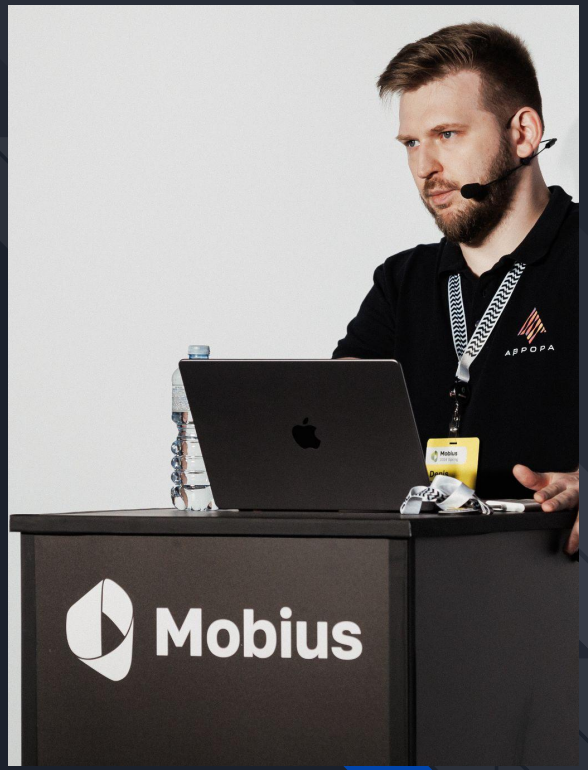
Пробуем Compose-alpha для ОС Аврора

- Супрун Денис Алексеевич
- стаж в мобильной разработке более 10 лет
- Участник программы бета-тестирования Аврора ОС
- Аврора-энтузиаст, спикер

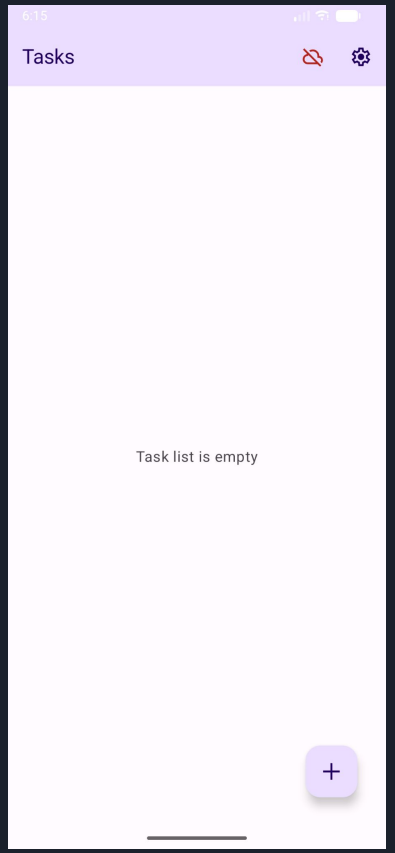


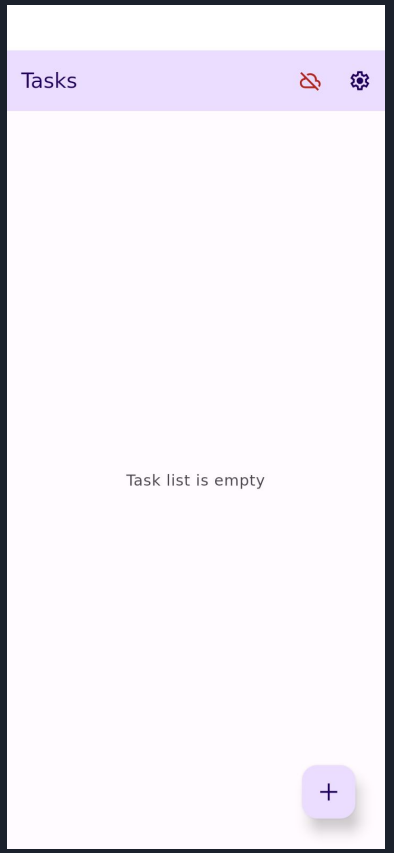
План

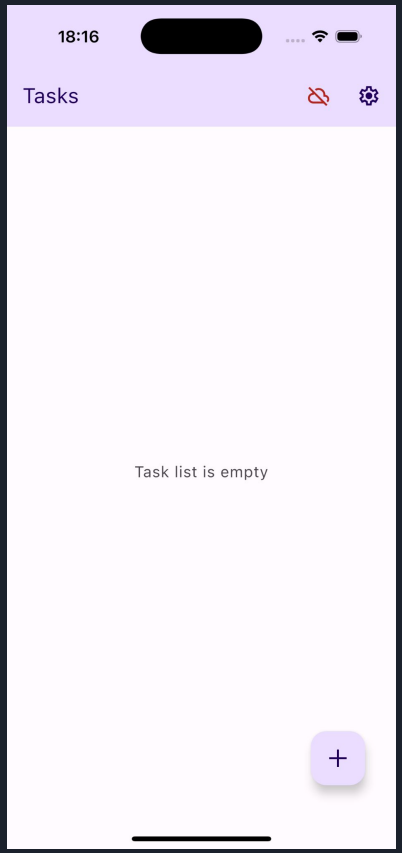
1. С чего начнём?
2. Как подключить Compose-alpha для Аврора ОС
3. Разбираемся с Compose-Preview
4. Дополняем Koin
5. Compose Resources
6. Отступы, клавиатура, тема
7. Modal Bottom Sheet
8. Итоги

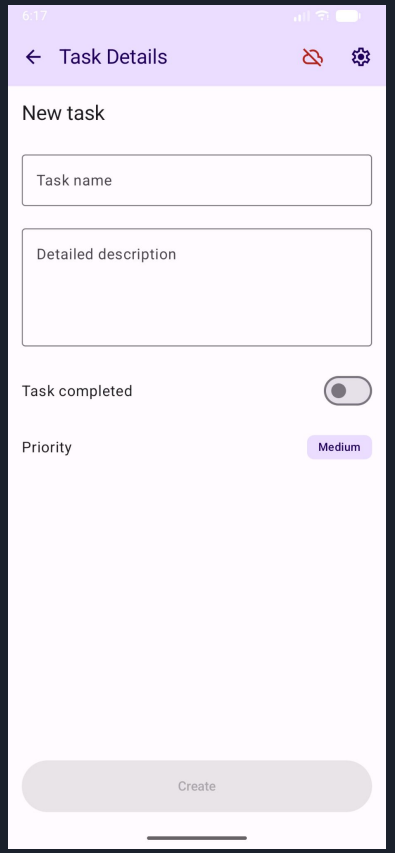
План

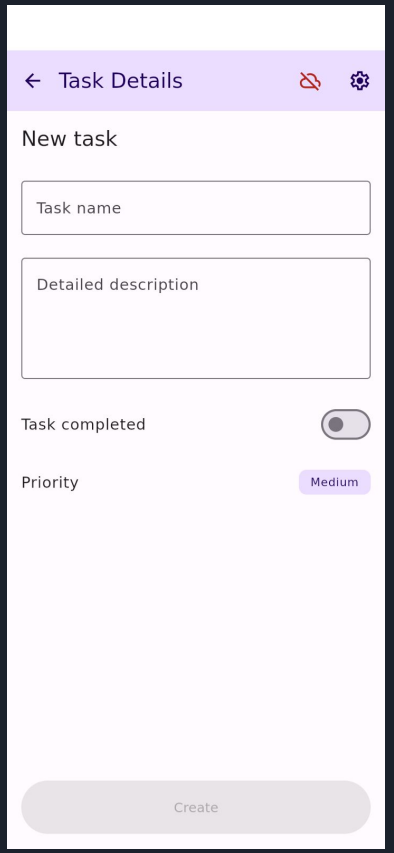
1. С чего начнём?
2. Как подключить Compose-alpha для Аврора ОС
3. Разбираемся с Compose-Preview
4. Дополняем Koin
5. Compose Resources
6. Отступы, клавиатура, тема
7. Modal Bottom Sheet
8. Итоги

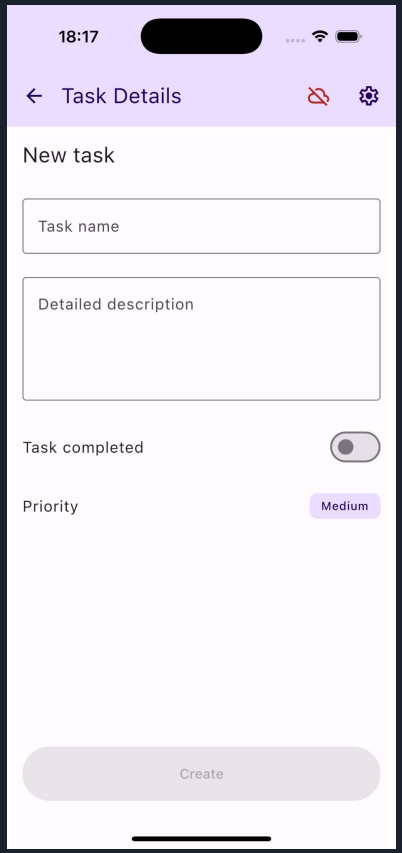

Tasks
Task list is empty

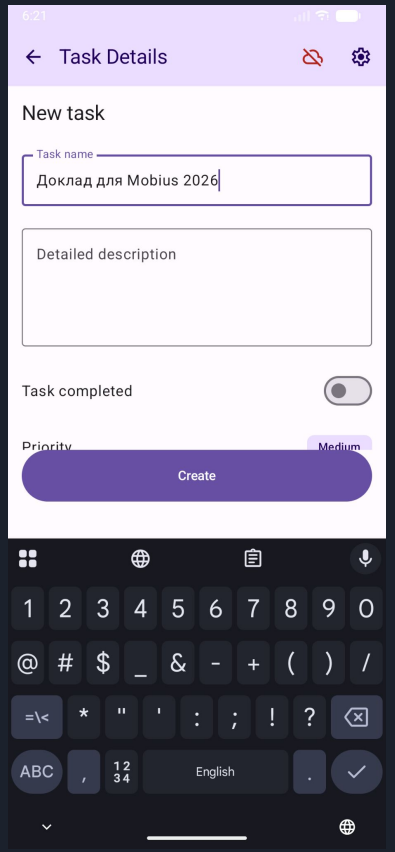

Tasks
Task list is empty

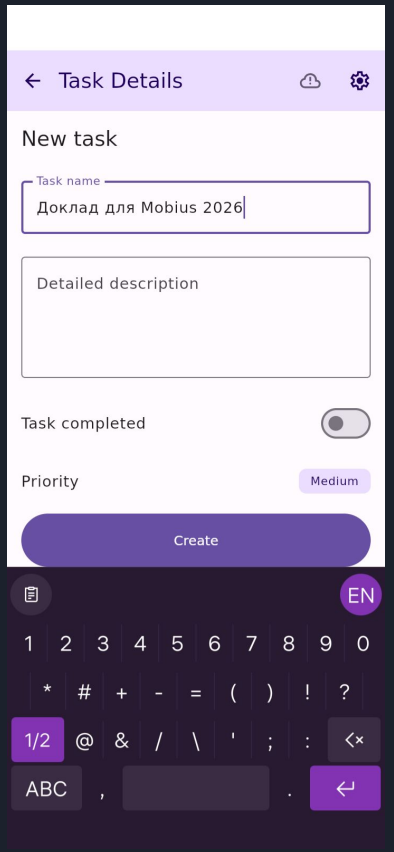

18:16
Tasks
Task list is empty

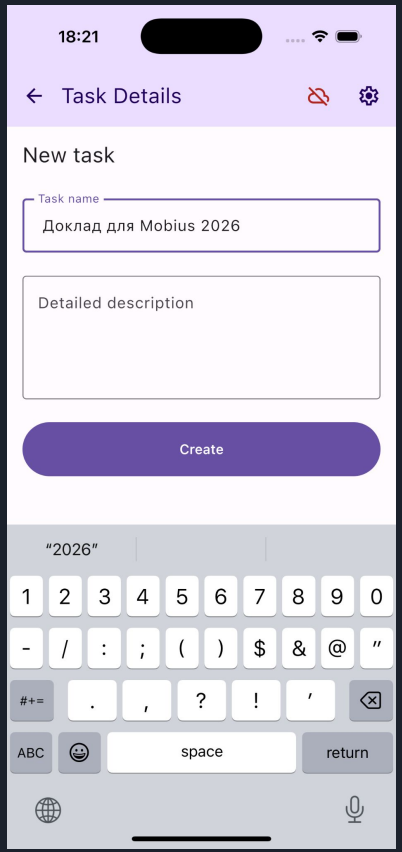

Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create

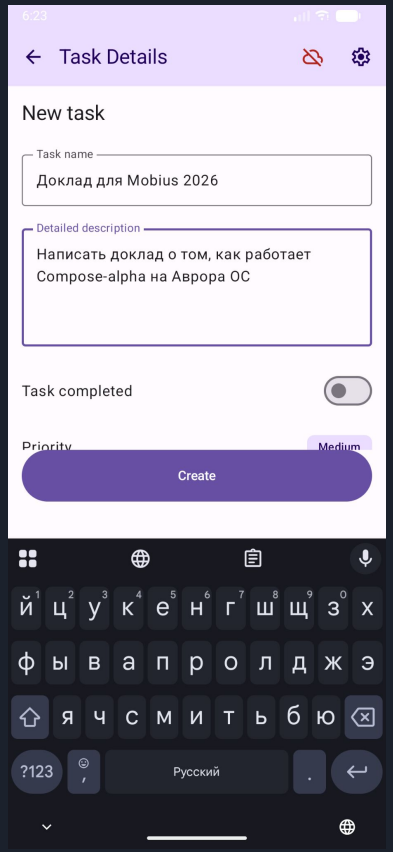

Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create

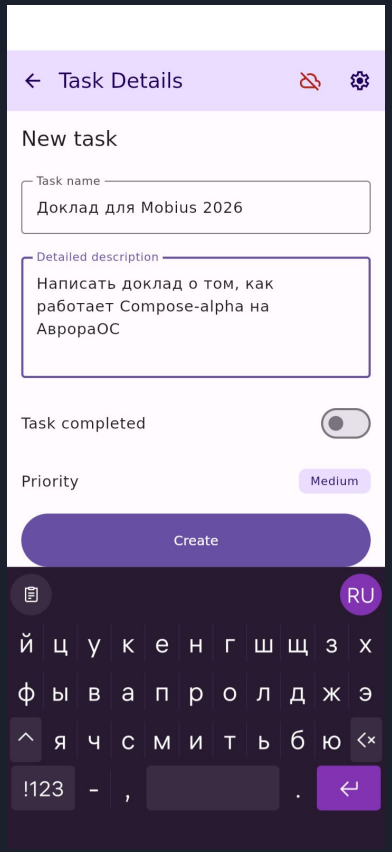

18:17
Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create

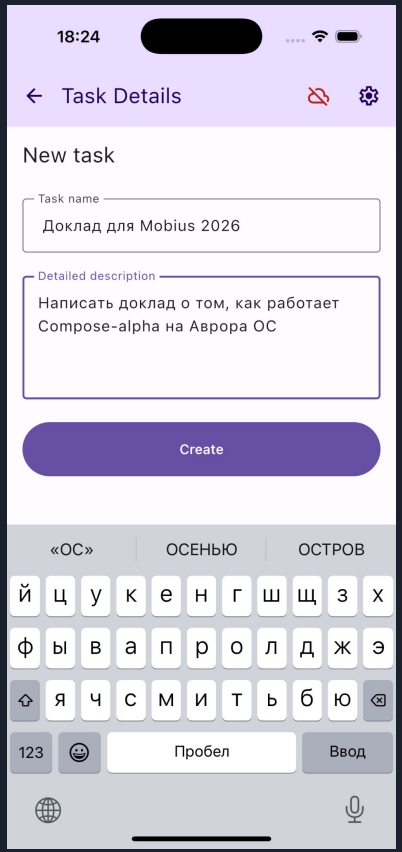

Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Task completed
Priority
Medium
Create
English

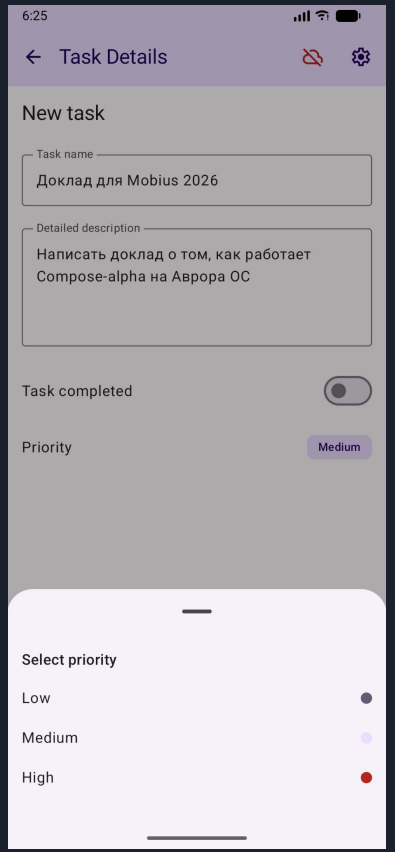

Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Task completed
Priority
Medium
Create
EN
ABC

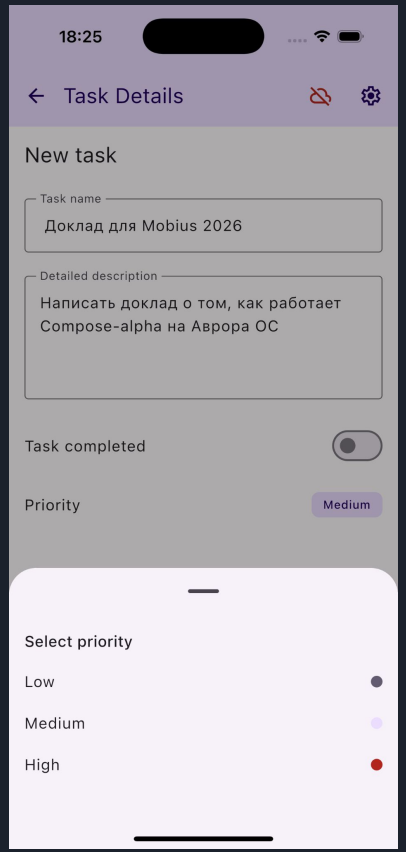

18:21
Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Create
“2026”
ABC
space
return

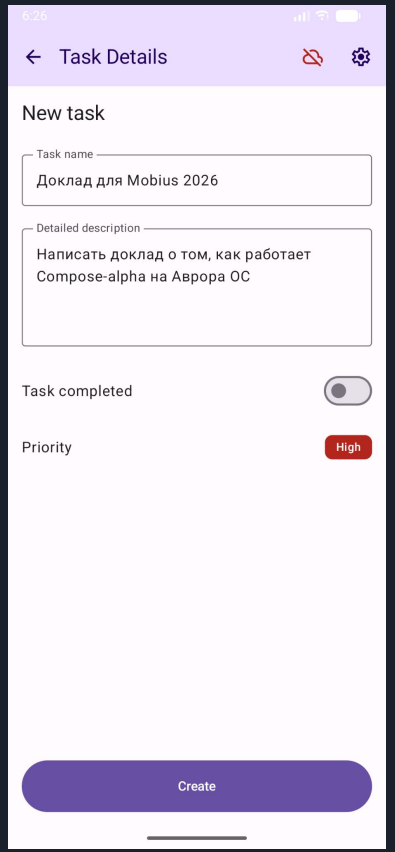

Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Написать доклад о том, как работает Compose-alpha на Аврора ОС
Task completed
Priority
Medium
Create
Русский

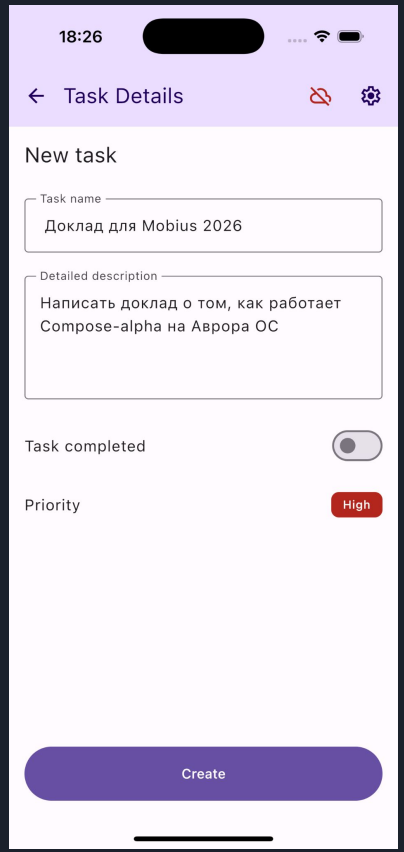

Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Написать доклад о том, как работает Compose-alpha на АвроraОС
Task completed
Priority
Medium
Create
RU

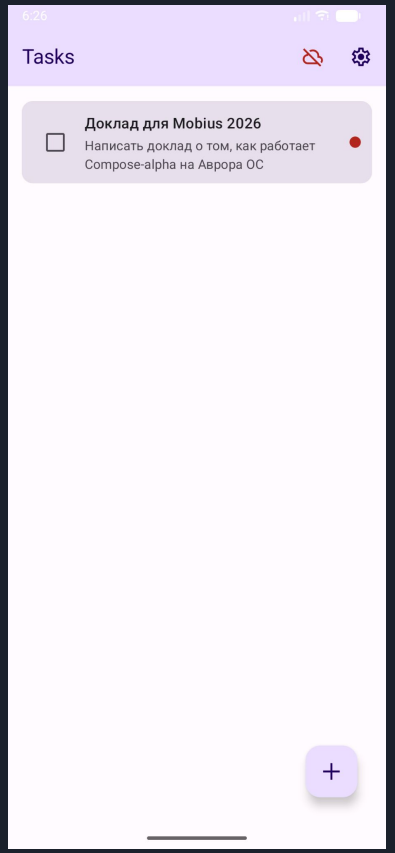

18:24
Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Написать доклад о том, как работает Compose-alpha на Аврора ОС
Create
«ОС»
ОСЕНЬЮ
ОСТРОВ
Пробел
Ввод

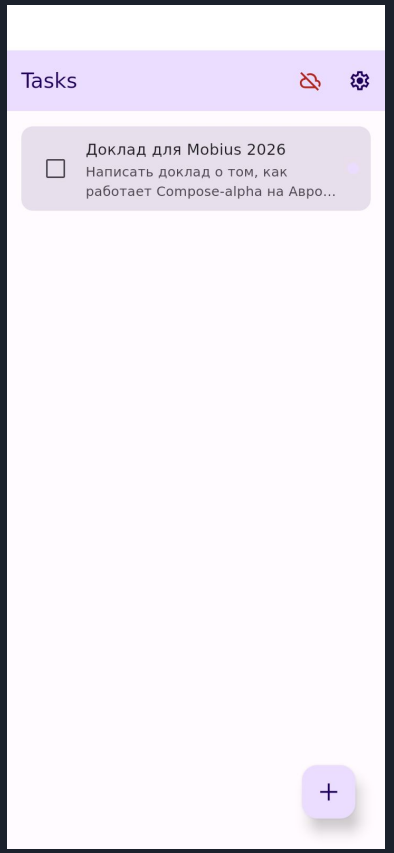

6:25
Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Написать доклад о том, как работает Compose-alpha на Аврора ОС
Task completed
Priority
Medium
Select priority
Low
Medium
High

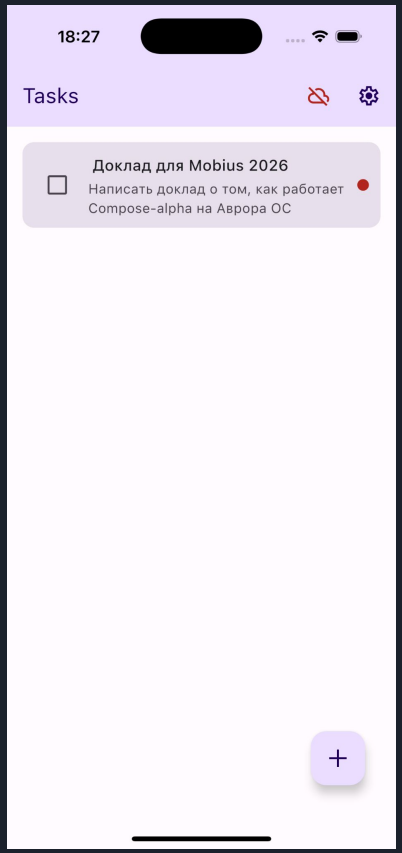

18:25
Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Написать доклад о том, как работает Compose-alpha на Аврора ОС
Task completed
Priority
Medium
Select priority
Low
Medium
High

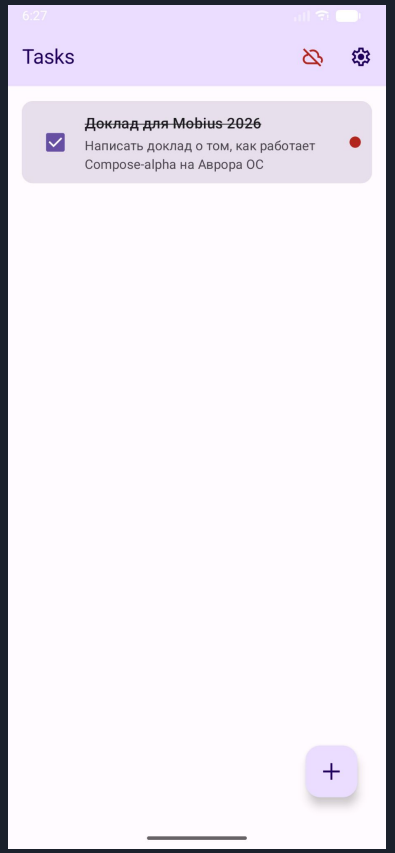

Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Написать доклад о том, как работает Compose-alpha на Аврора ОС
Task completed
Priority
High
Create

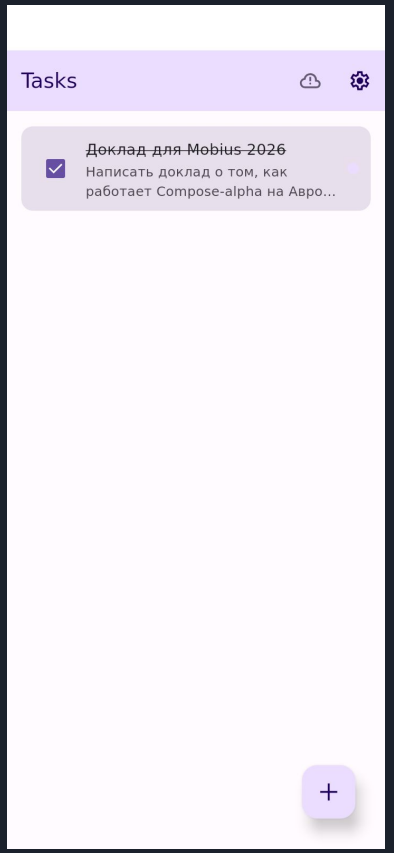

18:26
Task Details
New task
Task name
Доклад для Mobius 2026
Detailed description
Написать доклад о том, как работает Compose-alpha на Аврора ОС
Task completed
Priority
High
Create

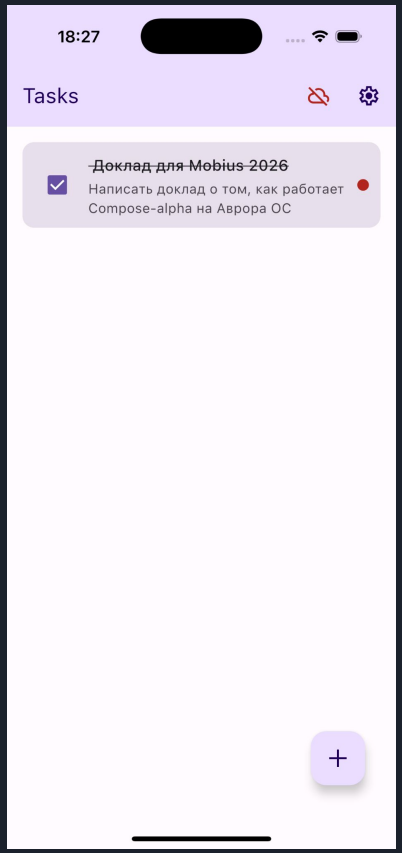

Tasks
Доклад для Mobius 2026
Написать доклад о том, как работает Compose-alpha на Аврора ОС

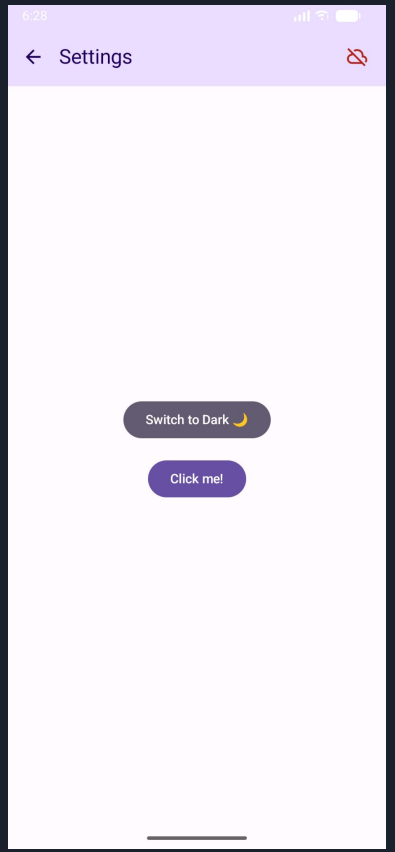

Tasks
Доклад для Mobius 2026
Написать доклад о том, как работает Compose-alpha на Авро...

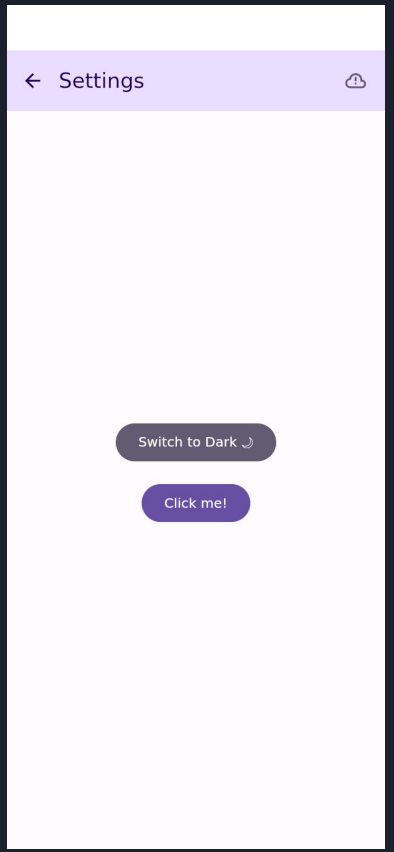

18:27
Tasks
Доклад для Mobius 2026
Написать доклад о том, как работает Compose-alpha на Аврора ОС

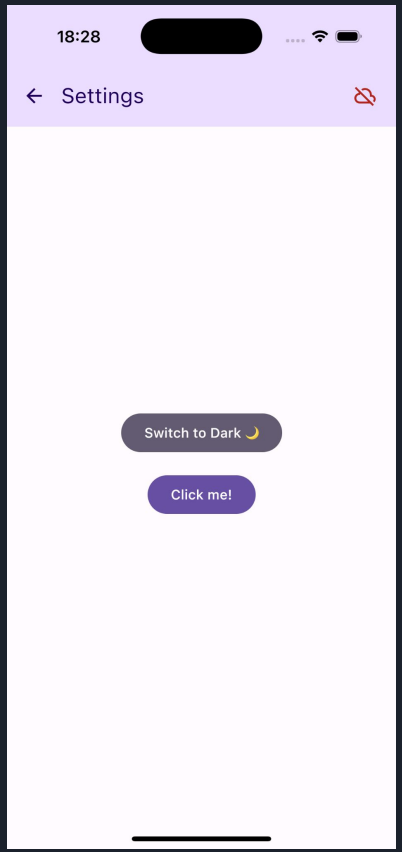

6:27
Tasks
~~Доклад для Mobius 2026~~
Написать доклад о том, как работает Compose-alpha на Аврора ОС

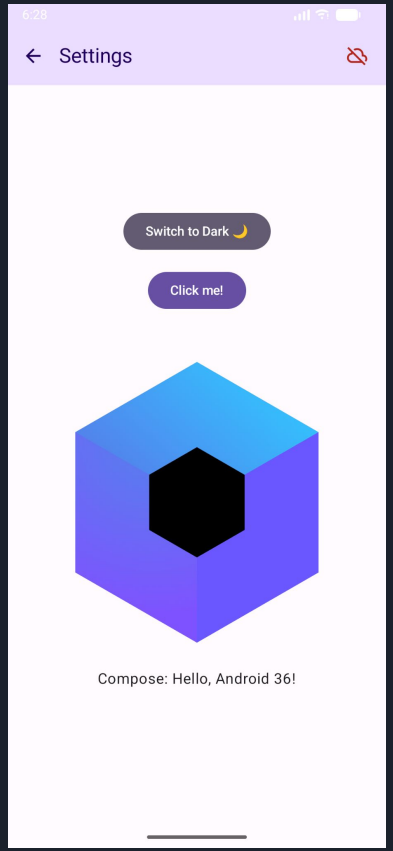

Tasks
~~Доклад для Mobius 2026~~
Написать доклад о том, как работает Compose-alpha на Авро...

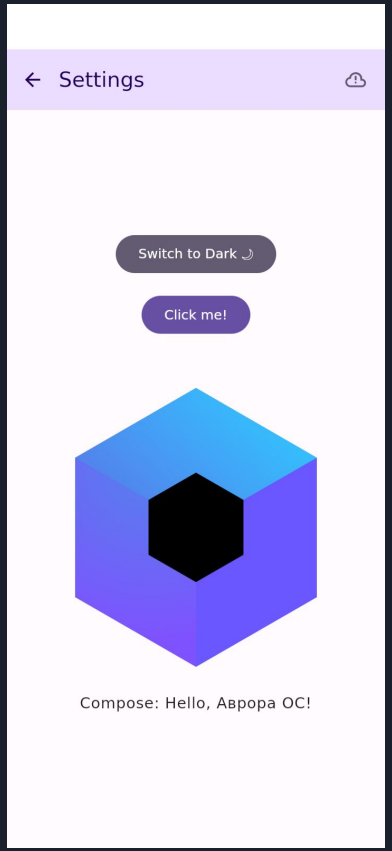

18:27
Tasks
~~Доклад для Mobius 2026~~
Написать доклад о том, как работает Compose-alpha на Аврора ОС

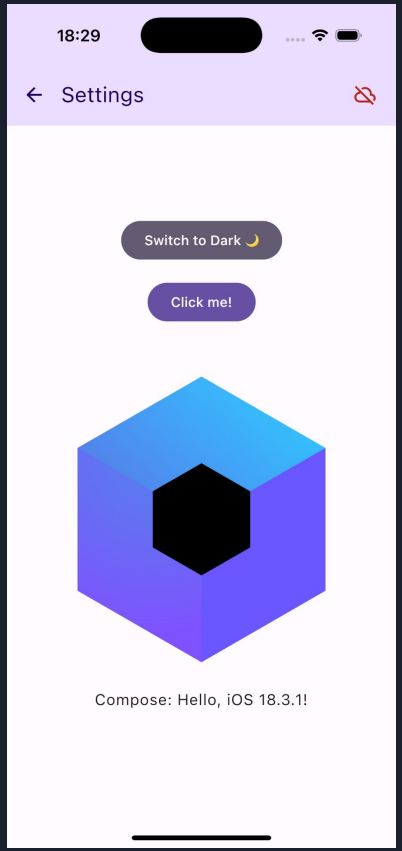

Settings
Switch to Dark
Click me!

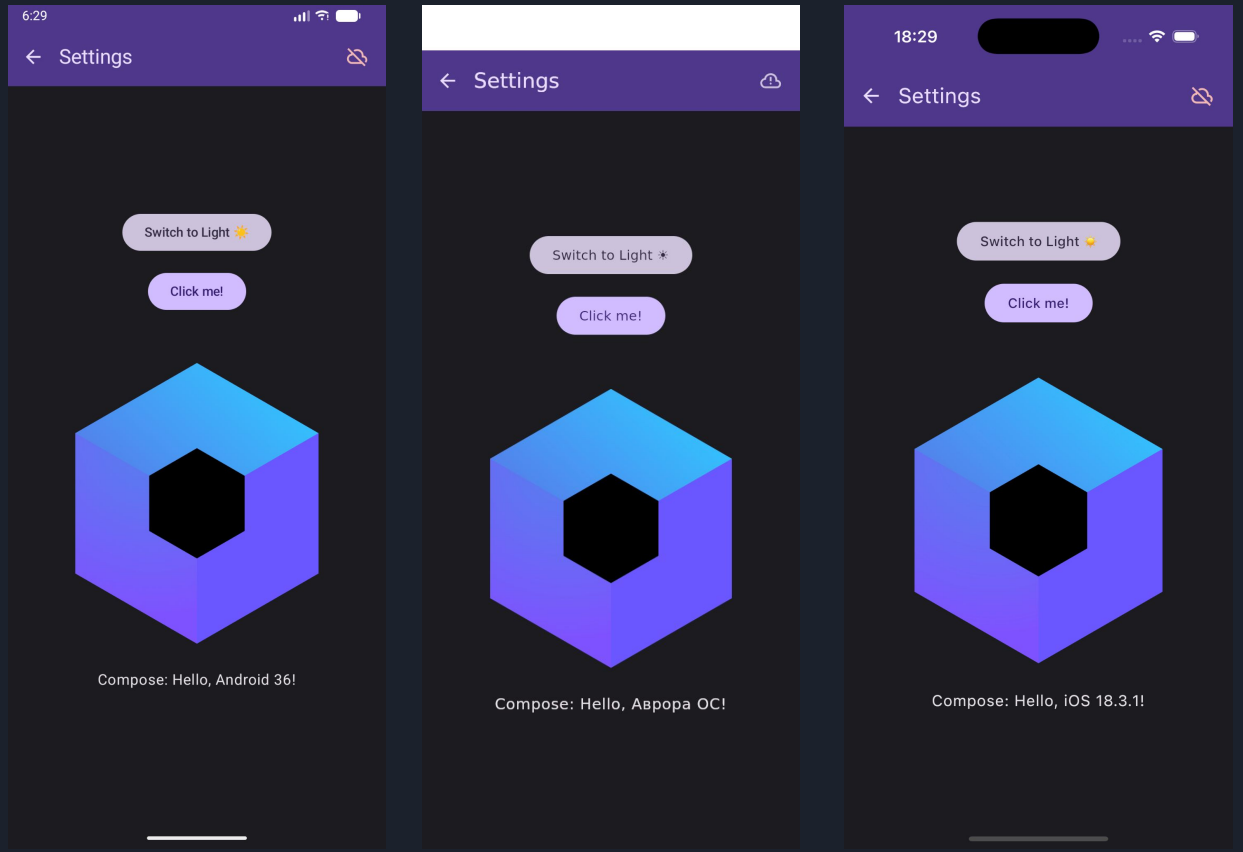

Settings
Switch to Dark
Click me!


18:28
Settings
Switch to Dark
Click me!


Settings
Switch to Dark
Click me!
Compose: Hello, Android 36!


Settings
Switch to Dark
Click me!
Compose: Hello, Аврора ОС!


18:29
Settings
Switch to Dark
Click me!
Compose: Hello, iOS 18.3.1!


6:29
Settings
Switch to Light ☀️
Click me!
Compose: Hello, Android 36!
Settings
Switch to Light ☀
Click me!
Compose: Hello, Аврора ОС!
18:29
Settings
Switch to Light ☀️
Click me!
Compose: Hello, iOS 18.3.1!

Список задач

```
dependencies {
    implementation(libs.compose.runtime)
    implementation(libs.compose.foundation)
    implementation(libs.compose.material3)
    implementation(libs.compose.ui)
    implementation(libs.compose.components.resources)
    implementation(libs.compose.uiToolingPreview)
    implementation(libs.androidx.lifecycle.viewmodelCompose)
    implementation(libs.androidx.lifecycle.runtimeCompose)
    implementation(libs.ktor.client.core)

    implementation(libs.koin.core)
    implementation(libs.koin.compose)
    implementation(libs.koin.compose.viewmodel)

    implementation(libs.kotlinx.serialization.json)
    implementation(libs.navigation.compose)
```

Список задач

```
    // Room
    implementation(libs.androidx.room.runtime)
    implementation(libs.androidx.sqlite.bundled)

    implementation(projects.shared)
}
androidMain.dependencies {
    implementation(libs.compose.uiToolingPreview)
    implementation(libs.androidx.activity.compose)
    implementation(libs.ktor.client.okhttp)
}
iosMain.dependencies {
    implementation(libs.ktor.client.darwin)
}
commonTest.dependencies {
    implementation(libs.kotlin.test)
}
```

Список задач

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        enableEdgeToEdge()
        super.onCreate(savedInstanceState)

        setContent {
            App(
                koinConfig = {
                    androidContext(applicationContext)
                }
            )
        }
    }
}
```

Список задач

```
@Composable
fun App(koinConfig: KoinAppDeclaration.() -> Unit = {}) {
    KoinApplication(application = {
        modules(platformModule, appModule)
        koinConfig()
    }) {
        val useDarkTheme = ...
        AppTheme(darkTheme = useDarkTheme) {
            val navController = rememberNavController()
            Scaffold(...) { paddingValues ->
                Box(modifier = Modifier...) {
                    AppNavigation(
                        navController = navController,
                        isDarkTheme = useDarkTheme
                    )
                }
            }
        }
    }
}
```

Список задач

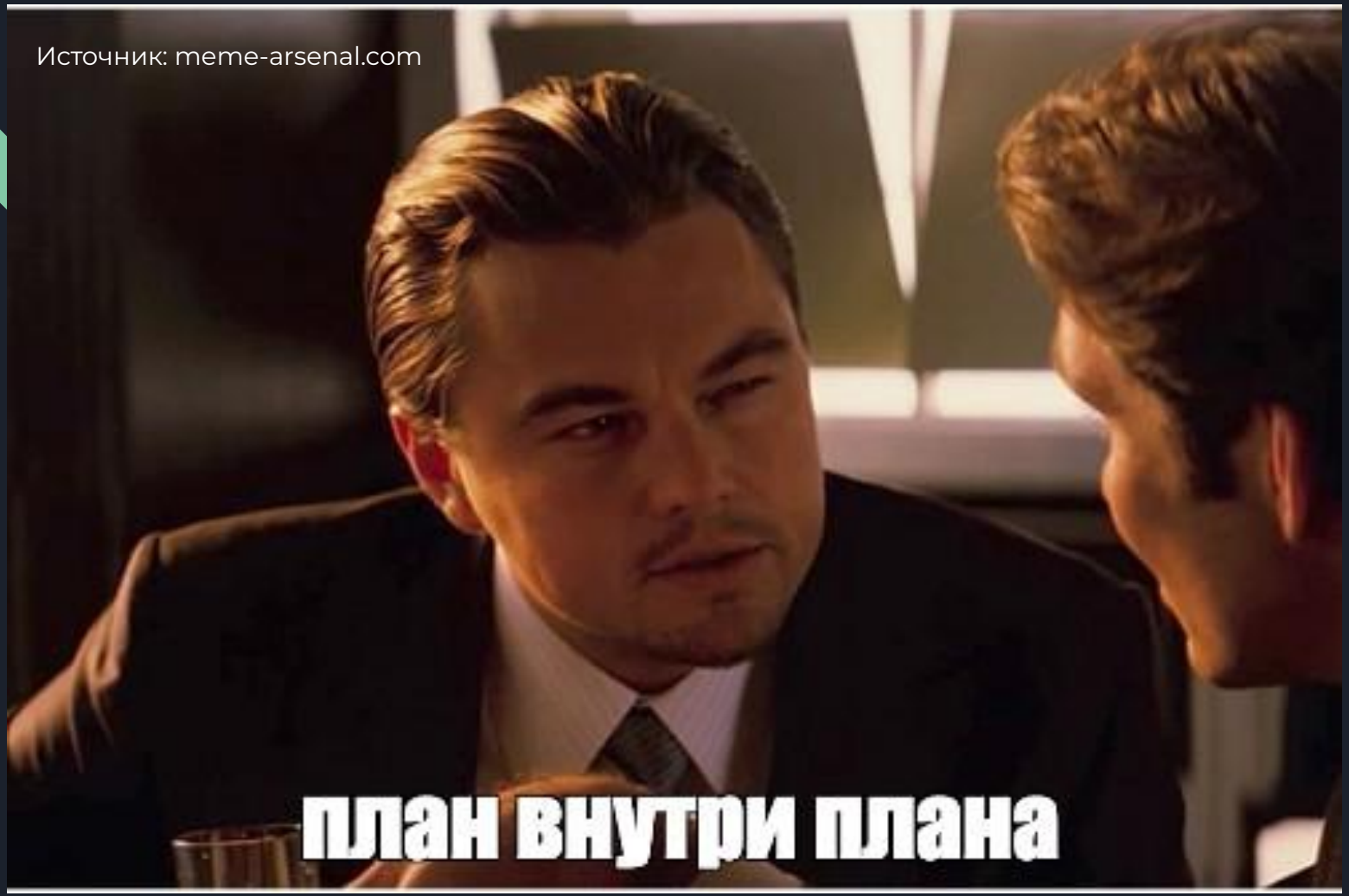
```
@Composable
fun AppNavigation(
    navController: NavHostController,
    settingsViewModel: SettingsViewModel,
    isDarkTheme: Boolean
) {
    NavHost(
        navController = navController,
        startDestination = TaskListRoute
    ) {
        composable<TaskListRoute> {
            TaskListScreen(
                onNavigateToTask = { taskId ->
                    navController.navigate(TaskDetailRoute(taskId))
                }
            )
        }
        ...
    }
}
```

Список задач

```
@Composable
fun TaskListScreen(
    viewModel: TaskListViewModel = koinViewModel(),
    onNavigateToTask: (Int) -> Unit
) {
    val tasks by viewModel.tasks.collectAsState()

    TaskListContent(
        tasks = tasks,
        onToggleCompletion = { viewModel.toggleTaskCompletion(it) },
        onNavigateToTask = onNavigateToTask
    )
}
```

План

1. С чего начнём?
2. Как подключить Compose-alpha для Аврора ОС
3. Разбираемся с Compose-Preview
4. Дополняем Koin
5. Compose Resources
6. Отступы, клавиатура, тема
7. Modal Bottom Sheet
8. Итоги


Источник: meme-arsenal.com
план внутри плана

Как подключить Compose-alpha для Аврора ОС

1. Окружение и константы
2. Настройка плагинов
3. Финальная настройка сборки

Окружение и константы

```
// Расположен по адресу
https://hub.mos.ru/auroraos/kotlin-multiplatform/aurora-maven

// Инструкции по установке
wget -O aurora-maven.tar.gz
https://hub.mos.ru/auroraos/kotlin-multiplatform/aurora-maven/-/archive/aurora-0.0.3/aurora-maven-aurora-0.0.3.tar.gz
tar -xzf aurora-maven.tar.gz
mkdir -p ~/.m2/
mv aurora-maven-* ~/.m2/repository
```

Окружение и константы

```
// Расположен по адресу
https://hub.mos.ru/auroraos/kotlin-multiplatform/aurora-maven

// Инструкции по установке
wget -O aurora-maven.tar.gz
https://hub.mos.ru/auroraos/kotlin-multiplatform/aurora-maven/-/archive/aurora-0.0.3/aurora-maven-aurora-0.0.3.tar.gz
tar -xzf aurora-maven.tar.gz
mkdir -p ~/.m2/
mv aurora-maven-* ~/.m2/repository
```

Окружение и константы

```
// Расположен по адресу
https://hub.mos.ru/auroraos/kotlin-multiplatform/aurora-maven

// Инструкции по установке
wget -O aurora-maven.tar.gz
https://hub.mos.ru/auroraos/kotlin-multiplatform/aurora-maven/-/archive/aurora-0.0.3/aurora-maven-aurora-0.0.3.tar.gz
tar -xzf aurora-maven.tar.gz
mkdir -p ~/.m2/
mv aurora-maven-* ~/.m2/repository
```

Окружение и константы

```
// gradle.properties

# Aurora
compose.aurora.enabled=false
compose.version.aurora=0.0.3-aurora
compose.version.upstream=1.10.2
```

Окружение и константы

```
// libs.versions.toml

[versions]
...
# Aurora
akPathInfo="0.0.1"
auroraBuildTools = "0.0.1"
spotless="7.0.3"
ktor-aurora = "3.1.2-aurora"
```

Окружение и константы

```
// libs.versions.toml

[libraries]
...
#Ktor
...
ktor-client-curl = { module = "io.ktor:ktor-client-curl", version.ref = "ktor-aurora" }

# Aurora
aurora-akPathInfo = { module = "ru.auroraos.kmp:ak-path-info", version.ref = "akPathInfo" }
```

Окружение и константы

```
// libs.versions.toml

[plugins]
...
# Aurora
auroraBuildTools = { id = "ru.auroraos.kmp.buildtools", version.ref = "auroraBuildTools" }
spotless = { id = "com.diffplug.spotless", version.ref = "spotless" }
```

Окружение и константы

```
// settings.gradle.kts
...
pluginManagement {
    repositories {
        google { ... }
        mavenCentral()
        mavenLocal()
        gradlePluginPortal()
    }
}
...
dependencyResolutionManagement {
    repositories {
        google { ... }
        mavenLocal()
        mavenCentral()
    }
}
```

Окружение и константы

```
// build.gradle.kts
plugins {
    alias(libs.plugins.androidApplication) apply false
    alias(libs.plugins.androidLibrary) apply false
    alias(libs.plugins.composeMultiplatform) apply false
    alias(libs.plugins.composeCompiler) apply false
    alias(libs.plugins.kotlinJvm) apply false
    alias(libs.plugins.kotlinMultiplatform) apply false
    alias(libs.plugins.ktor) apply false
    alias(libs.plugins.ksp) apply false
    alias(libs.plugins.androidx.room) apply false

    alias(libs.plugins.auroraBuildTools) apply false
    alias(libs.plugins.spotless) apply true
}
```

Окружение и константы

```
// composeApp/build.gradle.kts

val Project.auroraEnabled: Boolean
    get() = findProperty("compose.aurora.enabled") == "true"

plugins {
    alias(libs.plugins.kotlinMultiplatform)
    alias(libs.plugins.androidApplication)
    alias(libs.plugins.composeMultiplatform)
    alias(libs.plugins.composeCompiler)
    alias(libs.plugins.kotlinSerialization)
    alias(libs.plugins.ksp)
    alias(libs.plugins.androidx.room)

    alias(libs.plugins.auroraBuildTools)
}
```

Как подключить Compose-alpha для Аврора ОС

1. Окружение и константы
2. Настройка плагинов
3. Финальная настройка сборки

Настройка плагинов

```
// settings.gradle.kts

pluginManagement {
    repositories {
        ...
    }

    plugins {
        val composeVersion = if (providers.gradleProperty("compose.aurora.enabled").orNull == "true") {
            providers.gradleProperty("compose.version.aurora").get()
        } else {
            providers.gradleProperty("compose.version.upstream").get()
        }
        id("org.jetbrains.compose").version(composeVersion) apply false
    }
}
```

Настройка плагинов

```
// settings.gradle

pluginManagement {
    repositories {
        ...
    }

    plugins {
        val composeVersion = if (providers.gradleProperty("compose.aurora.enabled").orNull == "true") {
            providers.gradleProperty("compose.version.aurora").get()
        } else {
            providers.gradleProperty("compose.version.upstream").get()
        }
        id("org.jetbrains.compose").version(composeVersion) apply false
    }
}
```

Настройка плагинов

```
// settings.gradle

pluginManagement {
    repositories {
        ...
    }

    plugins {
        val composeVersion = if (providers.gradleProperty("compose.aurora.enabled").orNull == "true") {
            providers.gradleProperty("compose.version.aurora").get()
        } else {
            providers.gradleProperty("compose.version.upstream").get()
        }
        id("org.jetbrains.compose").version(composeVersion) apply false
    }
}
```

Настройка плагинов

```
// build.gradle.kts
plugins {
    alias(libs.plugins.androidApplication) apply false
    alias(libs.plugins.androidLibrary) apply false
    id("org.jetbrains.compose") apply false
    alias(libs.plugins.composeCompiler) apply false
    alias(libs.plugins.kotlinJvm) apply false
    alias(libs.plugins.kotlinMultiplatform) apply false
    alias(libs.plugins.ktor) apply false
    alias(libs.plugins.ksp) apply false
    alias(libs.plugins.androidx.room) apply false

    alias(libs.plugins.auroraBuildTools) apply false
    alias(libs.plugins.spotless) apply true
}

spotless {
    ...
}
```

Настройка плагинов

```
// build.gradle.kts
plugins {
    ...
    alias(libs.plugins.spotless) apply true
}

spotless {
    kotlin {
        ...
    }
    format("misc") {
        ...
    }
}
```

Настройка плагинов

```
// composeApp/build.gradle.kts

plugins {
    alias(libs.plugins.kotlinMultiplatform)
    alias(libs.plugins.androidApplication)
    id("org.jetbrains.compose")
    alias(libs.plugins.composeCompiler)
    alias(libs.plugins.kotlinSerialization)
    alias(libs.plugins.ksp)
    alias(libs.plugins.androidx.room)

    alias(libs.plugins.auroraBuildTools)
}
```

Настройка плагинов

```
// composeApp/build.gradle.kts

buildTools {
    rpm {
        id = "ru.den.writes.code.corporate_task_tracker"
        name = "Corporate Task Tracker"
        description = "Corpora Task Tracker built with KMP for Aurora OS"
        version = "0.0.1"
        permissions = listOf("Internet")
        libs3rdParty = listOf("maliit-glib")
        icons = projectDir.toPath().resolve("icons")
        resources = projectDir.toPath().resolve("src/commonMain/composeResources")
    }

    // Run on device
    run {
        ...
    }
}
```

Настройка плагинов

```
// composeApp/build.gradle.kts

buildTools {
    rpm {
        id = "ru.den.writes.code.corporate_task_tracker"
        name = "Corporate Task Tracker"
        description = "Corpora Task Tracker built with KMP for Aurora OS"
        version = "0.0.1"
        permissions = listOf("Internet")
        libs3rdParty = listOf("maliit-glib")
        icons = projectDir.toPath().resolve("icons")
        resources = projectDir.toPath().resolve("src/commonMain/composeResources")
    }

    // Run on device
    run {
        ...
    }
}
```

Настройка плагинов

```
// composeApp/build.gradle.kts

buildTools {
    rpm {
        ...
    }

    // Run on device
    run {
       host = "192.168.0.18"
       user = "defaultuser"
       port = 22
       validate = true
       sshKey = File(System.getProperty("user.home")).resolve(".ssh/qtc_id").toPath()
    }
}
```

Как подключить Compose-alpha для Аврора ОС

1. Окружение и константы
2. Настройка плагинов
3. Финальная настройка сборки

Финальная настройка сборки

```
// composeApp/build.gradle.kts

...
apply(from = "aurora-tasks.gradle.kts")
```

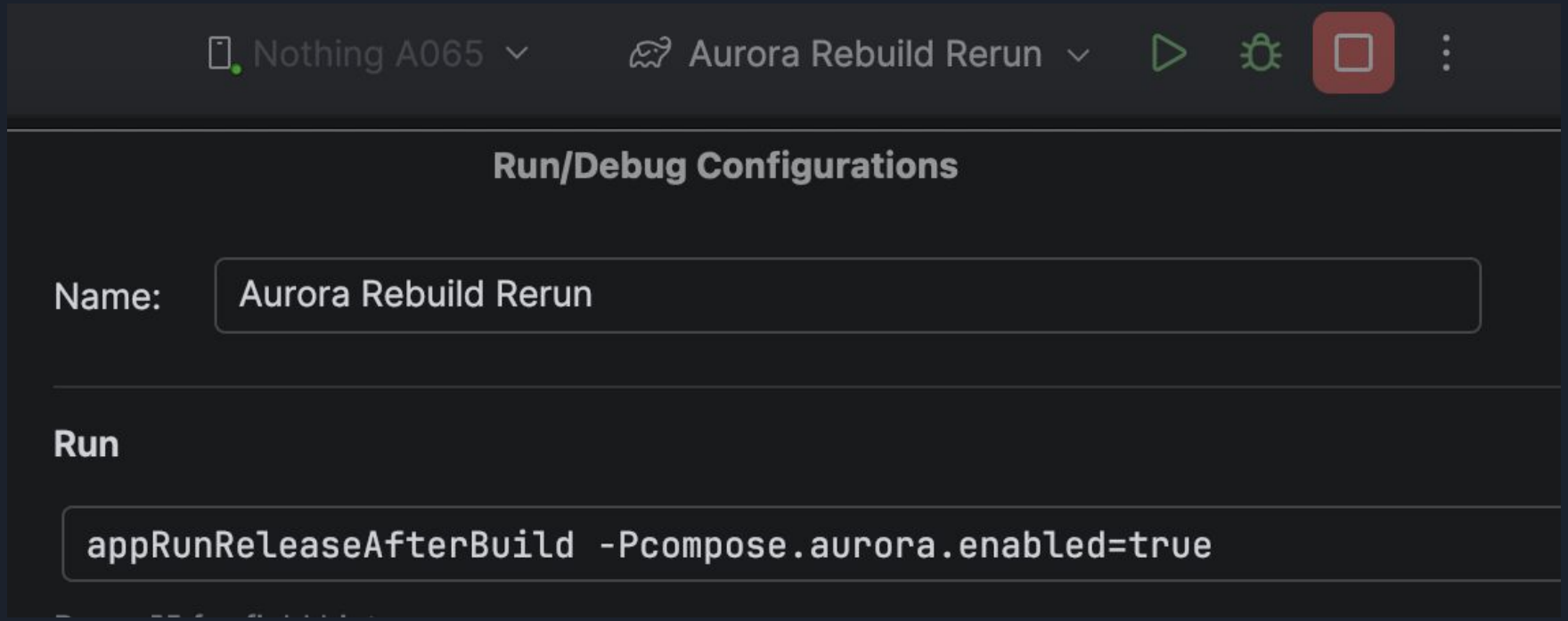
Финальная настройка сборки

```
// aurora-tasks.gradle.kts
val Project.auroraEnabled: Boolean
    get() = findProperty("compose.aurora.enabled") == "true"
if (auroraEnabled) {
    // 1. Init sysroot for dependency
    tasks.register("appSysroot") { ... }
    // 2. Build KMP application
    tasks.register("appBuildRelease") { ... }
    // 3. Build RPM package
    tasks.register("appPackageRelease") { ... }
    // 4. Run application
    tasks.register("appRunRelease") { .... }
    // Strict order of command execution
    val appPackageReleaseOrder = ...
    val appBuildReleaseOrder = ...
    tasks.register("appRunReleaseAfterBuild") {
        group = "Aurora Build-Tools"
        description = "Strict order: initSysroot > build KMP > build RPM > run."
        dependsOn("appSysroot")
        finalizedBy(appBuildReleaseOrder)
    }
```

Финальная настройка сборки

```
// aurora-tasks.gradle.kts
val Project.auroraEnabled: Boolean
    get() = findProperty("compose.aurora.enabled") == "true"
if (auroraEnabled) {
    // 1. Init sysroot for dependency
    tasks.register("appSysroot") { ... }
    // 2. Build KMP application
    tasks.register("appBuildRelease") { ... }
    // 3. Build RPM package
    tasks.register("appPackageRelease") { ... }
    // 4. Run application
    tasks.register("appRunRelease") { .... }
    // Strict order of command execution
    val appPackageReleaseOrder = ...
    val appBuildReleaseOrder = ...
    tasks.register("appRunReleaseAfterBuild") {
        group = "Aurora Build-Tools"
        description = "Strict order: initSysroot > build KMP > build RPM > run."
        dependsOn("appSysroot")
        finalizedBy(appBuildReleaseOrder)
    }
```

Финальная настройка сборки



Финальная настройка сборки

```
// composeApp/build.gradle.kts
if (!auroraEnabled) {
    ...
} else {
    listOf(
        linuxArm64(),
        linuxX64()
    ).forEach { target ->
        target.binaries {
            executable {
                entryPoint = "ru.den.writes.code.main"
                linkerOpts.addAll(buildTools.cmpLinkerOpts(
                    project = project,
                    targetName = target.name,
                    "Qt5Core",
                    "Qt5Network",
                ))
            }
        }
    }
}
```

Финальная настройка сборки

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)

        dependencies {
            if (!auroraEnabled) {
                ...
            } else {
                ...
            }
            // Доступно везде
            implementation(libs.androidx.lifecycle.runtimeCompose)
            implementation(libs.kotlinx.serialization.json)
            implementation(libs.koin.core)
            implementation(libs.ktor.client.core)
            implementation(libs.androidx.room.runtime)
            implementation(libs.androidx.sqlite.bundled)
            implementation(projects.shared)
        }
    }
```

Финальная настройка сборки

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)
        dependencies { ... }
    }

    if (!auroraEnabled) {
        androidMain.dependencies { ... }
        iosMain.dependencies { ... }
        commonTest.dependencies { ... }
    } else {
        linuxMain.dependencies {
           implementation(libs.aurora.akPathInfo)
           implementation(libs.ktor.client.curl)
        }
    }
}
```

Финальная настройка сборки

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)

        dependencies {
            if (!auroraEnabled) {
                ...
            } else {
                implementation(compose.runtime)
                implementation(compose.foundation)
                implementation(compose.material3)
                implementation(compose.ui)

                val auroraNavigation = compose.javaClass.getMethod("getNavigation").invoke(compose)
                implementation(auroraNavigation)
            }
            ...
        }
    }
}
```

Финальная настройка сборки

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)

        dependencies {
            if (!auroraEnabled) {
                ...
            } else {
                implementation(compose.runtime)
                implementation(compose.foundation)
                implementation(compose.material3)
                implementation(compose.ui)

                val auroraNavigation = compose.javaClass.getMethod("getNavigation").invoke(compose)
                implementation(auroraNavigation)
            }
            ...
        }
    }
}
```

Финальная настройка сборки

```
// Platform.linux.kt

class AuroraPlatform: Platform {
    override val name: String = "Аврора ОС"
}
actual fun getPlatform(): Platform = AuroraPlatform()
```

Финальная настройка сборки

```kotlin
// PlatformModule.linux.kt

actual fun platformModule() = module {
    single {
        getDatabaseBuilder()
    }
}
```

Финальная настройка сборки

```kotlin
// Database.linux.kt

fun getDatabaseBuilder(): RoomDatabase.Builder<AppDatabase> {
    val path = "${PathInfo.getAppData()}/my_room.db"
    return Room.databaseBuilder<AppDatabase>(
        name = path,
    )
}
```

Финальная настройка сборки

```kotlin
// Database.linux.kt

fun getDatabaseBuilder(): RoomDatabase.Builder<AppDatabase> {
    val path = "${PathInfo.getAppData()}/my_room.db"
    return Room.databaseBuilder<AppDatabase>(
        name = path,
    )
}
```

Финальная настройка сборки

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)

        dependencies {
            if (!auroraEnabled) {
                // compose & compose navigation
                // Не доступно для Аврора ОС
                implementation(libs.compose.uiToolingPreview)
                implementation(libs.koin.compose)
                implementation(libs.koin.compose.viewmodel)
                implementation(libs.androidx.lifecycle.viewmodelCompose)
                implementation(libs.compose.components.resources)
                implementation("org.jetbrains.compose.material:material-icons-extended:1.7.0")
            } else {
                ...
            }
            ...
        }
    }
```

Финальная настройка сборки

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)

        dependencies {
            if (!auroraEnabled) {
                // compose & compose navigation
                // Не доступно для Аврора ОС
                implementation(libs.compose.uiToolingPreview)
                implementation(libs.koin.compose)
                implementation(libs.koin.compose.viewmodel)
                implementation(libs.androidx.lifecycle.viewmodelCompose)
                implementation(libs.compose.components.resources)
                implementation("org.jetbrains.compose.material:material-icons-extended:1.7.0")
            } else {
                ...
            }
            ...
        }
    }
```

Финальная настройка сборки

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)

        dependencies {
            if (!auroraEnabled) {
                // compose & compose navigation
                // Не доступно для Аврора ОС
                implementation(libs.compose.uiToolingPreview)
                implementation(libs.koin.compose)
                implementation(libs.koin.compose.viewmodel)
                implementation(libs.androidx.lifecycle.viewmodelCompose)
                implementation(libs.compose.components.resources)
                implementation("org.jetbrains.compose.material:material-icons-extended:1.7.0")
            } else {
                ...
            }
            ...
        }
    }
```

Финальная настройка сборки

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)

        dependencies {
            if (!auroraEnabled) {
                // compose & compose navigation
                // Не доступно для Аврора ОС
                implementation(libs.compose.uiToolingPreview)
                implementation(libs.koin.compose)
                implementation(libs.koin.compose.viewmodel)
                implementation(libs.androidx.lifecycle.viewmodelCompose)
                implementation(libs.compose.components.resources)
                implementation("org.jetbrains.compose.material:material-icons-extended:1.7.0")
            } else {
                ...
            }
            ...
        }
    }
```

План

1. С чего начнём?
2. Как подключить Compose-alpha для Аврора ОС
3. Разбираемся с Compose-Preview
4. Дополняем Koin
5. Compose Resources
6. Отступы, клавиатура, тема
7. Modal Bottom Sheet
8. Итоги

Источник: meme-arsenal.com

Разбираемся с Compose-Preview

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)

        if (auroraEnabled) {
           kotlin.srcDir("src/previewStub/kotlin")
        }

        dependencies {
            if (!auroraEnabled) {
                ...
            } else {
                ...
            }
        ...
    }
    ...
}
```

Разбираемся с Compose-Preview

```
// composeApp/src/previewStub/kotlin/androidx/compose/ui/tooling/preview/Preview.kt
package androidx.compose.ui.tooling.preview

@Target(
    AnnotationTarget.ANNOTATION_CLASS,
    AnnotationTarget.FUNCTION
)
@Retention(AnnotationRetention.BINARY)
annotation class Preview
```

Разбираемся с Compose-Preview

```
// composeApp/src/previewStub/kotlin/androidx/compose/ui/tooling/preview/Preview.kt
package androidx.compose.ui.tooling.preview

@Target(
    AnnotationTarget.ANNOTATION_CLASS,
    AnnotationTarget.FUNCTION
)
@Retention(AnnotationRetention.BINARY)
annotation class Preview
```

План

1. С чего начнём?
2. Как подключить Compose-alpha для Аврора ОС
3. Разбираемся с Compose-Preview
4. Дополняем Koin
5. Compose Resources
6. Отступы, клавиатура, тема
7. Modal Bottom Sheet
8. Итоги

Источник: meme-arsenal.com

Дополняем Koin

```
// composeApp/build.gradle.kts
sourceSets {
    commonMain {
        kotlin.srcDir(generateAppConfigTask)

        if (auroraEnabled) {
           kotlin.srcDir("src/previewStub/kotlin")
           kotlin.srcDir("src/koinCompat/kotlin")
        }

        dependencies {
            if (!auroraEnabled) {
                ...
            } else {
                ...
            }
        ...
    }
    ...
}
```

Дополняем Koin

```
// composeApp/src/koinCompat/kotlin/org/koin/compose/viewmodel/KoinViewModel.kt
package org.koin.compose.viewmodel

import androidx.compose.runtime.Composable
import androidx.lifecycle.ViewModel
import org.koin.core.annotation.KoinInternalApi
import org.koin.core.parameter.ParametersDefinition
import org.koin.core.qualifier.Qualifier
import org.koin.mp.KoinPlatformTools

@OptIn(KoinInternalApi::class)
@Composable
inline fun <reified T : ViewModel> koinViewModel(
    qualifier: Qualifier? = null,
    noinline parameters: ParametersDefinition? = null,
): T {
    return KoinPlatformTools.defaultContext().get().get<T>(qualifier, parameters)
}
```

Дополняем Koin

```
// composeApp/src/koinCompat/kotlin/org/koin/compose/KoinApplication.kt
package org.koin.compose

@OptIn(KoinInternalApi::class)
@Composable
fun KoinApplication(
    application: KoinAppDeclaration, //Better to directly use KoinConfiguration class
    content: @Composable () -> Unit
) {
    val koin = rememberKoinApplication(application)
    CompositionLocalProvider(
        LocalKoinApplication provides ComposeContextWrapper(koin) { getDefaultKoinContext() },
        LocalKoinScope provides ComposeContextWrapper(koin.scopeRegistry.rootScope) { getDefaultRootScope() },
        content = content
    )
}
```

Дополняем Koin

```
// composeApp/src/koinCompat/kotlin/org/koin/compose/KoinApplication.kt
@Composable
@KoinInternalApi
inline fun rememberKoinApplication(noinline koinAppDeclaration: KoinAppDeclaration): Koin {
    val wrapper = remember(koinAppDeclaration) {
        CompositionKoinApplicationLoader(koinApplication(koinAppDeclaration))
    }
    return wrapper.koin ?: error("Koin context has not been initialized in rememberKoinApplication")
}

@OptIn(KoinInternalApi::class)
val LocalKoinApplication: ProvidableCompositionLocal<ComposeContextWrapper<Koin>> =
    compositionLocalOf { ComposeContextWrapper(getDefaultKoinContext()) { getDefaultKoinContext() } }

@OptIn(KoinInternalApi::class)
val LocalKoinScope: ProvidableCompositionLocal<ComposeContextWrapper<Scope>> =
    compositionLocalOf { ComposeContextWrapper(getDefaultRootScope()) { getDefaultRootScope() } }
```

Дополняем Koin

```
// composeApp/src/koinCompat/kotlin/org/koin/compose/KoinApplication.kt
private fun getDefaultKoinContext() = KoinPlatform.getKoin()

@OptIn(KoinInternalApi::class)
private fun getDefaultRootScope() = KoinPlatform.getKoin().scopeRegistry.rootScope
```

Compose Resources

```
composeApp
└── src
    └── koinCompat
        └── kotlin.org.koin
            └── compose
            |   ├── application
            |   |   └── CompositionKoinApplicationLoader.kt
            |   ├── viewmodel
            |   |   └── KoinViewModel.kt
            |   ├── ComposeContextWrapper.kt
            |   └── KoinApplication.kt
            └── core
                └── module
                    └── module
                        └── dsl
                            └── ViewModelOf.kt
```

План

1. С чего начнём?
2. Как подключить Compose-alpha для Аврора ОС
3. Разбираемся с Compose-Preview
4. Дополняем Koin
5. Compose Resources
6. Отступы, клавиатура, тема
7. Modal Bottom Sheet
8. Итоги

Источник: meme-arsenal.com

Compose Resources

- У нас есть много типов ресурсов:
  - текстовые (локализация, интренационализация, числительные)
  - массивы
  - растровые изображения (jpg, png)
  - векторные изображения (xml, svg)

Compose Resources

- У нас есть много типов ресурсов:
  - **текстовые** (локализация, интренационализация, числительные)
  - массивы
  - растровые изображения (jpg, png)
  - **векторные изображения** (**xml**, svg)

Compose Resources

- Нужно выносить в отдельный модуль, из-за конфликта имён с Koin - и там и там есть интерфейс Qualifier

Compose Resources

```
// compResAuroraCompat/build.gradle.kts
plugins {
    alias(libs.plugins.kotlinMultiplatform)
    id("org.jetbrains.compose")
    alias(libs.plugins.composeCompiler)
}
kotlin {
    linuxArm64()
    linuxX64()
    sourceSets {
        commonMain.dependencies {
            implementation(compose.runtime)
            implementation(compose.foundation)
            implementation(compose.material3)
            implementation(compose.ui)
        }
        linuxMain.dependencies {
            implementation(libs.aurora.akPathInfo)
        }
    }
}
```

Compose Resources

```
// compResAuroraCompat/build.gradle.kts
plugins {
    alias(libs.plugins.kotlinMultiplatform)
    id("org.jetbrains.compose")
    alias(libs.plugins.composeCompiler)
}
kotlin {
    linuxArm64()
    linuxX64()
    sourceSets {
        commonMain.dependencies {
            implementation(compose.runtime)
            implementation(compose.foundation)
            implementation(compose.material3)
            implementation(compose.ui)
        }
        linuxMain.dependencies {
            implementation(libs.aurora.akPathInfo)
        }
    }
}
```

Compose Resources

```
// composaApp/build.gradle.kts
buildTools {
    rpm {
        ...
        resources =
projectDir.toPath().resolve("build/generated/compose/resourceGenerator/preparedResources/commonMain/composeResources")
    }
    run {
        ...
    }
}

compose.resources {
    generateResClass = always
}
```

Compose Resources

```
// composaApp/build.gradle.kts
buildTools {
    rpm {
        ...
        resources =
projectDir.toPath().resolve("build/generated/compose/resourceGenerator/preparedResources/commonMain/composeResources")
    }
    run {
        ...
    }
}

compose.resources {
    generateResClass = always
}
```

Compose Resources

```
compResAuroraCompat
└── src
    ├── commonMain
    │   └── kotlin
    │       └── org.jetbrains.compose.resources
    │           ├── plural
    │           │   └── PluralCategory
    │           └── vector
    │               ├── xmldom
    │               │   ├── Element
    │               │   ├── Node
    │               │   ├── NodeList
    │               │   ├── ValueParser.kt
    │               │   └── XmlVectorParser.kt
    │               ├── ...
    └── linuxMain
        └── kotlin
            └── org.jetbrains.compose.resources
                ├── ...
```

Compose Resources

```
compResAuroraCompat
└── src
    ├── commonMain
    │   └── kotlin
    │       └── org.jetbrains.compose.resources
    │           ├── ...
    │               ├── FontResources.kt
    │               ├── ImageResources.kt
    │               ├── PluralStringResources.kt
    │               ├── Qualifer.kt
    │               ├── Resource.kt
    │               ├── ResourceCaches.kt
    │               ├── ResourceEnvironment.kt
    │               ├── ResourceReader.kt
    │               ├── ResourceState.blocking.kt
    │               ├── StringArrayResources.kt
    │               ├── StringResources.kt
    │               └── StringResourcesUtils.kt
    └── ...
```

Compose Resources

```
compResAuroraCompat
└── src
    ├── commonMain
    │   └── kotlin
    │       └── org.jetbrains.compose.resources
    │           ├── plural
    │           └── vector
    │               ├── xmldom
    │               │   ├── ...
    │               ├── ...
    └── linuxMain
        └── kotlin
            └── org.jetbrains.compose.resources
                ├── vector.xmldom
                │   └── Element.linux.kt
                ├── ImageResources.linux.kt
                ├── ResourceEnvironment.linux.kt
                └── ResourceReader.linux.kt
```

Compose Resources

```
// ResourceReader.linux.kt
package org.jetbrains.compose.resources

import androidx.compose.runtime.Composable
import androidx.compose.runtime.ProvidableCompositionLocal

internal actual fun getPlatformResourceReader(): ResourceReader {
    TODO("Not yet implemented")
}

internal actual val ProvidableCompositionLocal<ResourceReader>.currentOrPreview: ResourceReader
    @Composable
    get() = TODO("Not yet implemented")
```

Compose Resources

```
// ResourceReader.linux.kt
package org.jetbrains.compose.resources

import androidx.compose.runtime.Composable
import androidx.compose.runtime.ProvidableCompositionLocal

internal actual fun getPlatformResourceReader(): ResourceReader {
    TODO("Not yet implemented")
}

internal actual val ProvidableCompositionLocal<ResourceReader>.currentOrPreview: ResourceReader
    @Composable
    get() = current
```

Compose Resources

```kotlin
// ResourceReader.linux.kt
@OptIn(ExperimentalForeignApi::class)
fun PathInfo.listDirectoryContents(dirPath: String): List<String> {
    val dirPtr = opendir(dirPath)
        ?: throw IllegalArgumentException("Не удалось открыть папку: $dirPath")
    val contents = mutableListOf<String>()
    try {
        while (true) {
            val entry = readdir(dirPtr) ?: break
            val fileName = entry.pointed.d_name.toKString()
            if (fileName != "." && fileName != "..") {
                contents.add(fileName)
            }
        }
    } finally {
        closedir(dirPtr)
    }
    return contents
}
```

Compose Resources

```
// ResourceReader.linux.kt
@OptIn(ExperimentalForeignApi::class)
fun PathInfo.listDirectoryContents(dirPath: String): List<String> {
    val dirPtr = opendir(dirPath)
        ?: throw IllegalArgumentException("Не удалось открыть папку: $dirPath")
    val contents = mutableListOf<String>()
    try {
        while (true) {
            val entry = readdir(dirPtr) ?: break
            val fileName = entry.pointed.d_name.toKString()
            if (fileName != "." && fileName != "..") {
                contents.add(fileName)
            }
        }
    } finally {
        closedir(dirPtr)
    }
    return contents
}
```

Compose Resources

```kotlin
// ResourceReader.linux.kt
@OptIn(ExperimentalForeignApi::class)
fun PathInfo.getResourceBytes(
    path: String,
    offset: Long,
    size: Long
): ByteArray {
    val filePath = getPathResources() + "/${path.replace("/", "_")}"
    val file = fopen(filePath, "rb") ?: throw NoSuchElementException("Cannot open file: $filePath")

    try {
        if (size > Int.MAX_VALUE) {
            throw IllegalArgumentException("Requested size is too large for a single ByteArray")
        }

        if (fseek(file, offset, SEEK_SET) != 0) {
            throw IllegalArgumentException("Failed to seek to offset $offset in file: $filePath")
        }

        val bytes = ByteArray(size.toInt())
```

Compose Resources

```
// ResourceReader.linux.kt
@OptIn(ExperimentalForeignApi::class)
fun PathInfo.getResourceBytes( ..): ByteArray { ...
        val bytes = ByteArray(size.toInt())
        if (size > 0L) {
            memScoped {
                val pinned = bytes.pin()
                try {
                    val bytesRead = fread(pinned.addressOf(0), 1u, size.toULong(), file).toLong()
                    if (bytesRead != size) {
                        throw IllegalArgumentException(..)
                    }
                } finally {
                    pinned.unpin()
                }
            }
        }
        return bytes
    } finally {
        fclose(file)
    }
```

Compose Resources

```
// ResourceReader.linux.kt
internal object DefaultLinuxResourceReader : ResourceReader {
    @OptIn(ExperimentalForeignApi::class)
    override suspend fun readPart(
        path: String,
        offset: Long,
        size: Long
    ): ByteArray { ... }

    private val composeResourcesDir: String by lazy { ... }
    @OptIn(ExperimentalForeignApi::class)
    override suspend fun read(path: String): ByteArray { ... }
    override fun getUri(path: String): String { ... }
    private fun getPathInBundle(path: String): String { ... }
    private fun findComposeResourcesPath(): String { ...}
}
```

Compose Resources

```
// ResourceReader.linux.kt
internal object DefaultLinuxResourceReader : ResourceReader {
    @OptIn(ExperimentalForeignApi::class)
    override suspend fun readPart(
        path: String,
        offset: Long,
        size: Long
    ): ByteArray {
        val cleanPath = path.substringAfter(".generated.resources/")
        val res = try {
            PathInfo.getResourceBytes(cleanPath, offset, size)
        } catch (e: Exception) {
            throw e
        }
        return res
    }

    ...
}
```

Compose Resources

```
// ResourceReader.linux.kt
internal object DefaultLinuxResourceReader : ResourceReader {
    @OptIn(ExperimentalForeignApi::class)
    override suspend fun readPart(
        path: String,
        offset: Long,
        size: Long
    ): ByteArray {
        val cleanPath = path.substringAfter(".generated.resources/")
        val res = try {
            PathInfo.getResourceBytes(cleanPath, offset, size)
        } catch (e: Exception) {
            throw e
        }
        return res
    }

    ...
}
```

Compose Resources

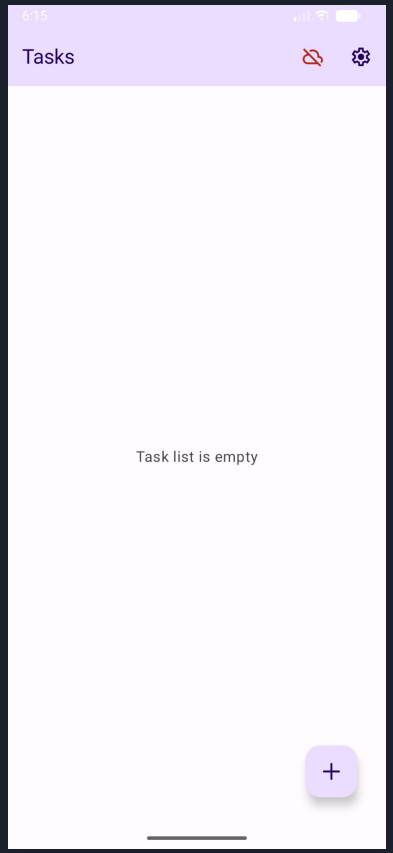
```
// ResourceReader.linux.kt
internal object DefaultLinuxResourceReader : ResourceReader {
    @OptIn(ExperimentalForeignApi::class)
    override suspend fun read(path: String): ByteArray {
        val cleanPath = path.substringAfter(".generated.resources/")
        val res = try {
            PathInfo.getResourceBytes(cleanPath)
        } catch (e: Exception) {
            throw e
        }
        return res
    }

    ...
}
```

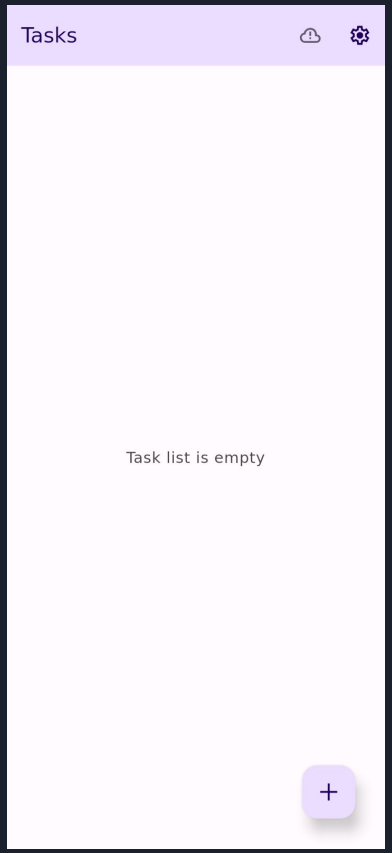
Compose Resources

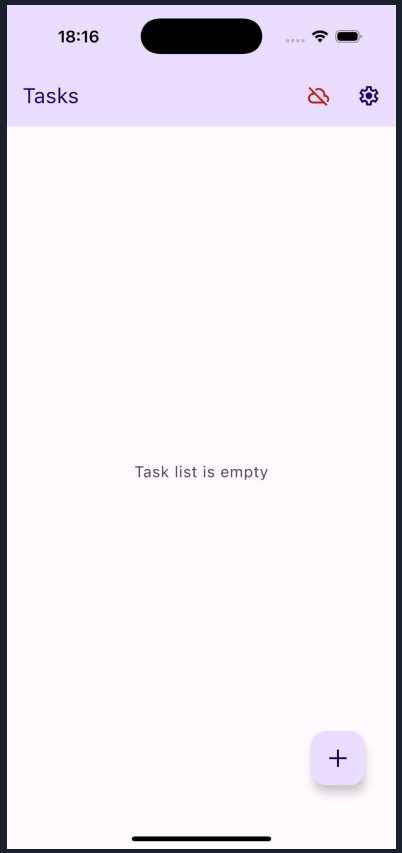
```
// ImageResources.linux.kt

internal actual fun ByteArray.toXmlElement() = parse(decodeToString())
```

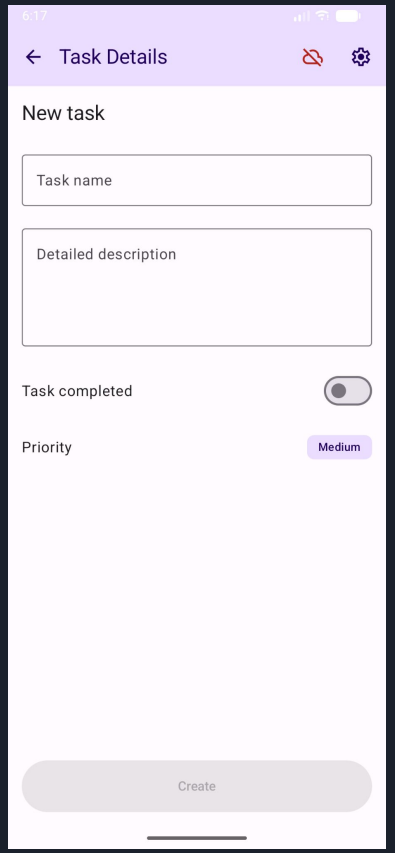
Compose Resources

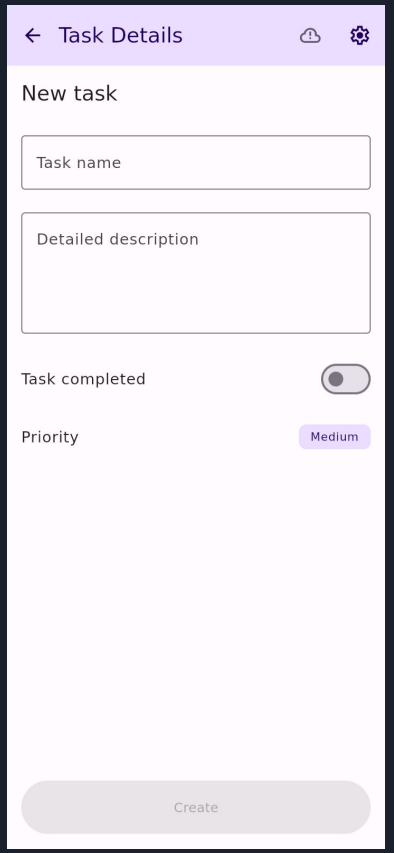
```
// Element.linux.kt
internal class ElementImpl(
    override val localName: String,
    override val nodeName: String,
    override val namespaceURI: String,
    val prefixMap: Map<String, String>,
    val attributes: Map<String, String>
) : Element {
    override var textContent: String? = null
    var childs = mutableListOf<Node>()
    override val childNodes: NodeList
        get() = object : NodeList { ... }
    override fun getAttributeNS(nameSpaceURI: String, localName: String): String { ... }
    override fun getAttribute(name: String): String = attributes[name] ?: ""
    override fun lookupPrefix(namespaceURI: String): String = prefixMap[namespaceURI] ?: ""
}

internal fun parse(xml: String): Element {
    ...
}
```

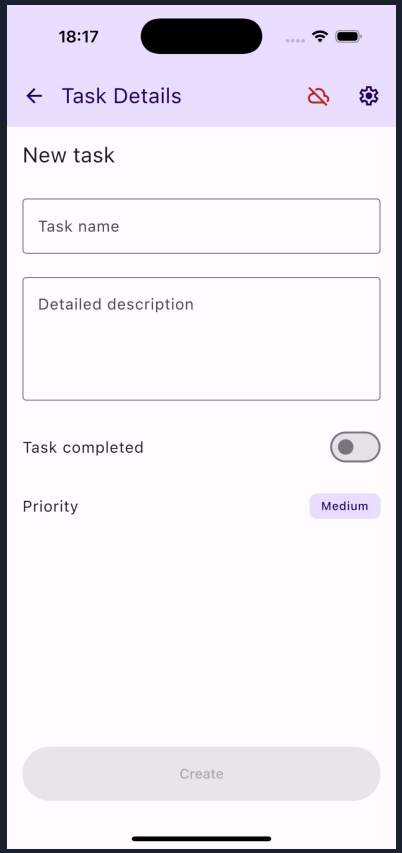

Tasks
Task list is empty

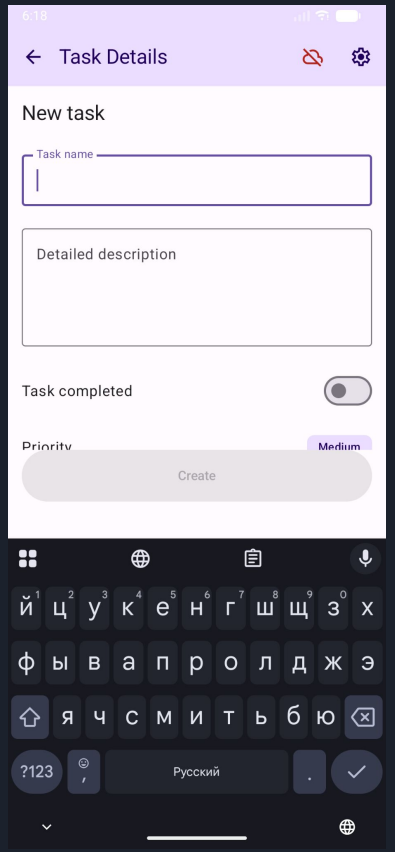

Tasks
Task list is empty

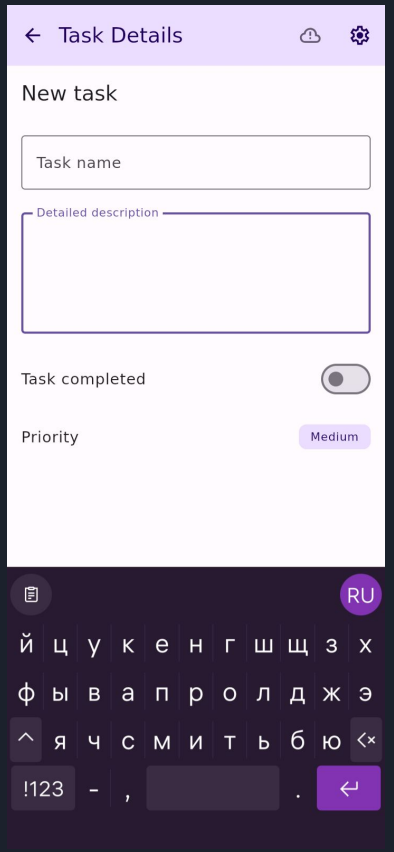

18:16
Tasks
Task list is empty

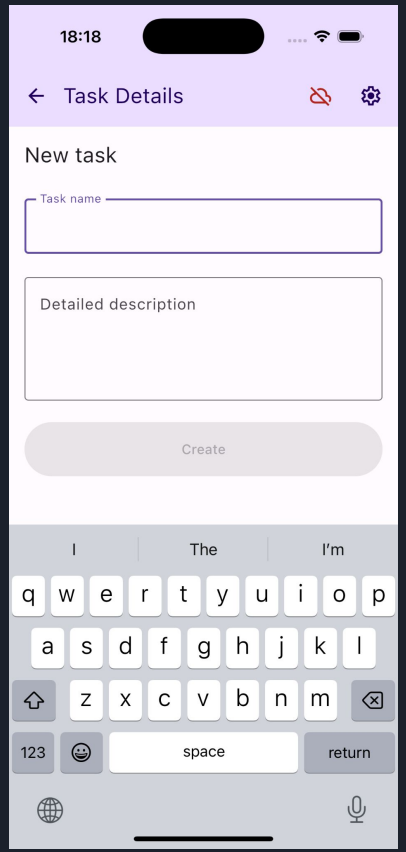

Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create


Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create


18:17
Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create


Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create
Русский


Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
RU


18:18
Task Details
New task
Task name
Detailed description
Create
I
The
I'm
space
return

План

1. С чего начнём?
2. Как подключить Compose-alpha для Аврора ОС
3. Разбираемся с Compose-Preview
4. Дополняем Koin
5. Compose Resources
6. Отступы, клавиатура, тема
7. Modal Bottom Sheet
8. Итоги

Отступы, клавиатура, тема

```
// linuxMain/Main.kt

fun main() = application {
    App()
}
```

Отступы, клавиатура, тема

```
// linuxMain/Main.kt

@Composable
fun HandleStatusBarOffset(content: @Composable (() -> Unit)) {
    Box(
        modifier = Modifier
            .fillMaxSize()
            .padding(top = 48.dp)
    ) {
        content()
    }
}

fun main() = application {
    HandleStatusBarOffset {
        App()
    }
}
```

Отступы, клавиатура, тема

```
// commonMain/App.kt
@Composable
fun App(...) {
    KoinApplication(...) {
        ...
        AppTheme(...) {
            ...
            Scaffold(...) { paddingValues ->
                Box(
                    modifier = Modifier
                        .fillMaxSize()
                        .imePadding()
                        .padding(paddingValues)
                        .background(MaterialTheme.colorScheme.background)
                ) {
                    ...
                }
            }
        }
    }
}
```

Отступы, клавиатура, тема

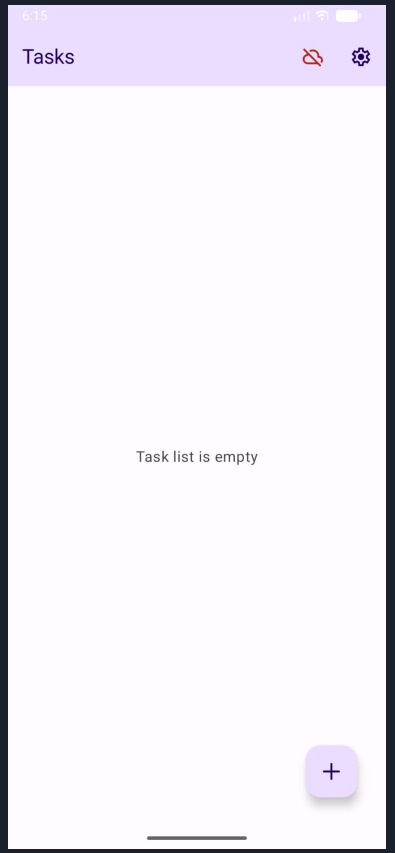
```
// common
expect fun Modifier.fillMaxSizeModifierWithKbdHandling(): Modifier
// android / ios
actual fun Modifier.fillMaxSizeModifierWithKbdHandling() = this
    .fillMaxSize()
    .imePadding()
// aurora
actual fun Modifier.fillMaxSizeModifierWithKbdHandling() = composed {
    val kbdHeight = forceFetchKbdHeight().collectAsState(0).value
    val windowHeight = LocalWindowInfo.current.containerSize.height

    val fraction = if (kbdHeight != 0)
        (windowHeight - kbdHeight).toFloat() / windowHeight
    else
        1.0f

    this
        .fillMaxWidth()
        .fillMaxHeight(fraction)
}
```

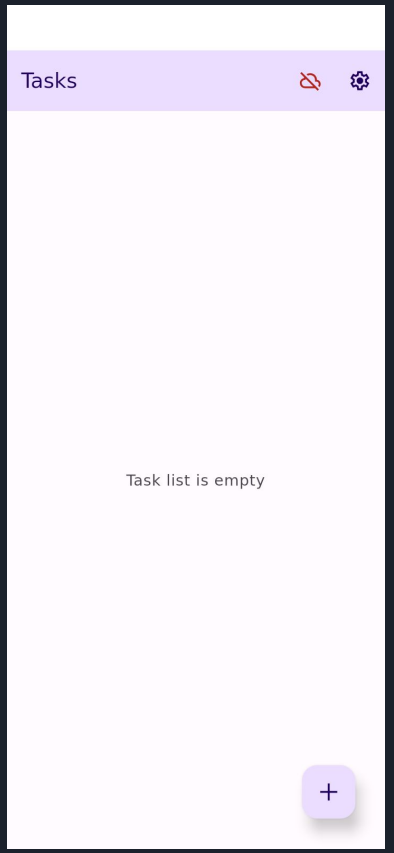
Отступы, клавиатура, тема

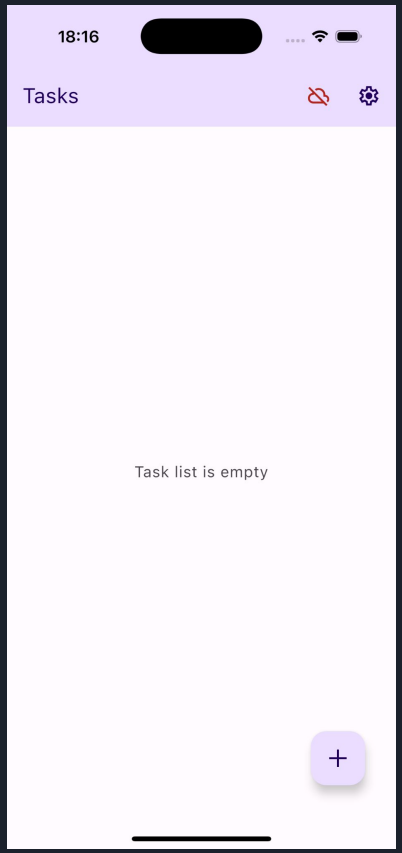
```
// common
expect fun Modifier.fillMaxSizeModifierWithKbdHandling(): Modifier
// android / ios
actual fun Modifier.fillMaxSizeModifierWithKbdHandling() = this
    .fillMaxSize()
    .imePadding()
// aurora
actual fun Modifier.fillMaxSizeModifierWithKbdHandling() = composed {
    val kbdHeight = forceFetchKbdHeight().collectAsState(0).value
    val windowHeight = LocalWindowInfo.current.containerSize.height

    val fraction = if (kbdHeight != 0)
        (windowHeight - kbdHeight).toFloat() / windowHeight
    else
        1.0f

    this
        .fillMaxWidth()
        .fillMaxHeight(fraction)
}
```

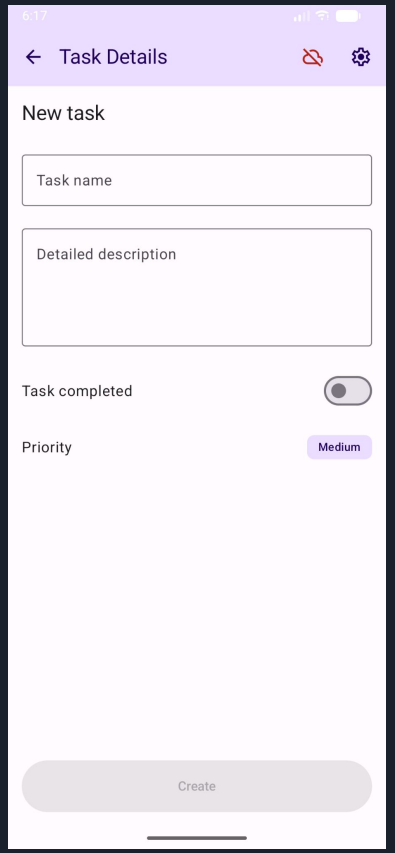
Отступы, клавиатура, тема

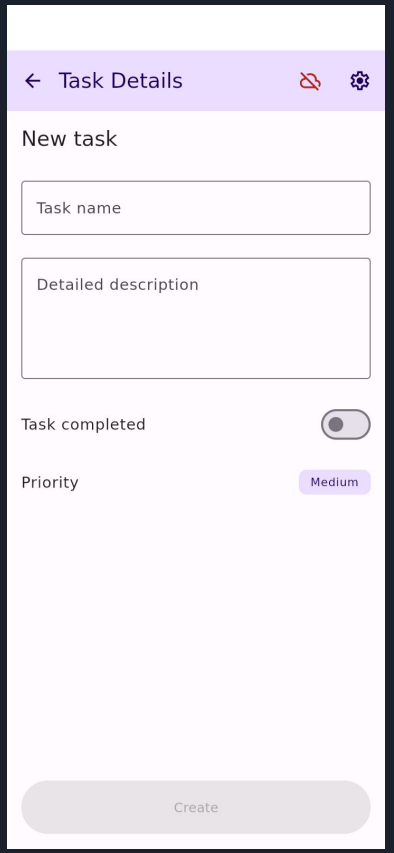
```
private fun forceFetchKbdHeight(): Flow<Int> {
    return flow {
        while (true) {
            if (Keyboard.isOpen()) {
                val height = Keyboard.height().toInt()
                emit(height)
            } else {
                emit(0)
            }
            delay(100)
        }
    }
}
```

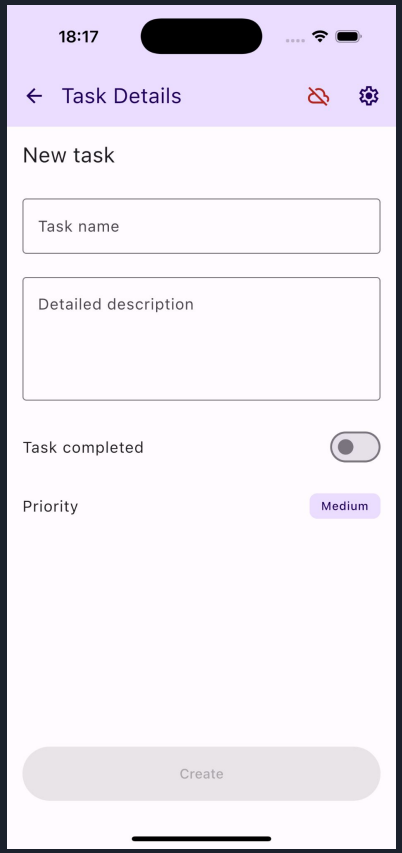

Tasks
Task list is empty

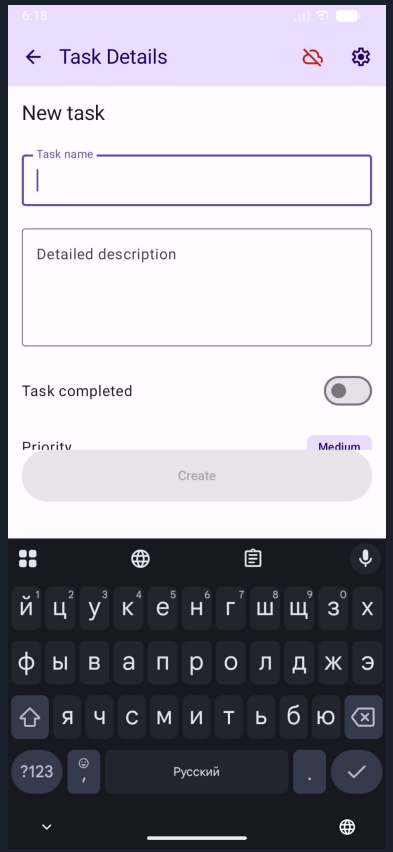

Tasks
Task list is empty

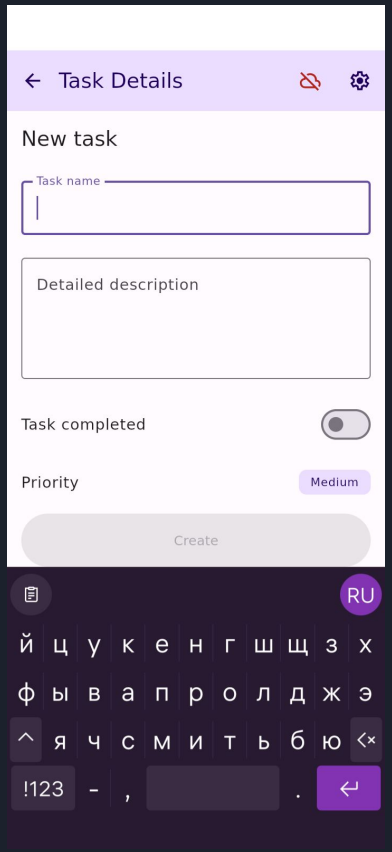

18:16
Tasks
Task list is empty

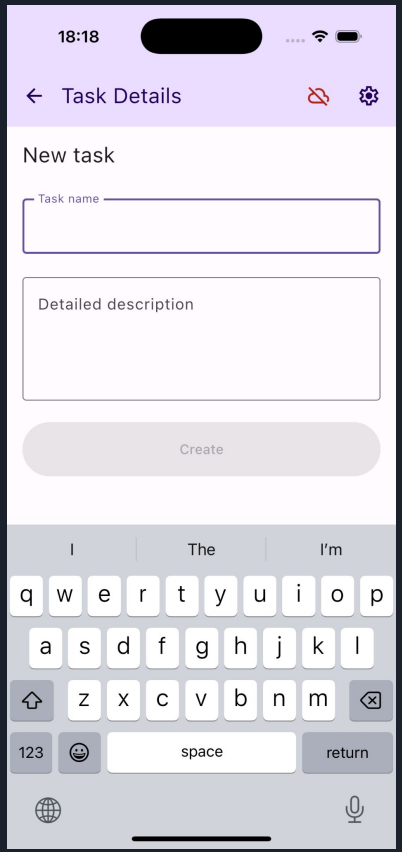

Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create


Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create


18:17
Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create


Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create
Русский


Task Details
New task
Task name
Detailed description
Task completed
Priority
Medium
Create
RU


18:18
Task Details
New task
Task name
Detailed description
Create
I
The
I'm
space
return

Отступы, клавиатура, тема

```
// AppModules.kt

val appModule = module {
    ...

    // ViewModels
    viewModelOf(::ServerStatusViewModel)
    viewModelOf(::SettingsViewModel)
}
```

Отступы, клавиатура, тема

```
// AppModules.kt

val appModule = module {
    ...

    // ViewModels
    viewModelOf(::ServerStatusViewModel)
    single { SettingsViewModel() }
}
```

План

1. С чего начнём?
2. Как подключить Compose-alpha для Аврора ОС
3. Разбираемся с Compose-Preview
4. Дополняем Koin
5. Compose Resources
6. Работа с фокусом и клавиатурой
7. Modal Bottom Sheet
8. Итоги

Modal Bottom Sheet

- Modal Bottom Sheet использует MainUIDispatcher из Skiko
- Compose-alpha для Аврора ОС использует свою версию Skiko
- При её портировании пока что MainUIDispatcher там не реализовали
- В сообществе Aurora Developers уже сделали порт, в котором эта проблема исправлена, так что можно воспользоваться

План

1. С чего начнём?
2. Как подключить Compose-alpha для Аврора ОС
3. Разбираемся с Compose-Preview
4. Дополняем Koin
5. Compose Resources
6. Работа с фокусом и клавиатурой
7. Modal Bottom Sheet
8. Итоги


Источник: meme-arsenal.com
ЭТО
УСПЕХ

Итоги

Время реально пришло, есть всё, для того, чтобы взять и попробовать:

- SDK, симулятор, всё работает даже на маках M чипах
- есть собственно порт Compose
- есть минимальный набор библиотек
- недостающее можно докрутить тем или иным способом, или как то обойти

Итоги

- Работа с отступами и клавиатурой
- MainUIDispatcher из Skiko и Modal Bottom Sheet
- Compose Resources
- Прочие фиксы стабильности
- Улучшения по грэдл
- Развитие интеграции со сторонними библиотеками

Спасибо за внимание