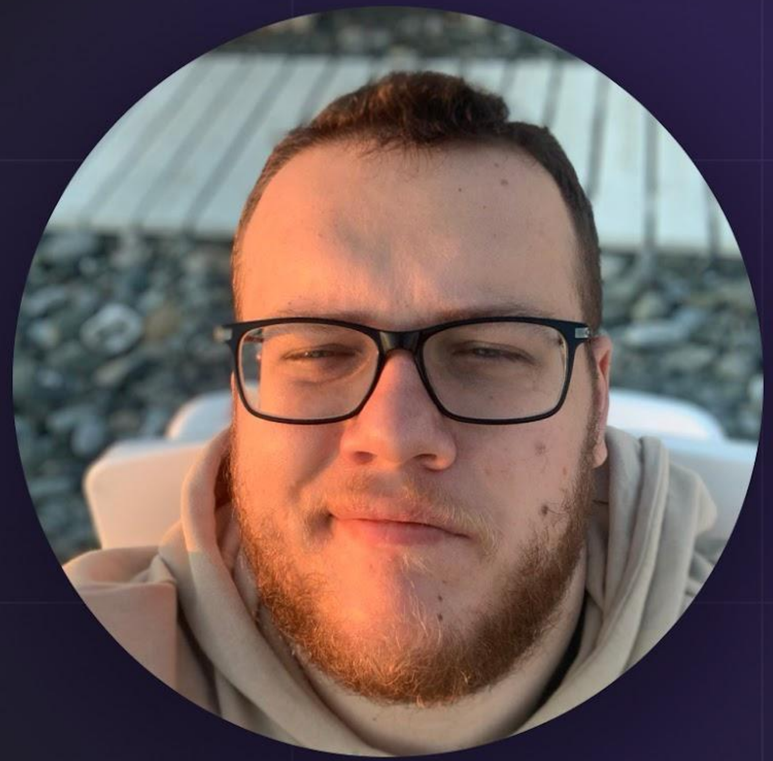
COMPOSE MULTIPLATFORM UX



АЛЕКСЕЙ ГЛАДКОВ

Mobile Developer

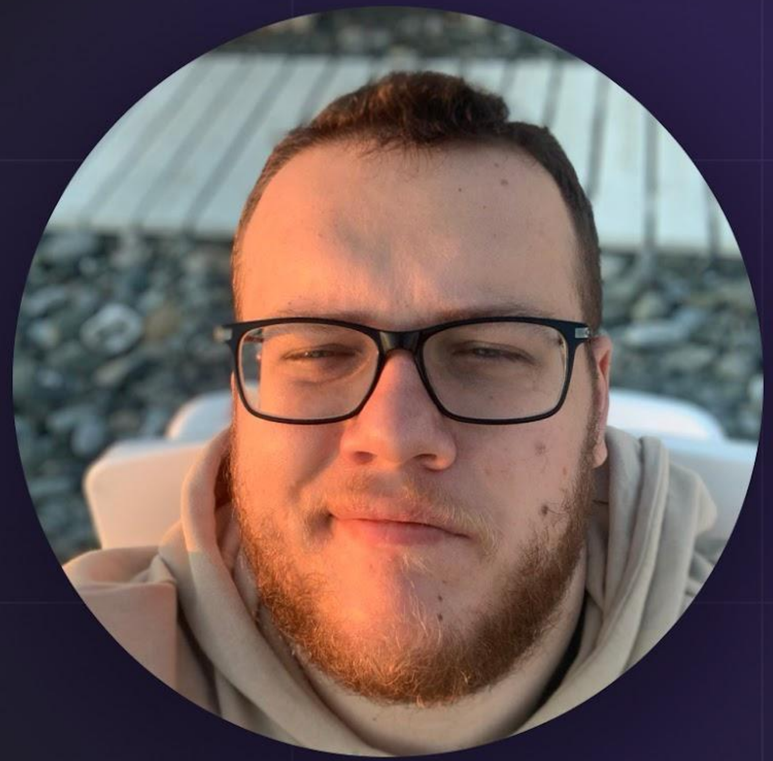
Mobiledeveloper

Mobiledeveloper

Mobiledevelopercourse@Gmail.Com

ОБО МНЕ

- ☑ Автор Mobile Developer
- ☑ Mobile Broadcast Expert
- ☑ Android и iOS разработчик > 11 лет



КОНТЕКСТ

- [x] Compose for iOS Alpha
- [ ] Compose for Web Dev Preview
- [ ] Очень сложный и не очевидный Gradle setup

КОНТЕКСТ

- [x] Compose for iOS Alpha
- [x] Compose for Web Dev Preview
- [ ] Очень сложный и не очевидный Gradle setup

КОНТЕКСТ

- [x] Compose for iOS Alpha
- [x] Compose for Web Dev Preview
- [x] Очень сложный и не очевидный Gradle setup

КОНТЕКСТ

- [x] Compose for iOS Beta
- [ ] Compose for Alpha
- [ ] Легкий setup, работа на всех платформах сразу

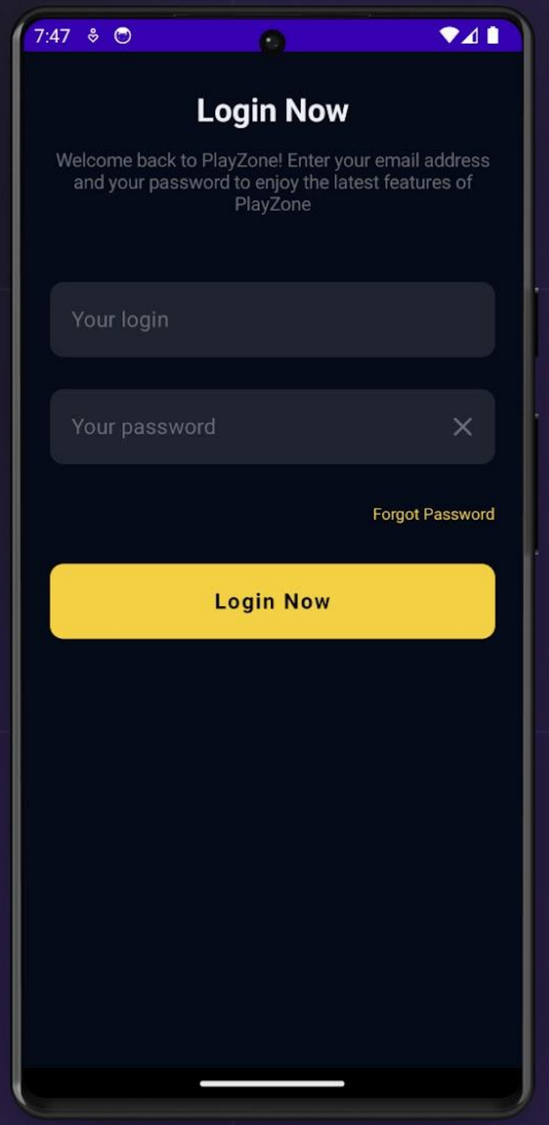
КОНТЕКСТ

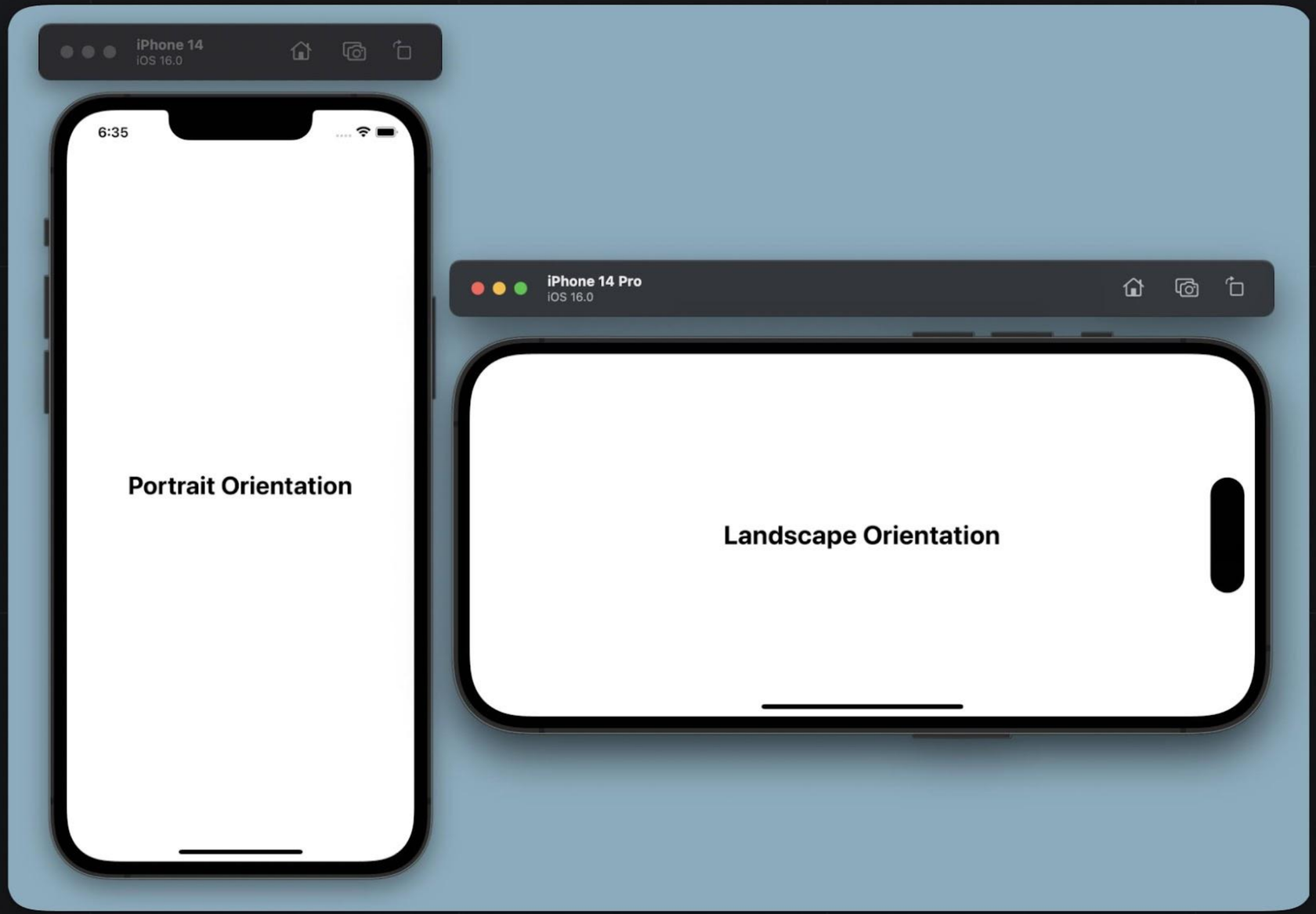
- [x] Compose for iOS Beta
- [x] Compose for Alpha
- [ ] Легкий setup, работа на всех платформах сразу

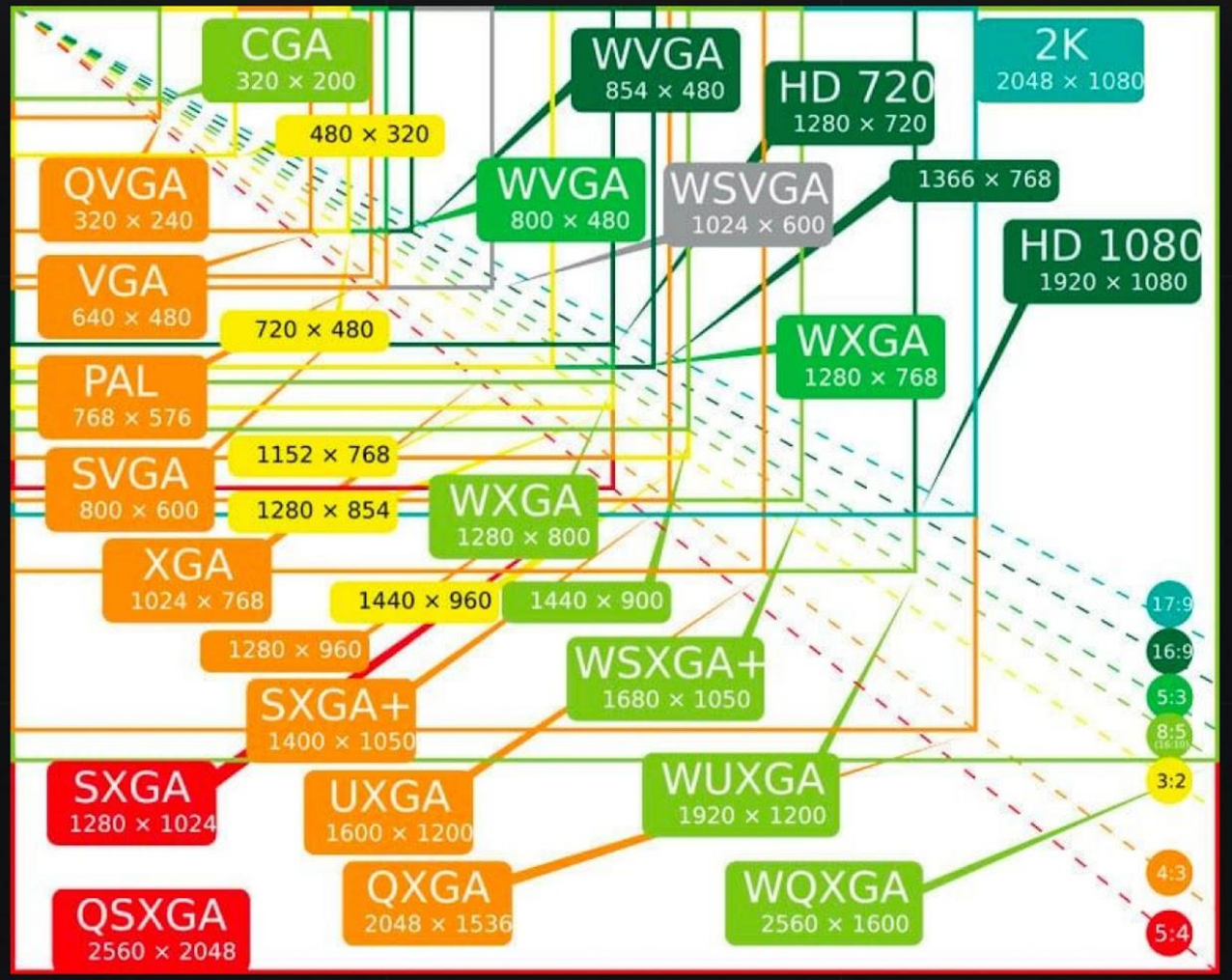
КОНТЕКСТ

- [x] Compose for iOS Beta
- [x] Compose for Alpha
- [x] Легкий setup, работа на всех платформах сразу

```
@Composable
fun LoginView(...) {
    Column(...) {
        Text(...)
        Text(...)
        Spacer(modifier = Modifier.height(50.dp))

        TextField(...)
        Spacer(modifier = Modifier.height(24.dp))

        TextField(...)
        Spacer(modifier = Modifier.height(30.dp))

        Row(...)
        Spacer(modifier = Modifier.height(30.dp))

        Button(...)
    }
```

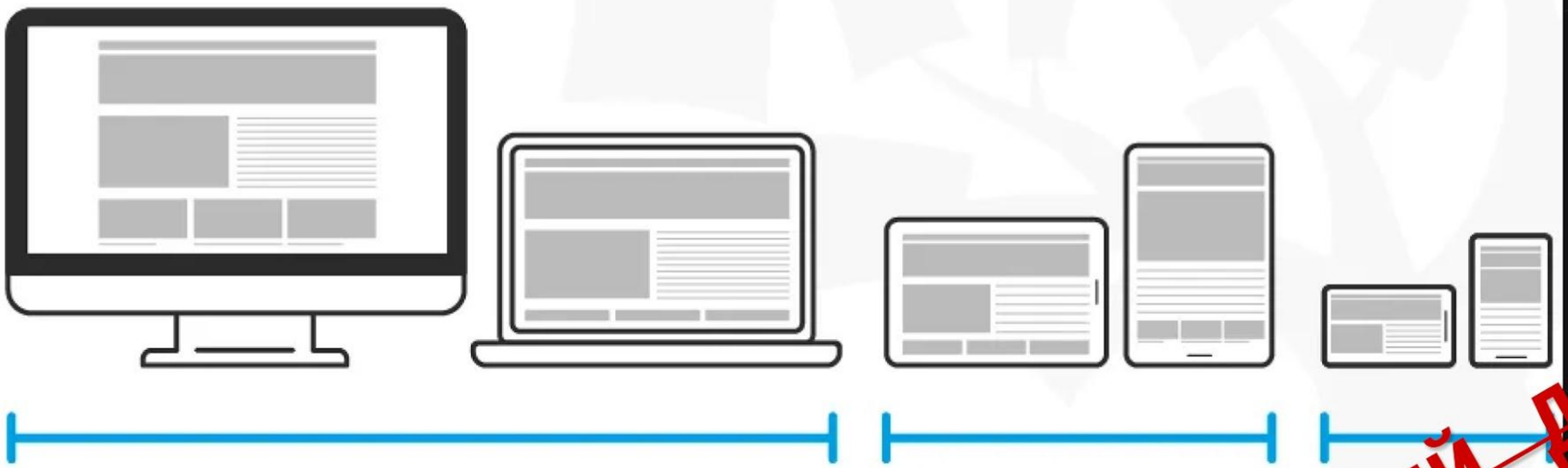



iPhone 14
iOS 16.0
6:35
Portrait Orientation
iPhone 14 Pro
iOS 16.0
Landscape Orientation

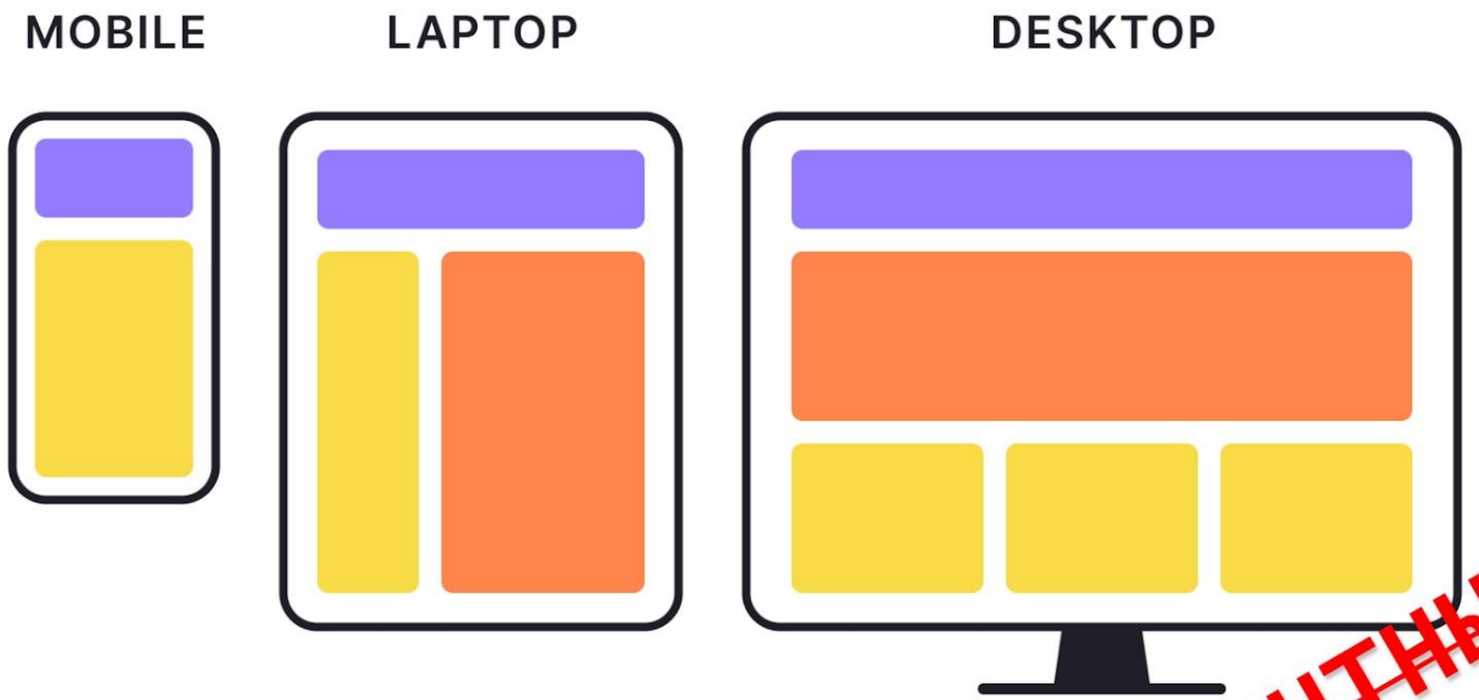

CGA
320 × 200
480 × 320
QVGA
320 × 240
VGA
640 × 480
720 × 480
PAL
768 × 576
SVGA
800 × 600
1152 × 768
1280 × 854
XGA
1024 × 768
1440 × 960
1440 × 900
1280 × 960
SXGA+
1400 × 1050
SXGA
1280 × 1024
UXGA
1600 × 1200
QSXGA
2560 × 2048
QXGA
2048 × 1536
WVGA
854 × 480
WVGA
800 × 480
WSVGA
1024 × 600
HD 720
1280 × 720
1366 × 768
WXGA
1280 × 800
WXGA
1280 × 768
WSXGA+
1680 × 1050
WUXGA
1920 × 1200
WQXGA
2560 × 1600
2K
2048 × 1080
HD 1080
1920 × 1080
17:9
16:9
5:3
8:5
(16:10)
3:2
4:3
5:4


FRONTENDERS
В ПЕРВЫЙ РАЗ?

Responsive Design



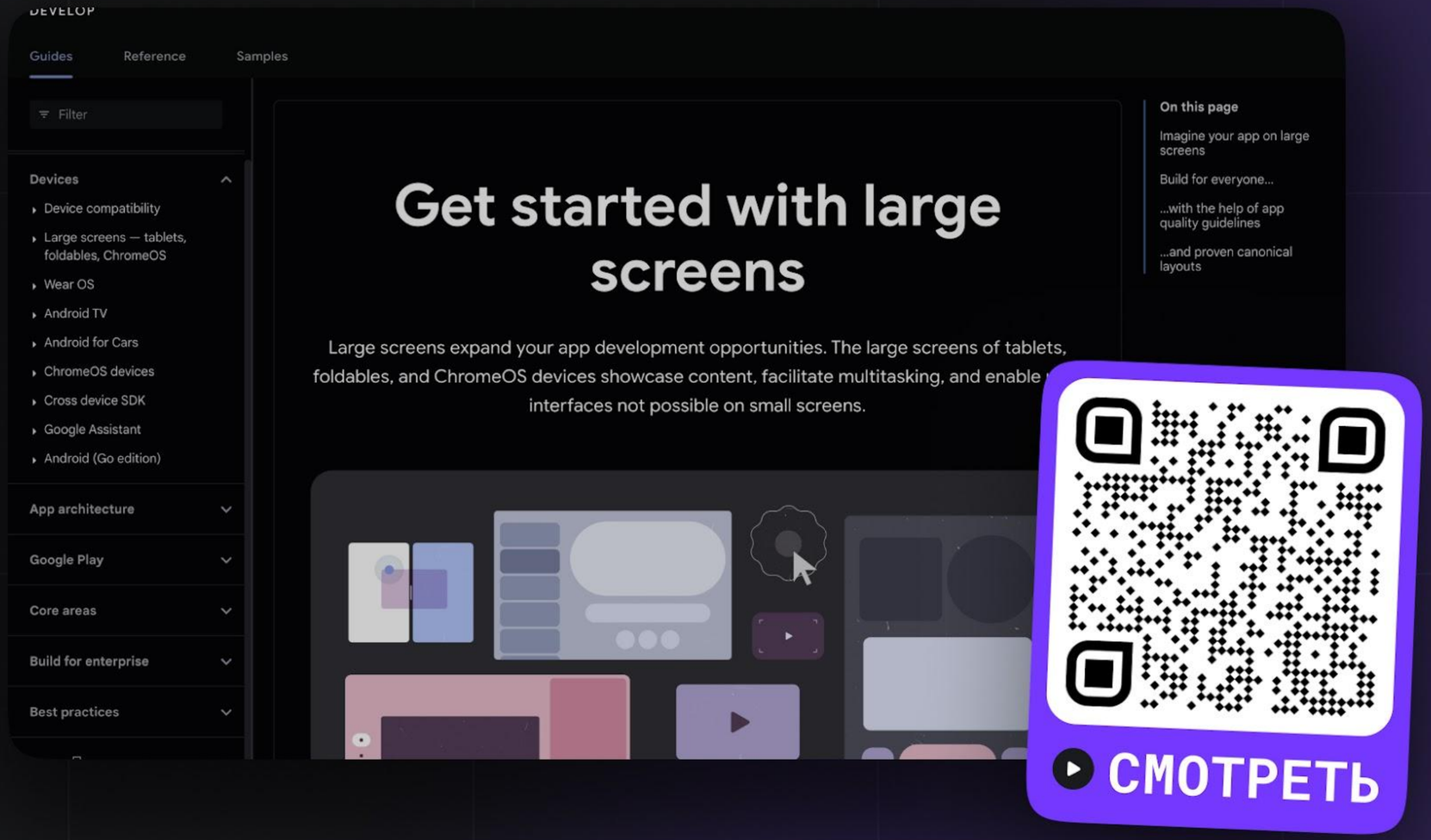
РЕЗИНОВЫЙ ДИЗАЙН

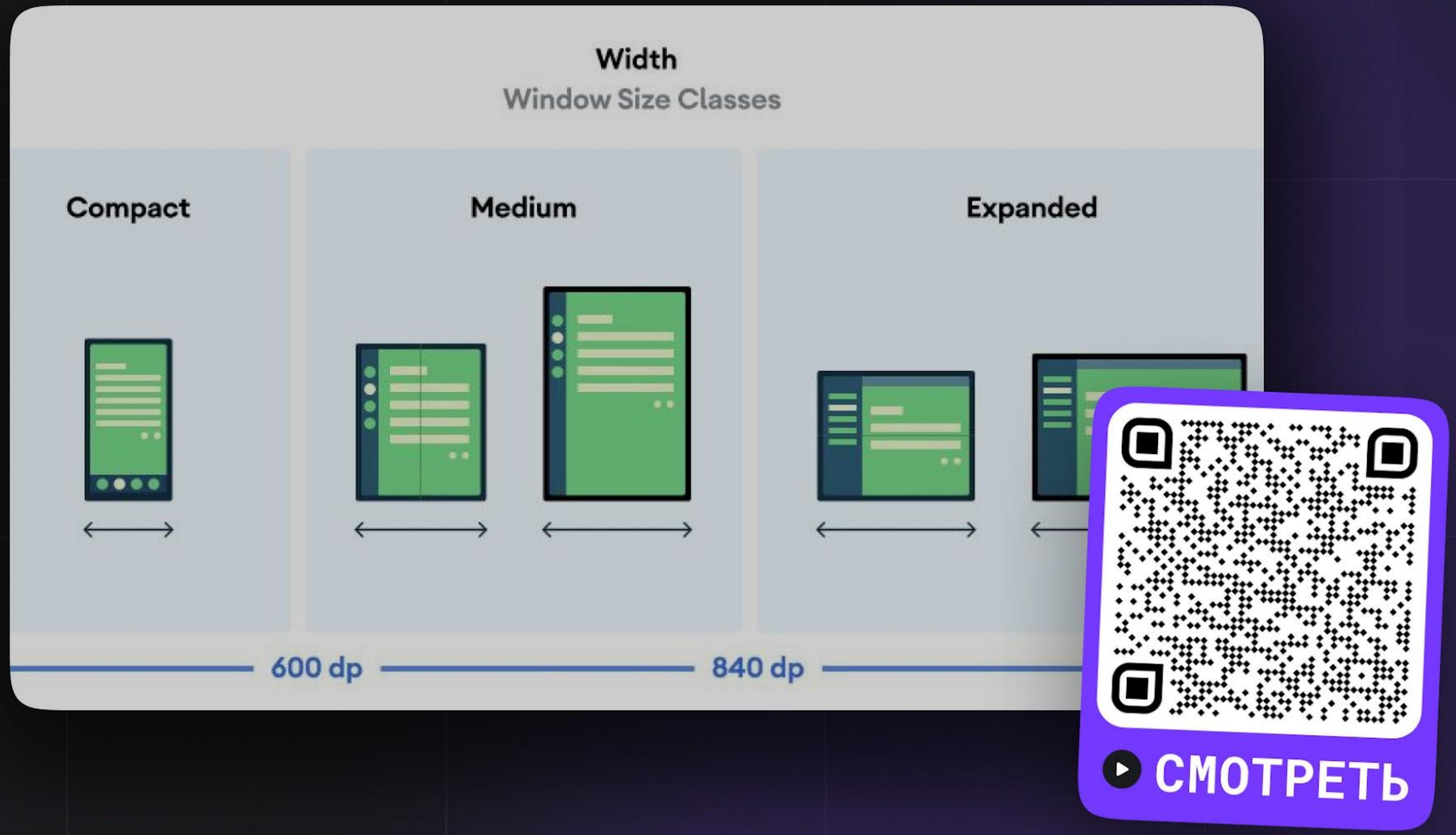


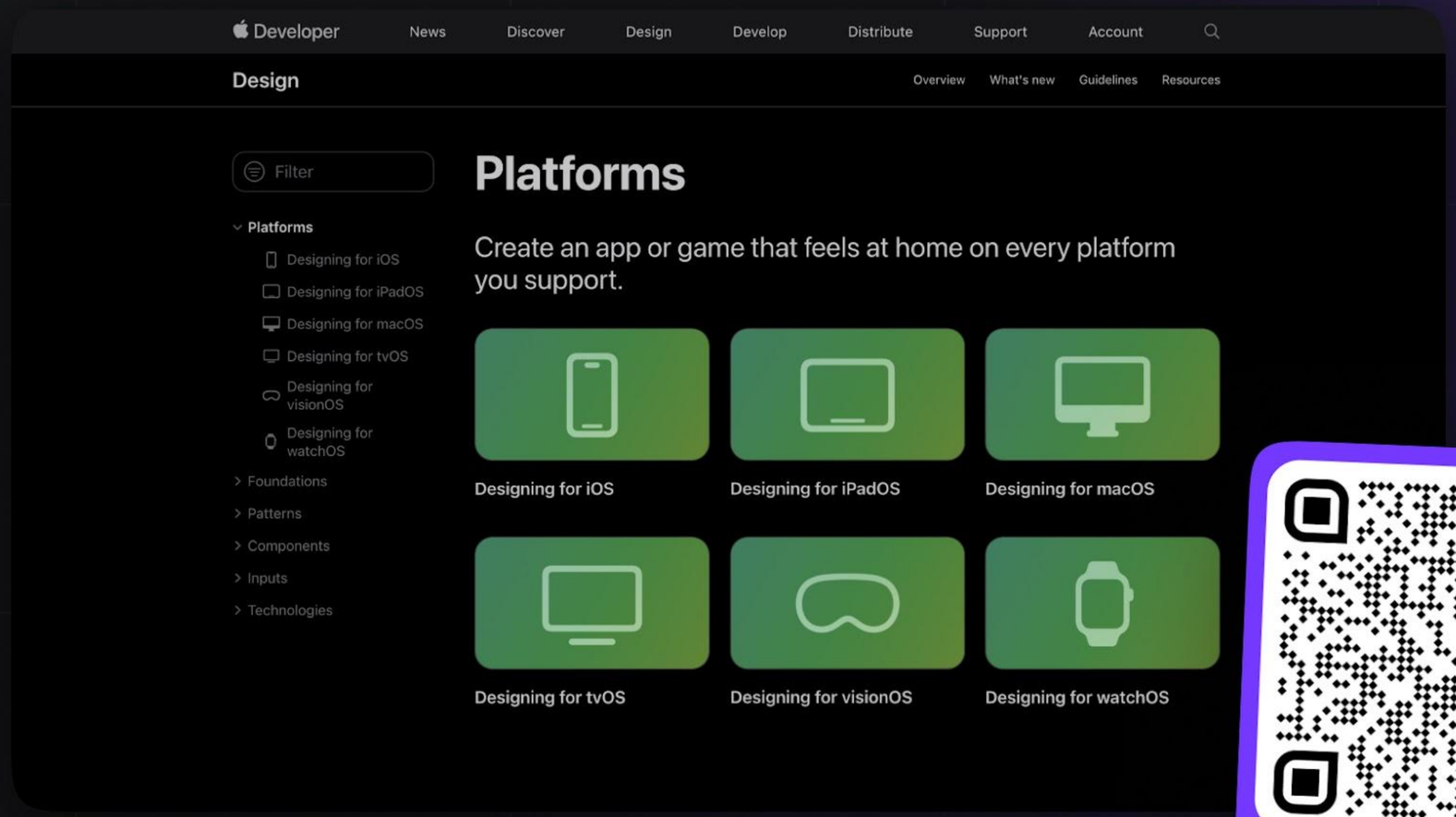
КОМПОНЕНТНЫЙ ДИЗАЙН


2 299.–
ЛЕРХАМН
СТУЛ
1 999.–
БЁРЬЕ
СТУЛ
Есть
два стула:
Теперь Вы можете
сесть на любой.
IKEA


НЕТ!
WEB


DEVELOP
Guides
Reference
Samples
Filter
Devices
Device compatibility
Large screens — tablets, foldables, ChromeOS
Wear OS
Android TV
Android for Cars
ChromeOS devices
Cross device SDK
Google Assistant
Android (Go edition)
App architecture
Google Play
Core areas
Build for enterprise
Best practices
Get started with large screens
Large screens expand your app development opportunities. The large screens of tablets, foldables, and ChromeOS devices showcase content, facilitate multitasking, and enable
interfaces not possible on small screens.
On this page
Imagine your app on large screens
Build for everyone...
...with the help of app quality guidelines
...and proven canonical layouts
СМОТРЕТЬ


Width
Window Size Classes
Compact
Medium
Expanded
600 dp
840 dp
СМОТРЕТЬ


Developer
News
Discover
Design
Develop
Distribute
Support
Account
Design
Overview
What's new
Guidelines
Resources
Filter
Platforms
Designing for iOS
Designing for iPadOS
Designing for macOS
Designing for tvOS
Designing for visionOS
Designing for watchOS
Foundations
Patterns
Components
Inputs
Technologies
Platforms
Create an app or game that feels at home on every platform you support.
Designing for iOS
Designing for iPadOS
Designing for macOS
Designing for tvOS
Designing for visionOS
Designing for watchOS

СМОТРЕТЬ

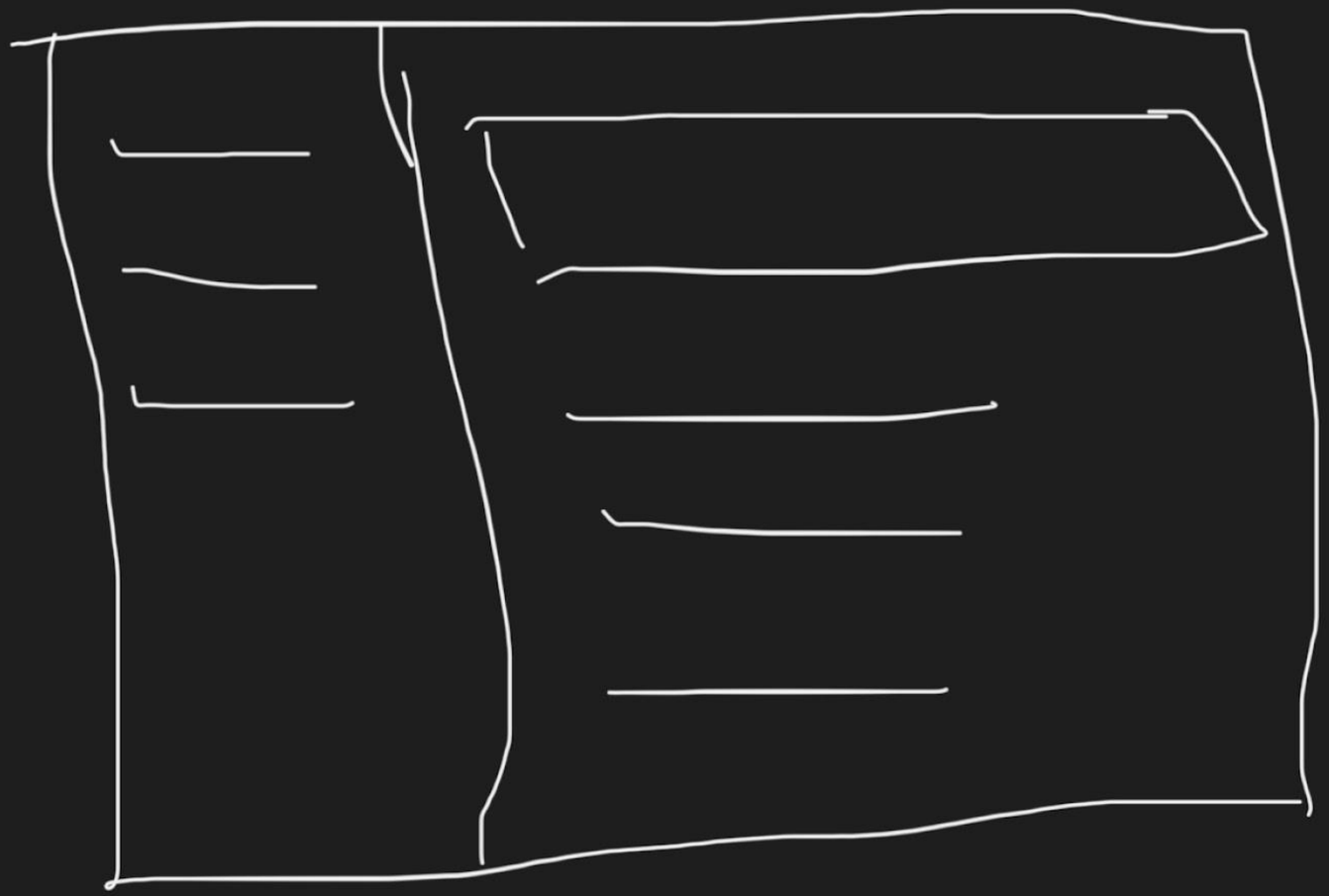

Google
2024

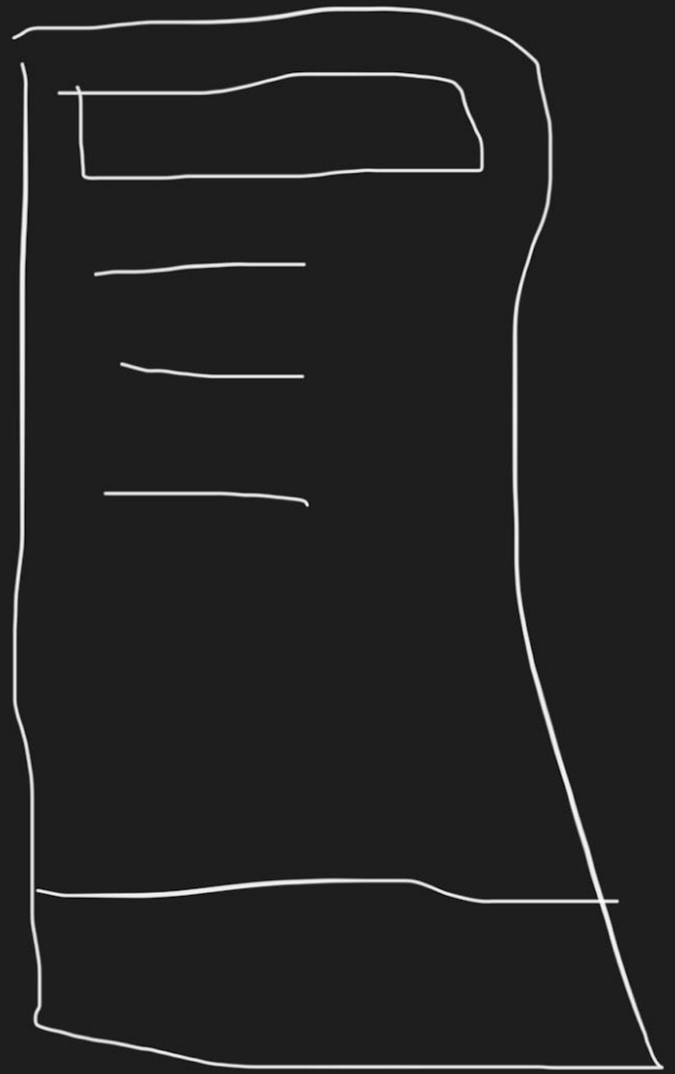

Kotlin
Multiplatform

ıg physical, hardware values for making layout decisions. It might be tempting to make decis
tangible value (Is the device a tablet? Does the physical screen have a certain aspect ratio?), but
› these questions may not be useful for determining the space your UI can work with.

Figure 1. *Phone, foldable, tablet, and laptop form factors*

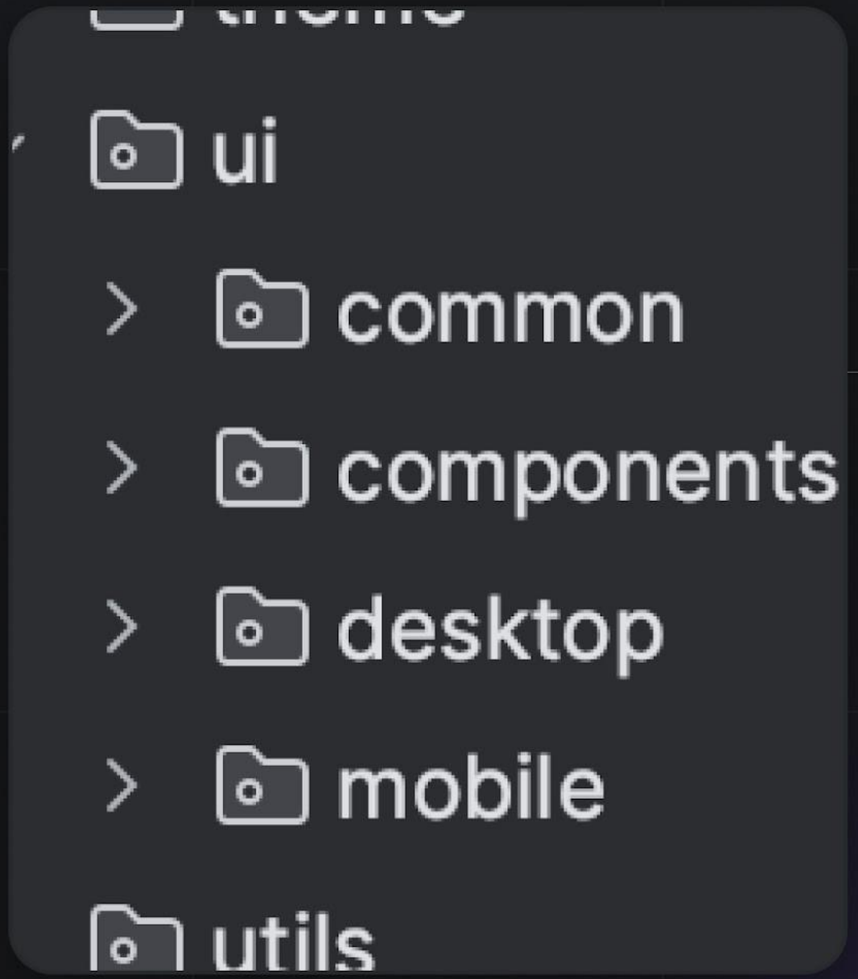
АРХИТЕКТУРА



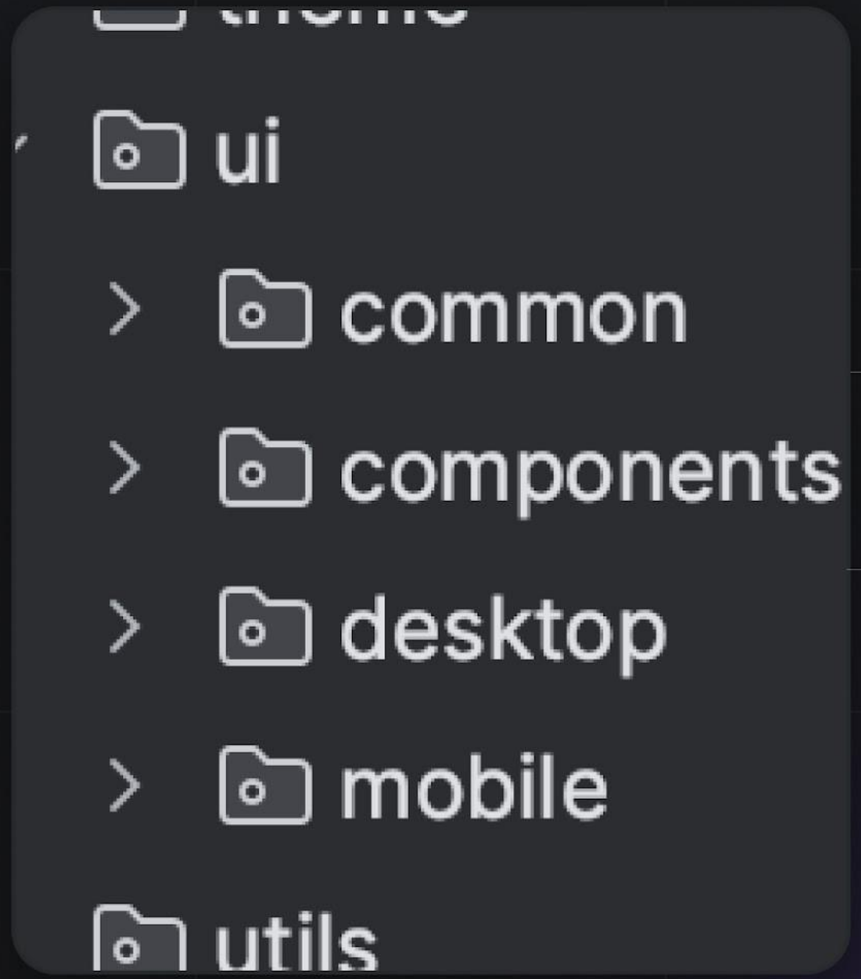


ОБЩИЕ ЭЛЕМЕНТЫ

- [x] Поисковая строка
- [x] Кнопки “Создать проект” / “Открыть проект”
- [x] Список проектов. Общая ячейка

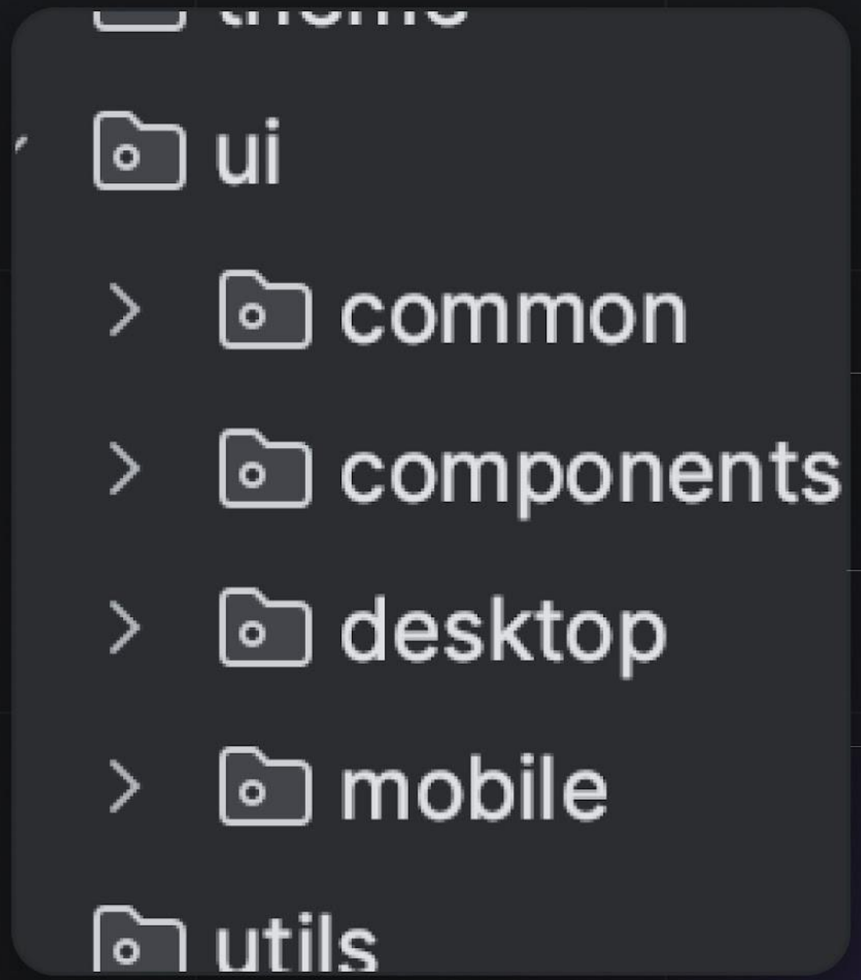


Screens + Views



Screens + Views

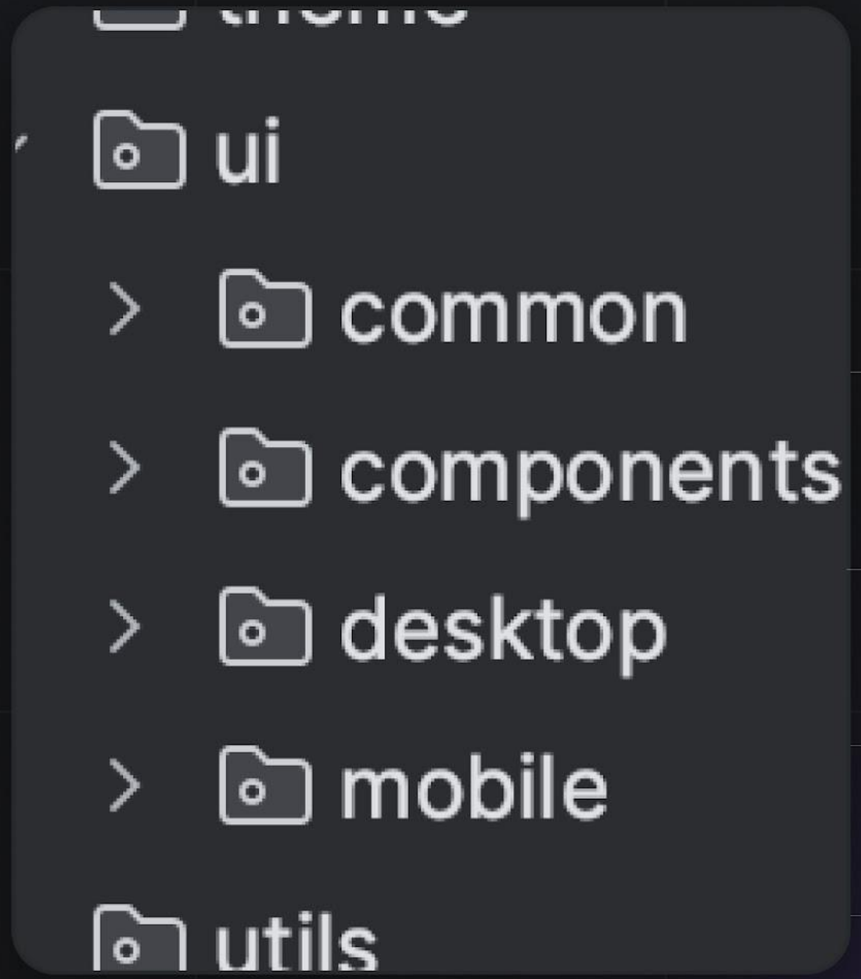
View Components



Screens + Views

View Components

Desktop Views



Screens + Views

View Components

Desktop Views

Mobile Views

COMPONENTS

Search Projects

New

Open

Component Level


```
@Composable
fun CommonButton(
    text: String,
    modifier: Modifier = Modifier,
    isEnabled: Boolean = true,
    isAction: Boolean = false,
    onClick: () -> Unit,
) {
    Button(
        modifier = modifier,
        onClick = onClick,
        colors = ButtonDefaults.buttonColors(
            backgroundColor = if (isAction) NarrowTheme.colors.primaryTintColor else NarrowTheme.colors.backgroundSecond

            disabledBackgroundColor = if (isAction) NarrowTheme.colors.primaryTintColor.copy(alpha = 0.3f) else NarrowTh

        ),
        enabled = isEnabled
    ) {
        Text(text, color = if (isEnabled) NarrowTheme.colors.textPrimary else NarrowTheme.colors.textMinor)
    }
```

Component Level


```
@Composable
fun CommonButton(
    text: String,
    modifier: Modifier = Modifier,
    isEnabled: Boolean = true,
    isAction: Boolean = false,
    onClick: () -> Unit,
) {
    Button(
        modifier = modifier,
        onClick = onClick,
        colors = ButtonDefaults.buttonColors(
            backgroundColor = if (isAction) NarrowTheme.colors.primaryTintColor
          else NarrowTheme.colors.backgroundSecondary,
            disabledBackgroundColor = if (isAction) NarrowTheme.colors.primaryTintColor.copy(alpha = 0.3f) else NarrowTh

        ),
        enabled = isEnabled
    ) {
        Text(text, color = if (isEnabled) NarrowTheme.colors.textPrimary else NarrowTheme.colors.textMinor)
    }
```

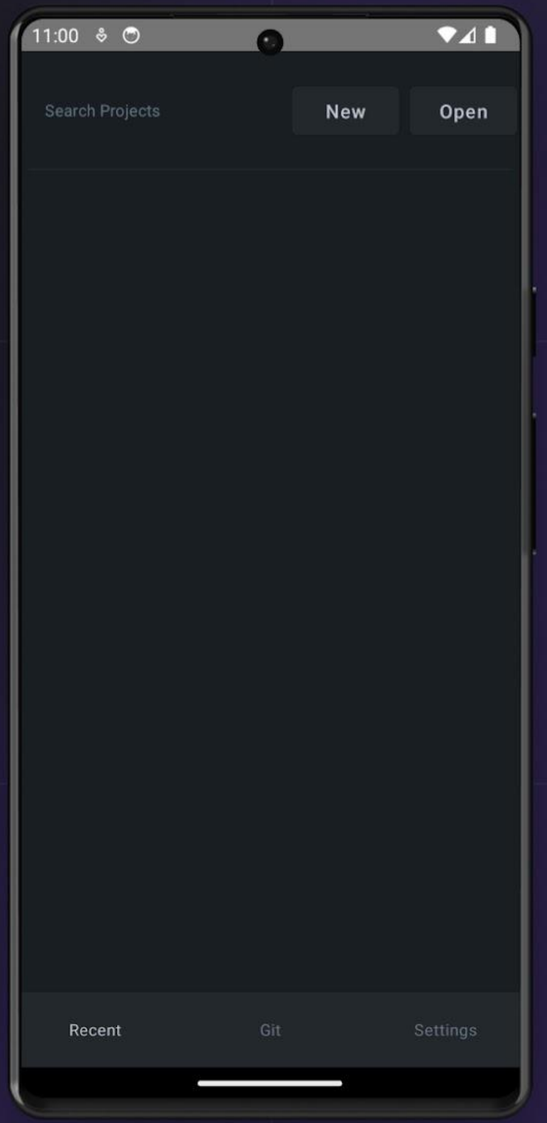
```
@Composable
fun NewProjectScreen() {
    ViewModel(factory = { NewProjectViewModel() }) { viewModel ->
        val platformConfiguration = LocalPlatformConfiguration.current

        if (platformConfiguration.appPlatform.isMobile()) {
            MobileNewProjectView(viewState) {
                viewModel.obtainEvent(it)
            }
        } else {
            DesktopNewProjectView(viewState) {
                viewModel.obtainEvent(it)
            }
        }
    }
}
```

```
@Composable
fun NewProjectScreen() {
    ViewModel(factory = { NewProjectViewModel() }) { viewModel ->
        val platformConfiguration = LocalPlatformConfiguration.current

        if (platformConfiguration.appPlatform.isMobile()) {
            MobileNewProjectView(viewState) {
                viewModel.obtainEvent(it)
            }
        } else {
            DesktopNewProjectView(viewState) {
                viewModel.obtainEvent(it)
            }
        }
    }
}
```

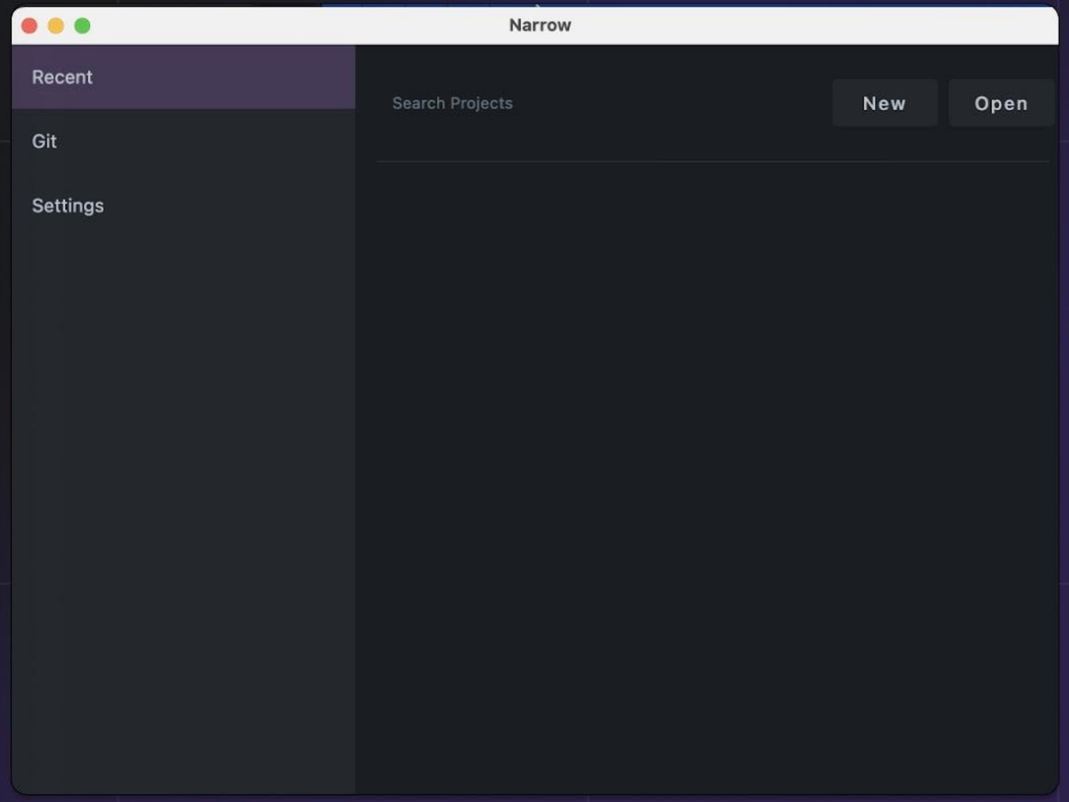
```
@Composable
fun NewProjectScreen() {
    ViewModel(factory = { NewProjectViewModel() }) { viewModel ->
        val platformConfiguration = LocalPlatformConfiguration.current

        if (platformConfiguration.appPlatform.isMobile()) {
            MobileNewProjectView(viewState) {
                viewModel.obtainEvent(it)
            }
        } else {
            DesktopNewProjectView(viewState) {
                viewModel.obtainEvent(it)
            }
        }
    }
}
```

```
@Composable
fun NewProjectScreen() {
    ViewModel(factory = { NewProjectViewModel() }) { viewModel ->
        val platformConfiguration = LocalPlatformConfiguration.current

        if (platformConfiguration.appPlatform.isMobile()) {
            MobileNewProjectView(viewState) {
                viewModel.obtainEvent(it)
            }
        } else {
            DesktopNewProjectView(viewState) {
                viewModel.obtainEvent(it)
            }
        }
    }
}
```

```
@Composable
fun RecentProjectView(...) {
    Row(...) {
        Column(...) {
            LogoAndTitleBlock()
            menuItems.forEach {
                MenuSelectableItem(...) { ... }
            }
        }

        Column(...) {
            when (selectedMenuItem.title) {
                title_1 -> RecentScreen()
                title_2 -> SettingsScreen()
            }
        }
    }
}
```



```
@Composable
fun DesktopMainScreen() {
    Row(...) {
        DesktopSideDrawerMenu(...) { ... }

        Box(...) {
            when (selectedTab) {
                title_1 -> RecentScreen()
                title_2 -> GetFromVCSScreen()
                title_3 -> SettingsScreen()
            }
        }
    }
}
```



HARDWARE + PLATFORM

HARDWARE

- [x] Keyboard Shortcuts
- [ ] Window controls
- [ ] Window State
- [ ] Platform tools (джойстики, камеры и так далее)
- [ ] Engine specifics (Browser, Desktop, etc)

HARDWARE

- [x] Keyboard Shortcuts
- [x] Window controls
- [ ] Window State
- [ ] Platform tools (джойстики, камеры и так далее)
- [ ] Engine specifics (Browser, Desktop, etc)

HARDWARE

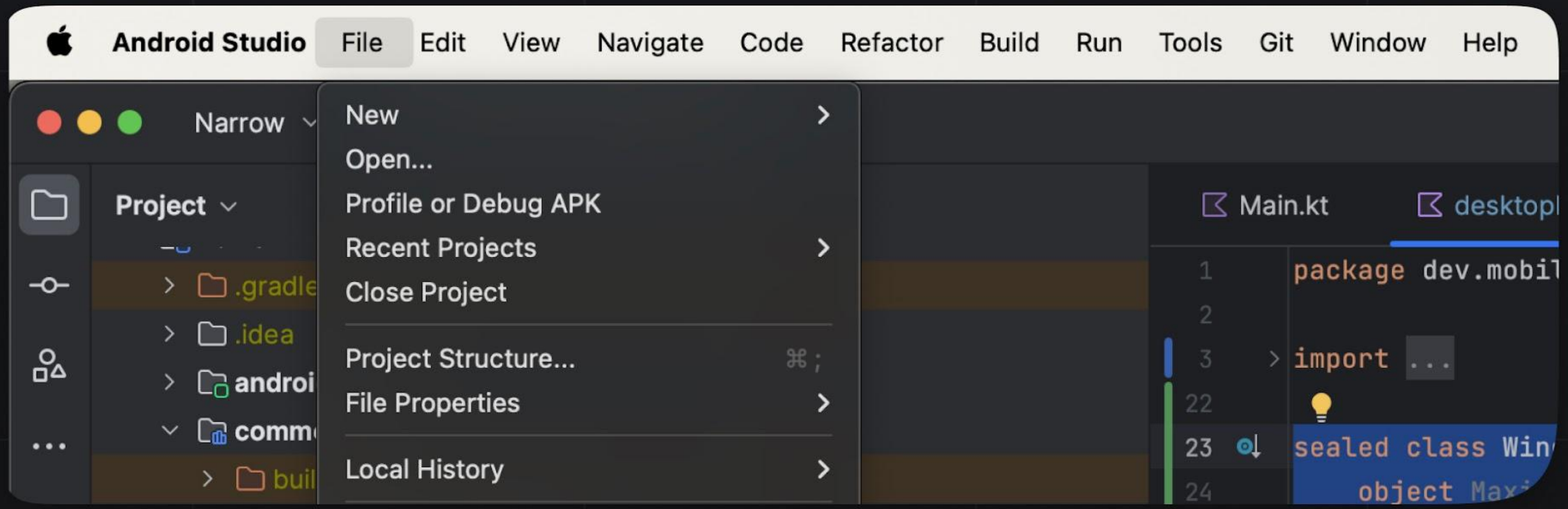
- [x] Keyboard Shortcuts
- [x] Window controls
- [x] Window State
- [ ] Platform tools (джойстики, камеры и так далее)
- [ ] Engine specifics (Browser, Desktop, etc)

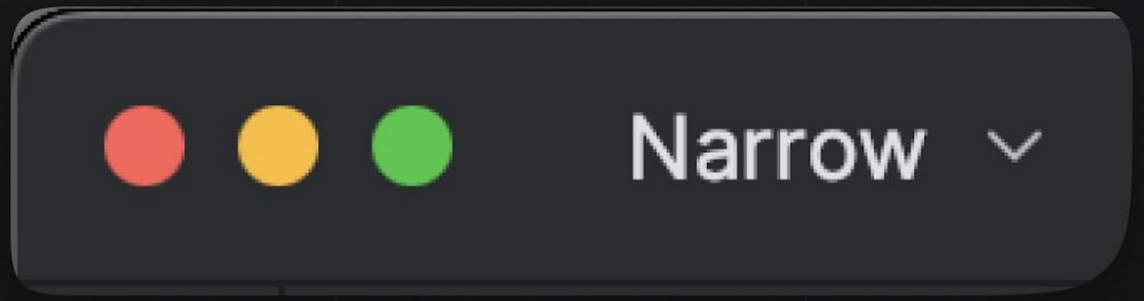
HARDWARE

- [x] Keyboard Shortcuts
- [x] Window controls
- [x] Window State
- [x] Platform tools (джойстики, камеры и так далее)
- [ ] Engine specifics (Browser, Desktop, etc)

HARDWARE

- [x] Keyboard Shortcuts
- [x] Window controls
- [x] Window State
- [x] Platform tools (джойстики, камеры и так далее)
- [x] Engine specifics (Browser, Desktop, etc)


Android Studio
File
Edit
View
Navigate
Code
Refactor
Build
Run
Tools
Git
Window
Help
Narrow
New
Open...
Profile or Debug APK
Recent Projects
Close Project
Project Structure... ⌘;
File Properties
Local History
Project
.gradle
.idea
Main.kt
1 package dev.mobil
2
3 import ...
22
23 sealed class Win
24 object Max


Narrow

Custom Size -> Maximized

```
sealed class WindowState {
    object Maximized : WindowState()
    object Minimized : WindowState()
    data class Custom(
        val width: Dp,
        val height: Dp
    ) : WindowState()
}
```

```
class AppSettings() {

    private val _windowState = MutableStateFlow<WindowState>(WindowState.Maximized)
    val windowState: StateFlow<WindowState?> = _windowState
}
```

```
fun updateWindowState(windowState: WindowState) {
    _windowState.value = windowState
}
```

```
val LocalAppSettings = staticCompositionLocalOf { AppSettings() }
```

```
fun main() = application {
    val appState = remember { AppSettings() }

    Window(...) {
        CompositionLocalProvider(
            LocalAppState provides appState
        ) {
            MainScreen { ... }
        }
    }
```

```
@Composable
fun NewProjectScreen() {
    ...
    val appSettings = LocalAppSettings.current

    when (viewAction) {
        is NewProjectAction.OpenCreatedProject -> appSettings.updateWindowState(WindowState.Maximized)
    }
}
```

```
class MainViewModel() {
    private val settingsRepository = Inject.instance<SettingsRepository>()

    init {
        fetchCurrentAppConfiguration()
    }

    private fun fetchCurrentAppConfiguration() {
        viewModelScope.launch {
            settingsRepository.appSettingsFlow.collectLatest {
                val settings = it ?: return@collectLatest
                viewState = viewState.copy(
                    appColor = settings.color,
                    isDarkTheme = settings.isDarkTheme
                )
            }
        }
    }
}
```

```
class MainViewModel() {
    private val settingsRepository = Inject.instance<SettingsRepository>()

    init {
        fetchCurrentAppConfiguration()
    }

    private fun fetchCurrentAppConfiguration() {
        viewModelScope.launch {
            settingsRepository.appSettingsFlow.collectLatest {
                val settings = it ?: return@collectLatest
                viewState = viewState.copy(
                    appColor = settings.color,
                    isDarkTheme = settings.isDarkTheme
                )
            }
        }
    }
}
```

NAVIGATION

NAVIGATION

- [x] Поддержка раздельных экранов (desktop vs mobile vs foldable)
- [ ] Поддержка навигации через окно
- [ ] Поддержка muttiplatform deeplinks
- [ ] Поддержка resize

NAVIGATION

- [x] Поддержка раздельных экранов (desktop vs mobile vs foldable)
- [x] Поддержка навигации через окно
- [ ] Поддержка muttiplatform deeplinks
- [ ] Поддержка resize

NAVIGATION

- [x] Поддержка раздельных экранов (desktop vs mobile vs foldable)
- [x] Поддержка навигации через окно
- [x] Поддержка muttiplatform deeplinks
- [ ] Поддержка resize

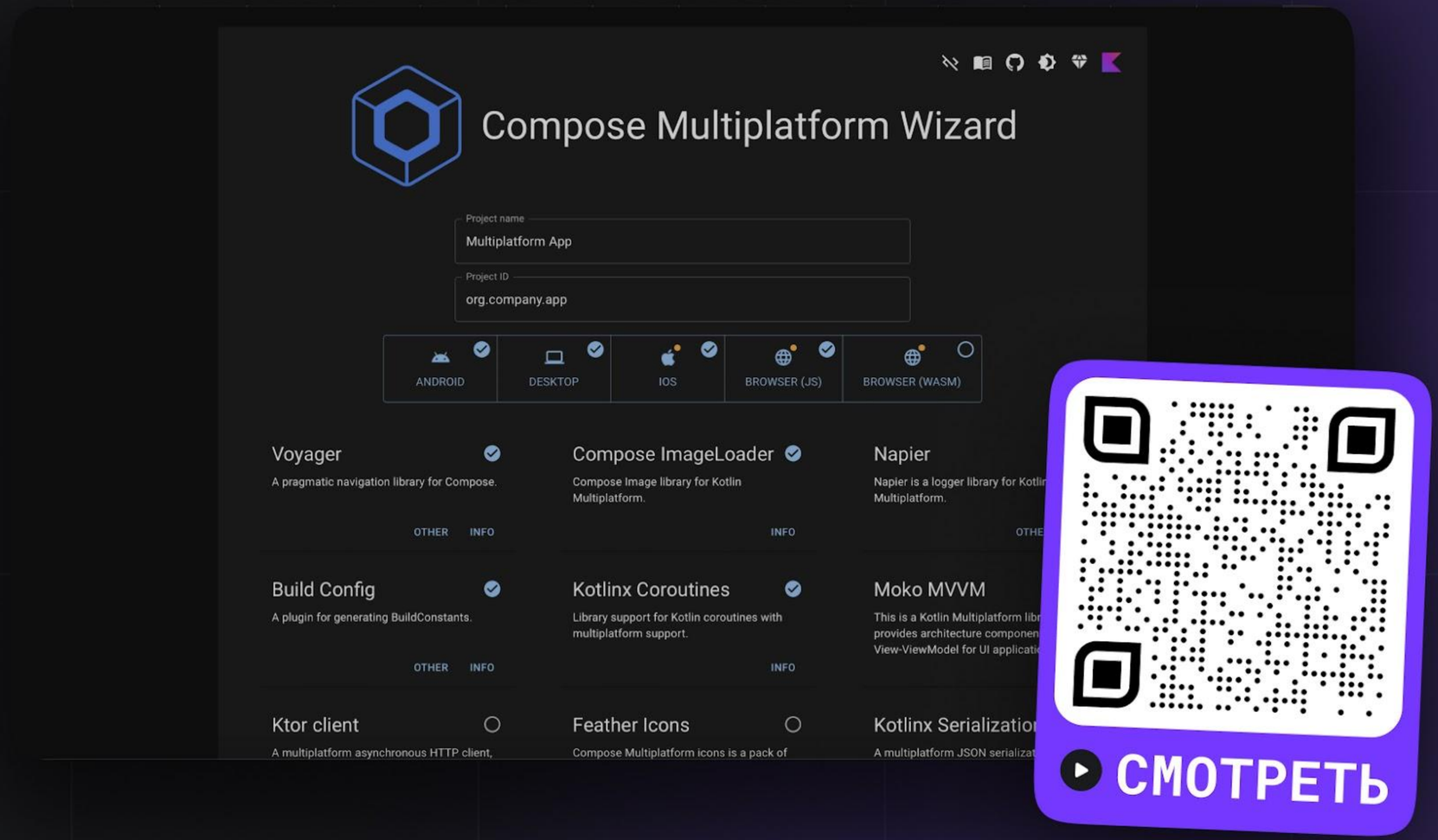
NAVIGATION

- [x] Поддержка раздельных экранов (desktop vs mobile vs foldable)
- [x] Поддержка навигации через окно
- [x] Поддержка muttiplatform deeplinks
- [x] Поддержка resize

```
fun RootComposeBuilder.navigationGraph(appPlatform: AppPlatform) {
    // Screens placed here

    if (appPlatform.isMobile()) {
        bottomNavigationGraph()
    } else {
        desktopNavigationGraph()
    }
}
```

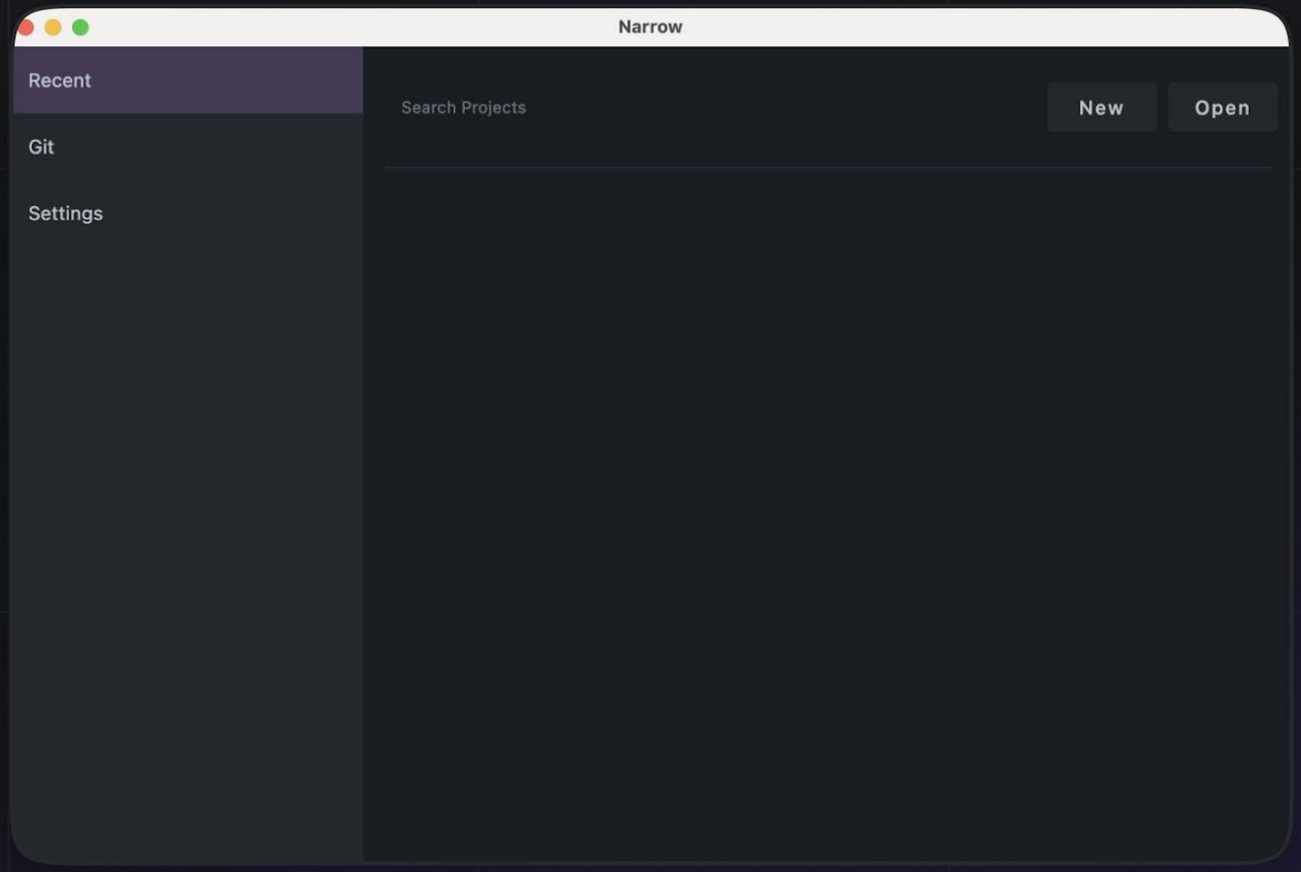
```
private fun RootComposeBuilder.bottomNavigationGraph() {
    bottomNavigation(...) {
        tab(...) {
            screen(NavigationTree.START) {
                RecentScreen()
            }
        }

        tab(...) {
            screen(NavigationTree.VCS) {
                GetFromVCSScreen()
            }
        }

        tab(...) {
            screen(NavigationTree.SETTINGS) {
                SettingsScreen()
            }
        }
    }
}
```

```
fun RootComposeBuilder.desktopNavigationGraph() {
    screen(NavigationTree.MAIN_SCREEN) {
        DesktopMainScreen()
    }
}
```

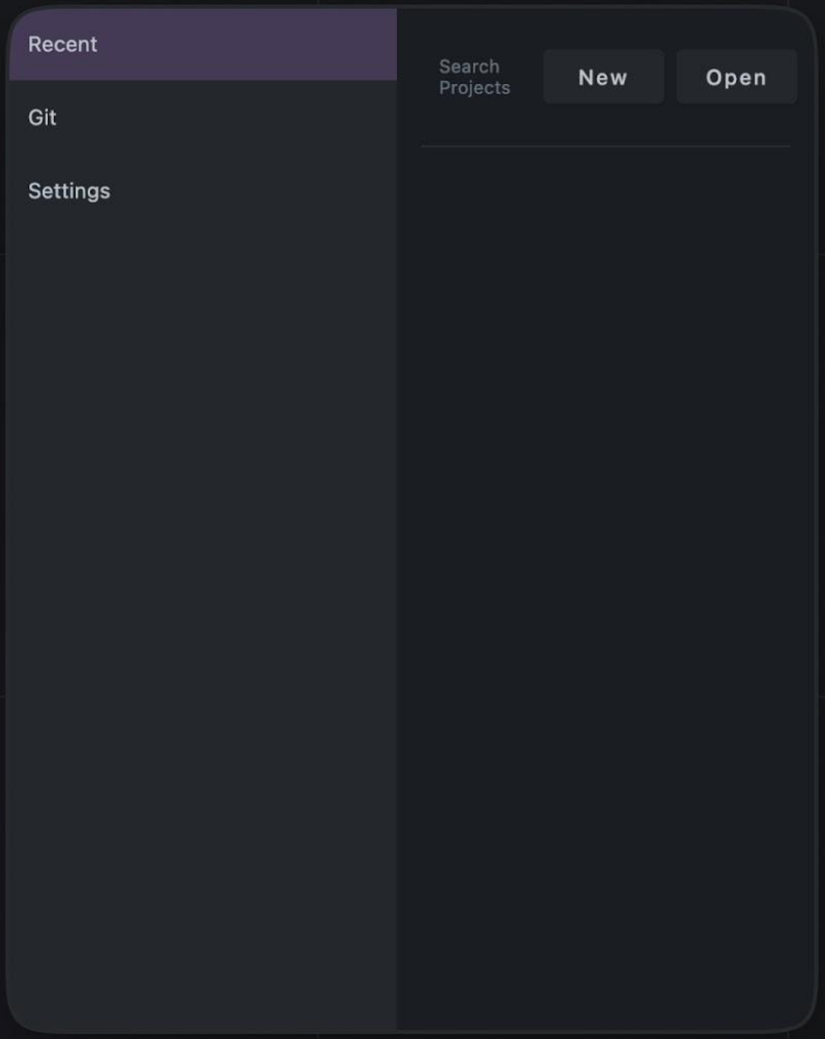

Compose Multiplatform Wizard
Project name
Multiplatform App
Project ID
org.company.app
ANDROID
DESKTOP
IOS
BROWSER (JS)
BROWSER (WASM)
Voyager
A pragmatic navigation library for Compose.
OTHER
INFO
Compose ImageLoader
Compose Image library for Kotlin Multiplatform.
INFO
Napier
Napier is a logger library for Kotlin Multiplatform.
Build Config
A plugin for generating BuildConstants.
OTHER
INFO
Kotlinx Coroutines
Library support for Kotlin coroutines with multiplatform support.
INFO
Moko MVVM
Ktor client
A multiplatform asynchronous HTTP client,
Feather Icons
Compose Multiplatform icons is a pack of
СМОТРЕТЬ

```
internal actual fun openUrl(url: String?) {
    val uri = url?.let { URI.create(it) } ?:
return
    Desktop.getDesktop().browse(uri)
}
```

WHAT ELSE?


Narrow
Recent
Git
Settings
Search Projects
New
Open

->


Recent
Git
Settings
Search Projects
New
Open

```
implementation(platform("androidx.compose:compose-bom:2023.08.00"))
implementation("androidx.compose.material3:material3-window-size-class")
```

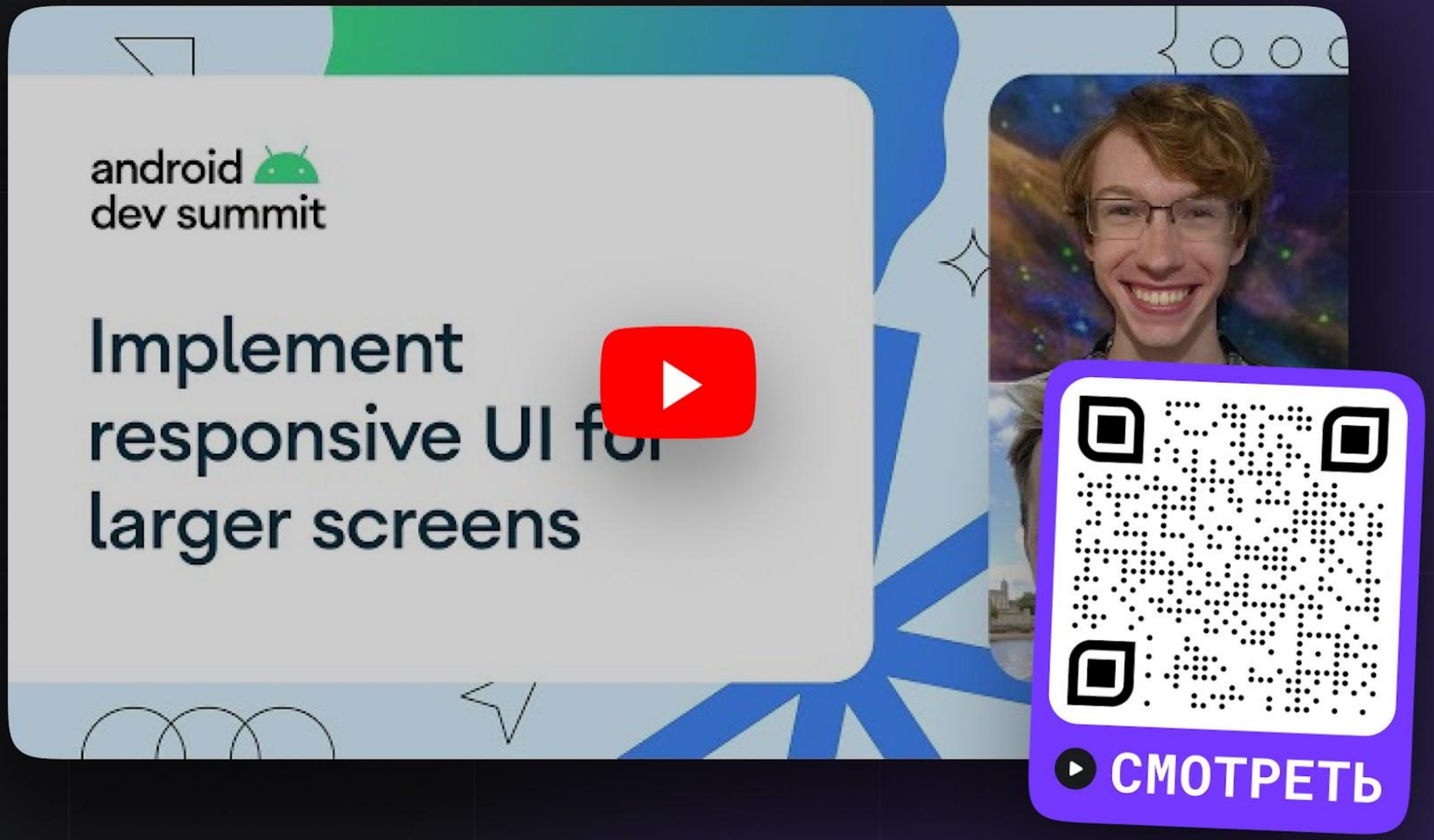
```
if (windowSize.widthSizeClass > WindowWidthSizeClass.Compact) {
            Row(
                Modifier
                    .fillMaxSize()
                    .padding(innerPadding)
            ) {
                MRailBar()

                MainNavHost(
                    modifier = Modifier,
                    windowSize = windowSize
                )
            }
        } else {
            MainNavHost(
                modifier = Modifier.padding(innerPadding),
                windowSize = windowSize
            )
        }
}
```

```
implementation "com.google.accompanist:accompanist-adaptive:<version>"
```

```
@Composable
fun TwoPane(first: @Composable () -> Unit, second: @Composable () -> Unit,
            strategy: TwoPaneStrategy, displayFeatures: List<DisplayFeature>,
            modifier: Modifier = Modifier,
            foldAwareConfiguration: FoldAwareConfiguration = FoldAwareConfiguration.AllFolds)
```

```
@Composable
fun TwoPane(first: @Composable () -> Unit, second: @Composable () -> Unit,
            strategy: TwoPaneStrategy, displayFeatures: List<DisplayFeature>,
            modifier: Modifier = Modifier,
            foldAwareConfiguration: FoldAwareConfiguration = FoldAwareConfiguration.AllFolds)
```

```
abstract fun calculateSplitResult(
    density: Density,
    layoutDirection: LayoutDirection,
    layoutCoordinates: LayoutCoordinates
): SplitResult
```


android
dev summit
Implement
responsive UI for
larger screens
СМОТРЕТЬ


COMPOSE
MULTIPLATFORM
SUPPORT

```
enum class WindowType {
    Portrait, Landscape, Box
}

@Composable
expect fun calculateWindowType(): WindowType
```

ИТОГИ

ИТОГИ

- [x] Пинайте дизайнера ногами, чтобы он учитывал все форм-факторы
- [ ] Заложите в проект различные форм-факторы
- [ ] Делайте максимально абстрактные экраны
- [ ] Используйте дизайн-систему и компоненты
- [ ] Используйте концепцию контейнеров

ИТОГИ

- [x] Пинайте дизайнера ногами, чтобы он учитывал все форм-факторы
- [x] Заложите в проект различные форм-факторы
- [ ] Делайте максимально абстрактные экраны
- [ ] Используйте дизайн-систему и компоненты
- [ ] Используйте концепцию контейнеров

ИТОГИ

- [x] Пинайте дизайнера ногами, чтобы он учитывал все форм-факторы
- [x] Заложите в проект различные форм-факторы
- [x] Делайте максимально абстрактные экраны
- [ ] Используйте дизайн-систему и компоненты
- [ ] Используйте концепцию контейнеров

ИТОГИ

- [x] Пинайте дизайнера ногами, чтобы он учитывал все форм-факторы
- [x] Заложите в проект различные форм-факторы
- [x] Делайте максимально абстрактные экраны
- [x] Используйте дизайн-систему и компоненты
- [ ] Используйте концепцию контейнеров

ИТОГИ

- [x] Пинайте дизайнера ногами, чтобы он учитывал все форм-факторы
- [x] Заложите в проект различные форм-факторы
- [x] Делайте максимально абстрактные экраны
- [x] Используйте дизайн-систему и компоненты
- [x] Используйте концепцию контейнеров

ПОДПИСАТЬСЯ
НА YOUTUBE

СПАСИБО
ЗА ВНИМАНИЕ!

ОСТАВИТЬ
ОТЗЫВ

АЛЕКСЕЙ ГЛАДКОВ
Mobile Developer

Mobiledevelopernews
Mobiledeveloper
Mobiledevelopercourse@Gmail.Com