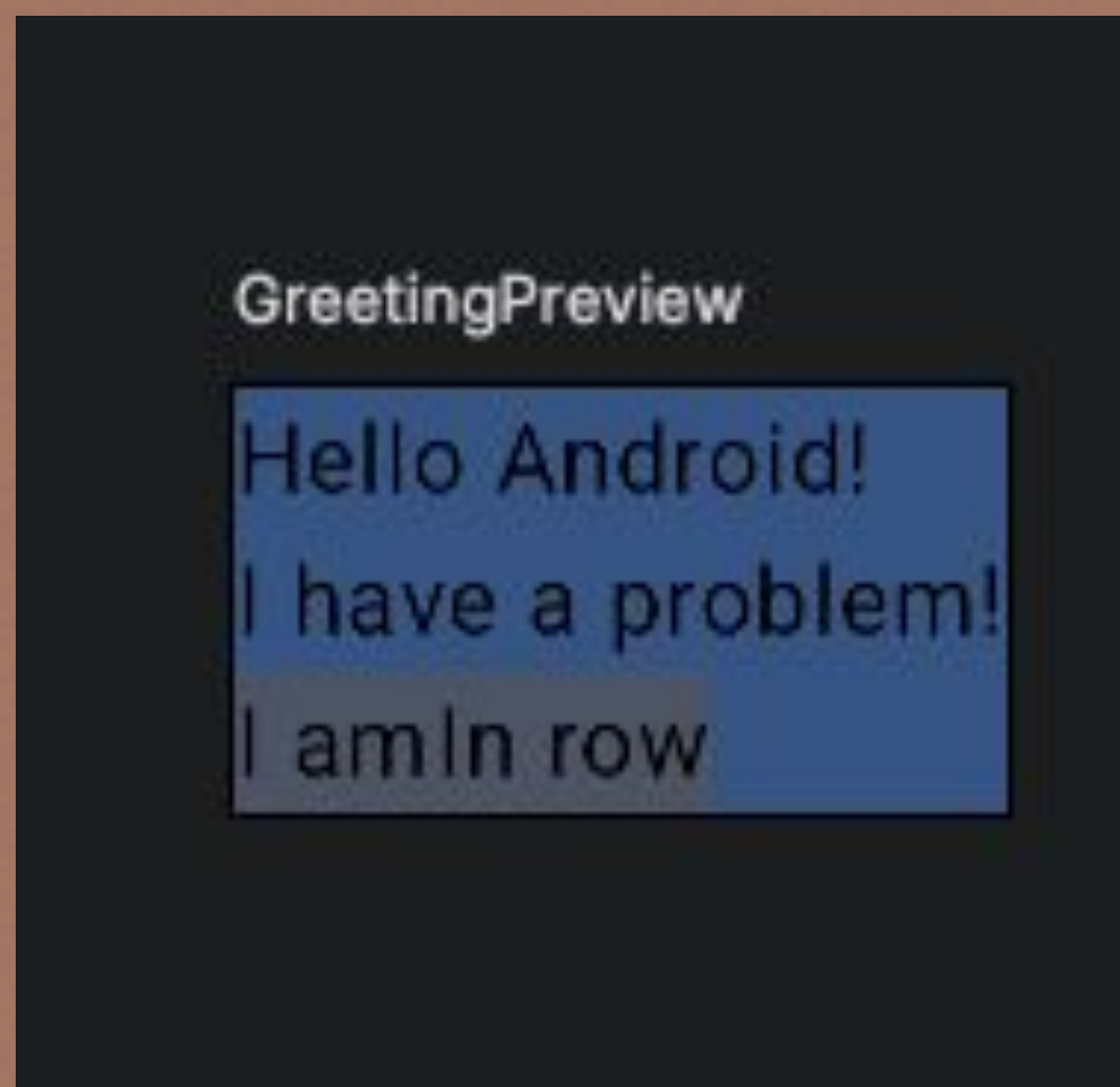
```
62
63  @Preview(showBackground = true)
64  @Composable
65  fun GreetingPreview() {
66      Greeting("Effective Band")
67  }
68
69
```

Run 'GreetingPreview' [^]_⬆R

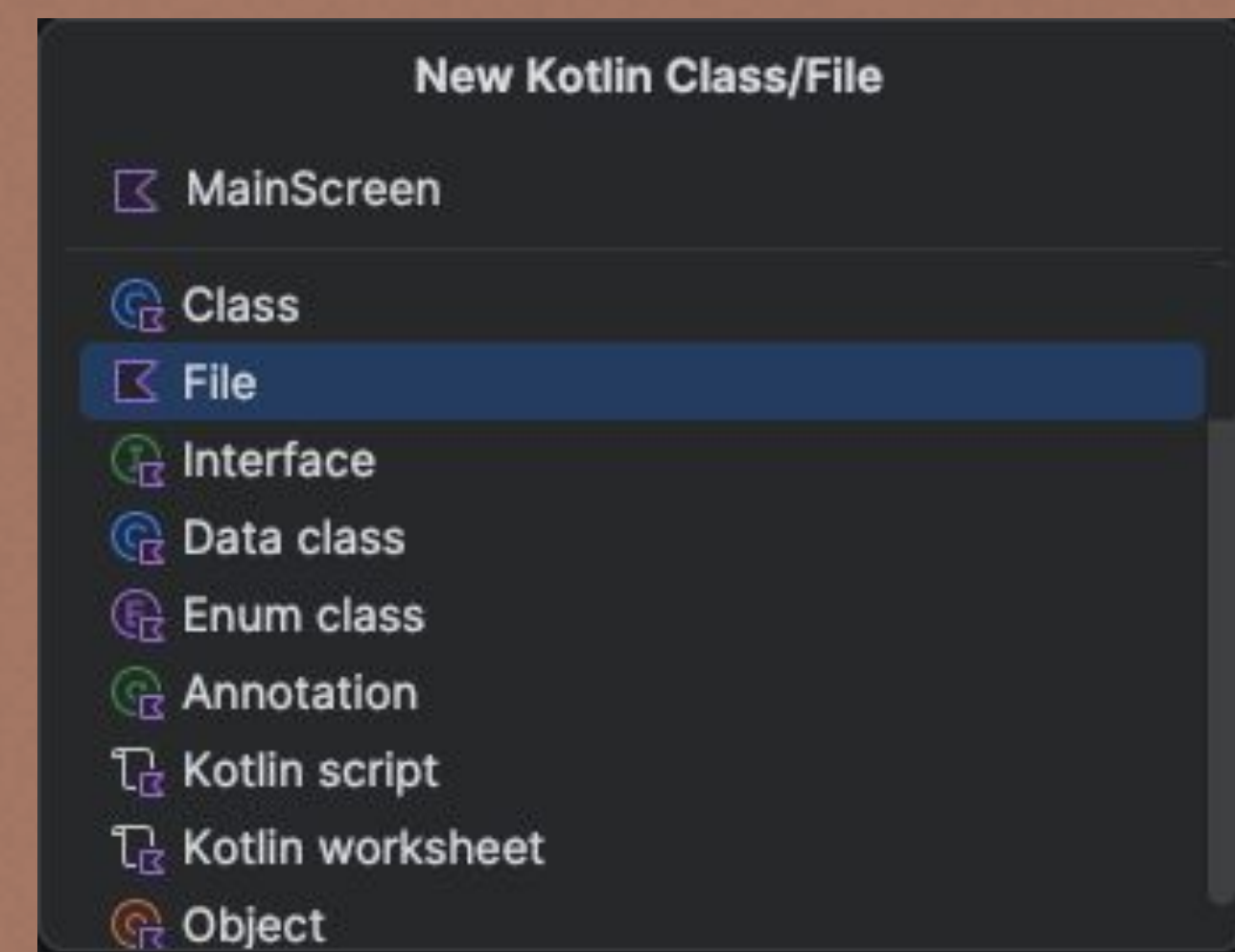
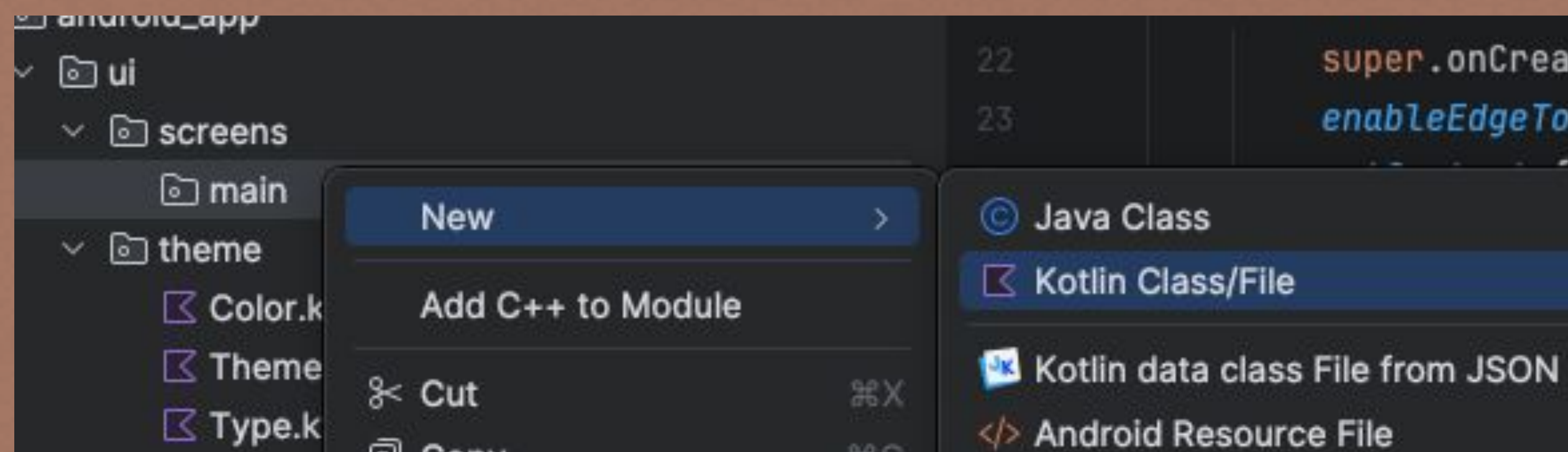
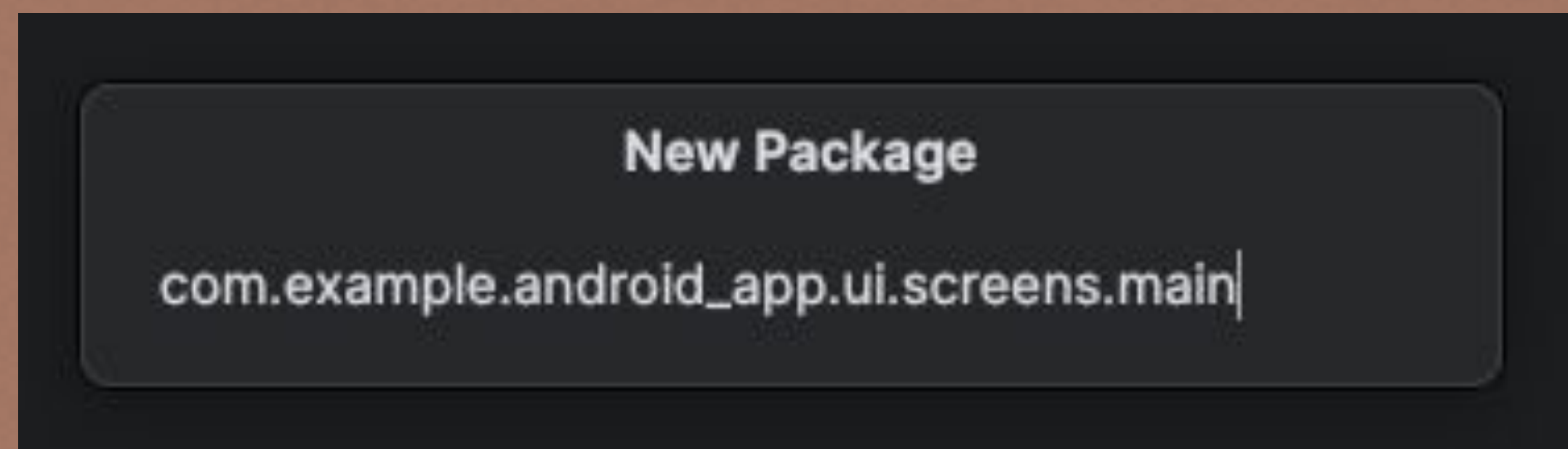
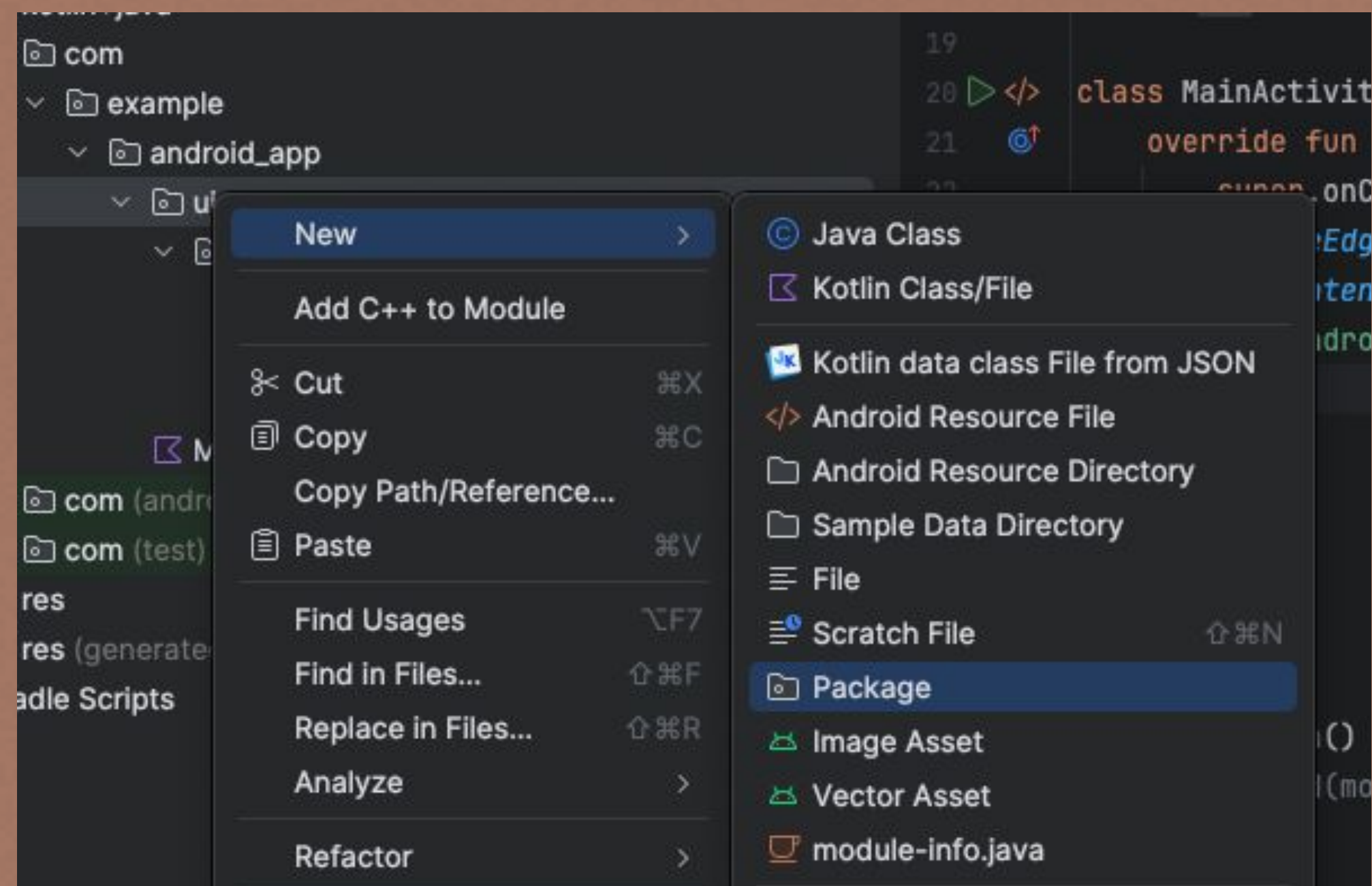
Debug 'GreetingPreview' [^]_⬆D

Modify Run Configuration...

Играемся с версткой



Создаем экран



Создаем экран

```
@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

```
@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

```
@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

```
@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

```
@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

```

└─ ui
  └─ consts
  └─ presentation
    └─ models
    └─ screens
      └─ components
        └─ ErrorView.kt
        └─ LoadingView.kt
        └─ TransparentSystemBars.kt
      └─ heroesList
        └─ components
        └─ models
        └─ HeroesListScreen.kt
      └─ heroInfo
        └─ components
        └─ models
        └─ HeroInfoScreen.kt
    └─ utils
    └─ theme
  └─ utils
  └─ App
  └─ MainActivity
```

```

└─ data
└─ di
└─ navigation
  └─ AppNavHost.kt
  └─ AppNavigation
  └─ AppNavigationImpl
  └─ AppScreens.kt
```


А что делать дальше?



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

ViewModel

- 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()  
        }  
    }  
}
```

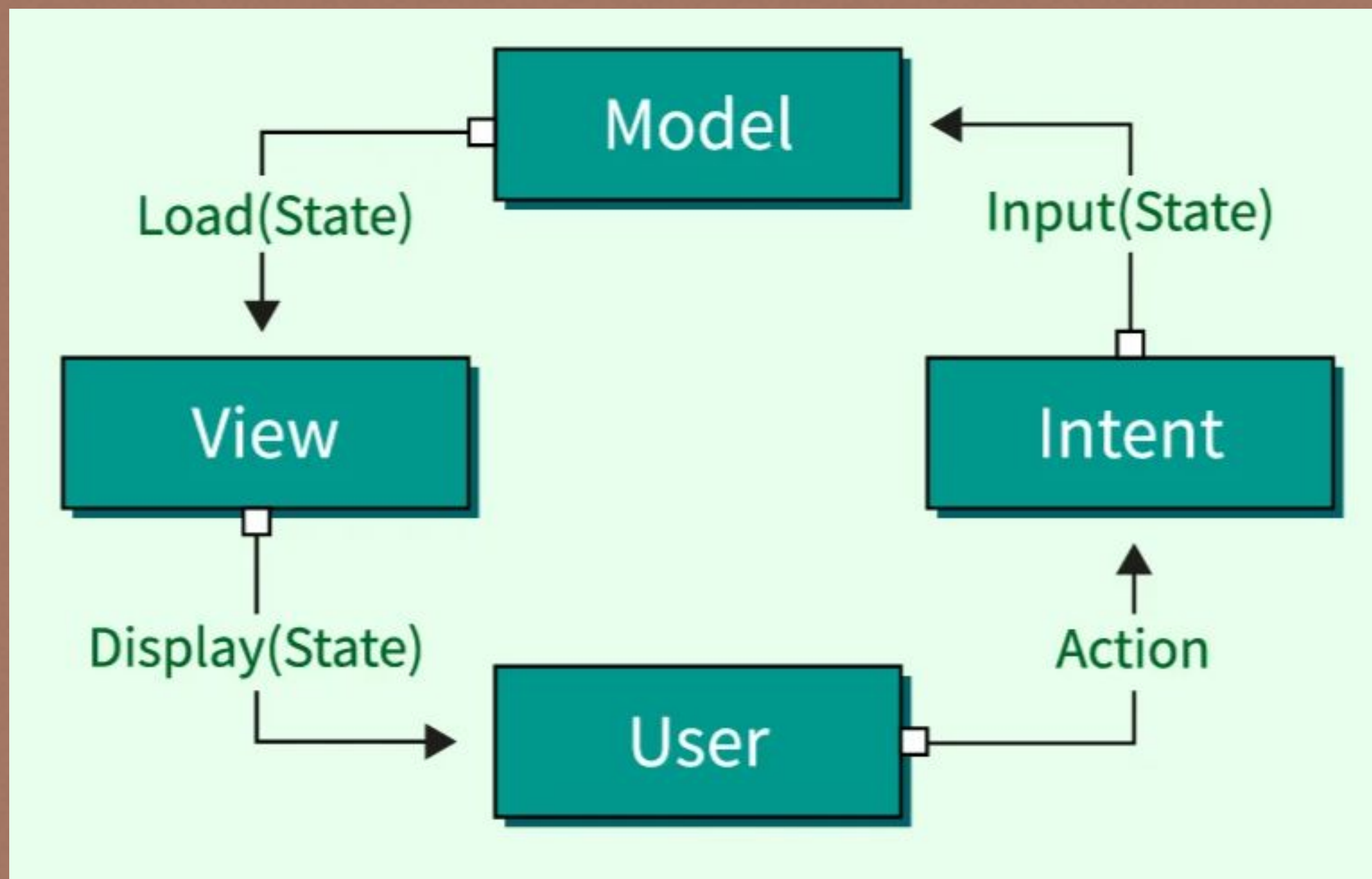


Паттерны

Паттерны



- MVC
- MVP
- MVVM
- MVI



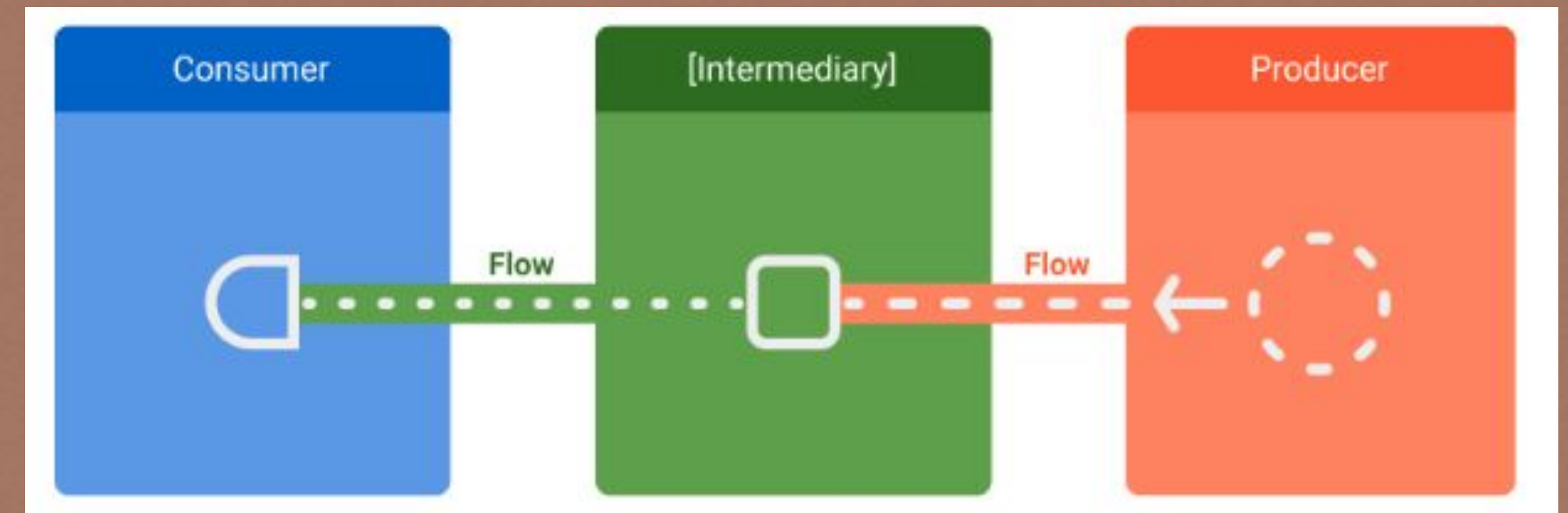
Состояние экрана



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

Flow

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



Состояние экрана



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



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()
    }
}
```


Навигация Compose



Навигация



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

NavHost



```
@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

```
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?

```
@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

```
@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)
        }
    }
}
```


Пример проекта



еще примеров



XML



Compose

**Возникают трудности в
создании?**

А как сделать это, а как то?

Гуглим:)

Лабь от Google - лучший старт



Пробежались по



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

Спасибо за внимание

Спасибо за внимание