Миграция больших приложений на KMP

Павел Шорохов 2025

Мало информации

Влияние на **приложение**

- Производительность
- Память
- Размер приложения

Влияние на сборку

- Время сборки
- Размер DerivedData

Влияние на разработку

- KMP генерирует не удобный API. Можно ли с этим что-то сделать?
- KMP генерирует Objective-C, а разработчики пишут на Swift. Это может вызвать какие-то проблемы?
- Удобно ли отлаживать?

Прочие вопросы

- Ограничение umbrella-модуля: что это значит на практике?
- Как использовать нативные Swift-зависимости?
- Какой тип интеграции выбрать?
- Сможем ли собрать SPM или Cocoapods зависимость?

Проблемы с Kotlin Multiplatform в iOS

1. Какую интеграцию выбрать?
2. Что делать с ограничением umbrella-модуля?
3. Что делать с неудобным API?

Как мигрировать **Android** на KMP?

Заменить JVM-библиотеки на Multiplatform

- Gson → kotlinx-serialization
- Thread → kotlinx-coroutines
- System.currentTimeMillis() → kotlinx-datetime
- java.util.concurrent.atomic.* → atomicfu
- java.io.* → okio
- UUID → expect/actual (или Uuid с Kotlin 2.0.20)

Миграция Android на KMP

- Использовать только Multiplatform библиотеки

- Отслеживать изменение размера приложения

Как мигрировать iOS на KMP?

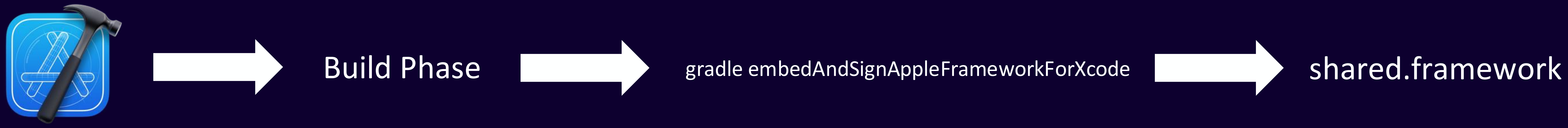
Проблема 1: Какую интеграцию выбрать?

Виды интеграций KMP в iOS

	No package manager	With SPM	With CocoaPods
Local	**Direct Integration**	**SPM Local**	**CocoaPods Local**
Remote	—	**SPM Remote**	**CocoaPods Remote**

Direct Integration

Direct Integration — Схема

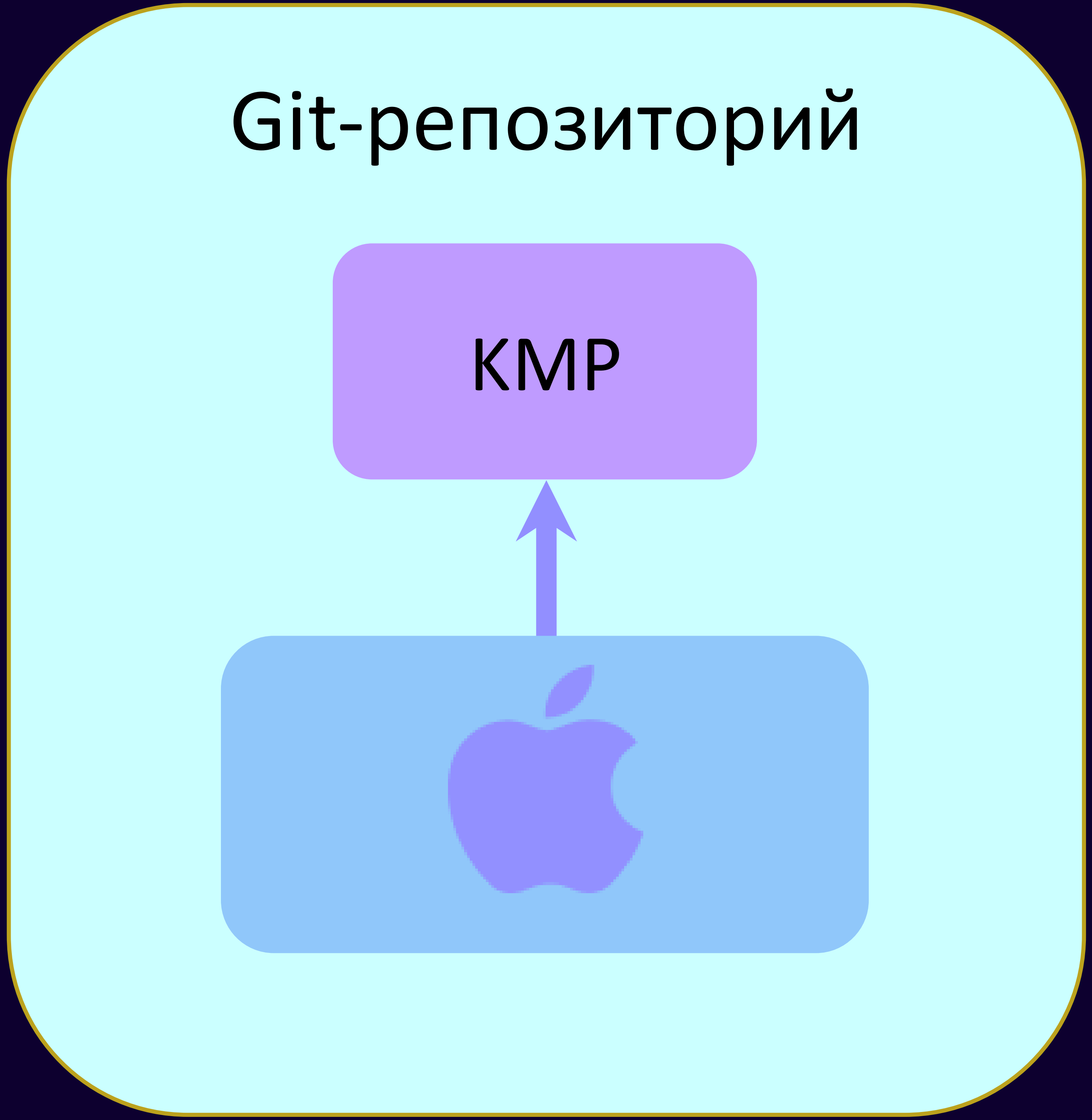


Build Phase

gradle embedAndSignAppleFrameworkForXcode

shared.framework

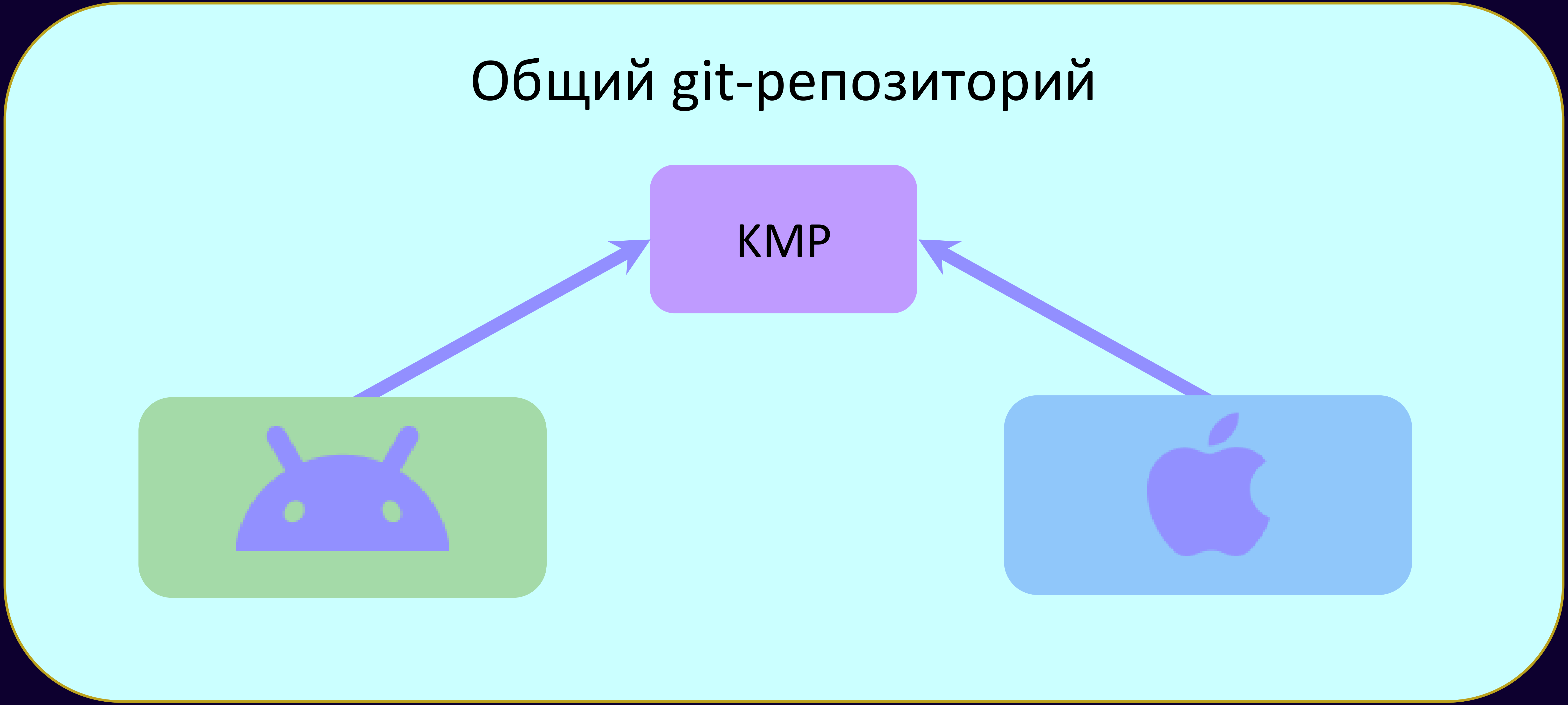
iOS и KMP в одном репозитории



Git-репозиторий

KMP

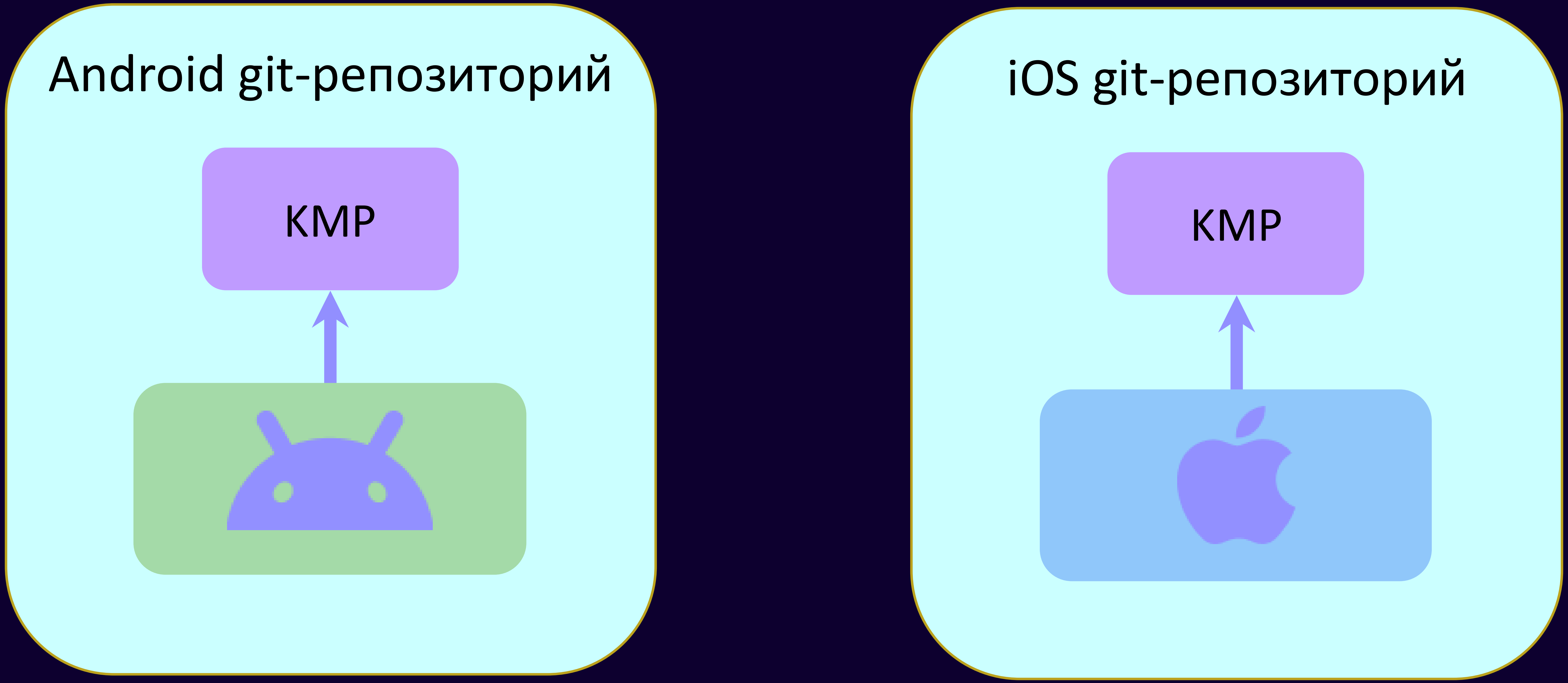
Общий KMP



Общий git-репозиторий

KMP

Индивидуальные KMP



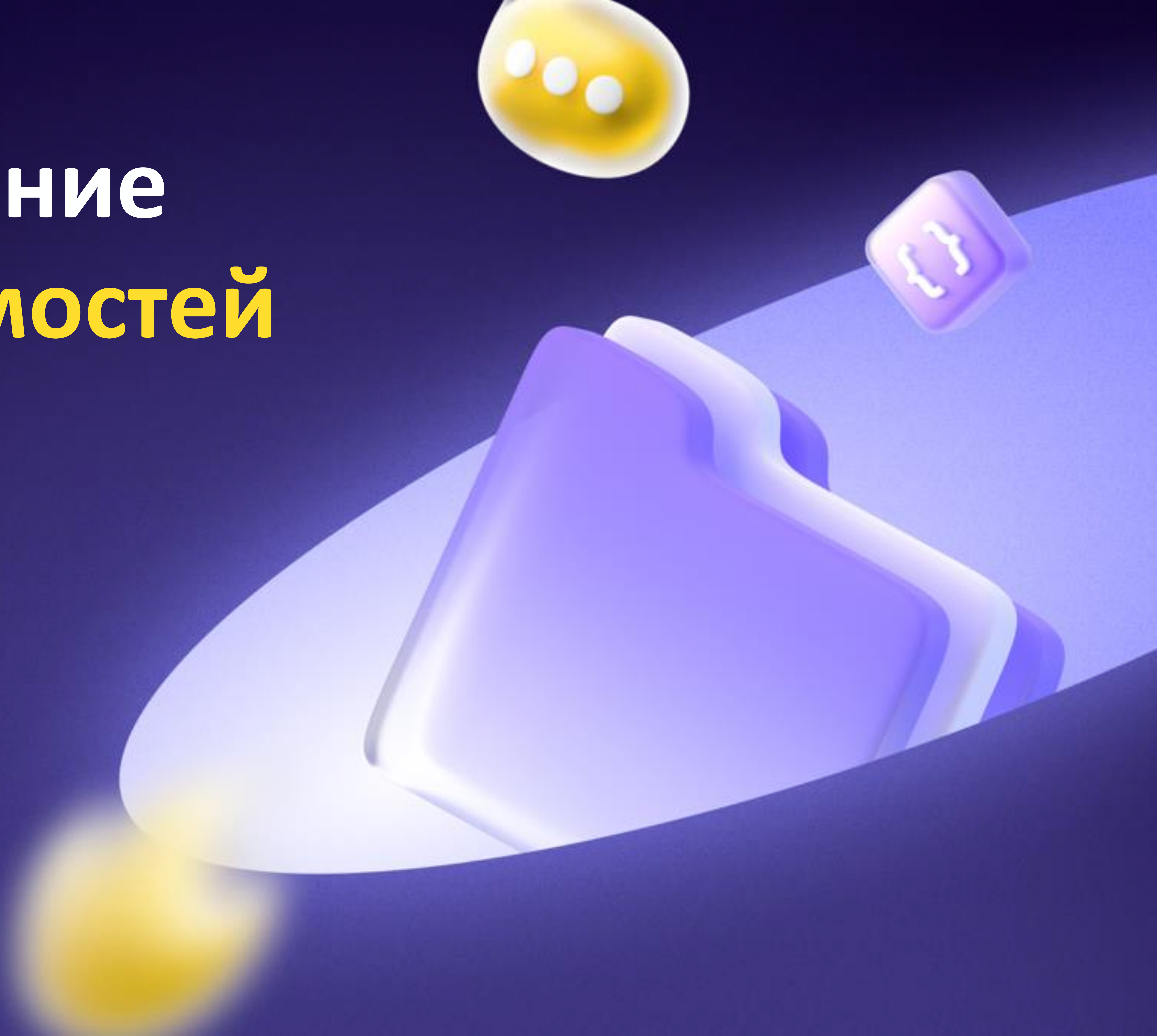
Android git-репозиторий

KMP

iOS git-репозиторий

KMP

Использование iOS-зависимостей



Еще про Direct Integration

- iOS-приложение будет знать о KMP
- увеличивает время сборки и размер DerivedData
- мгновенная доставка изменений

Why Kotlin Multiplatform Teams Should Share Source, not Binaries

https://touchlab.co/kmp-teams-use-source

Вердикт по Direct Integration — **не подходит**

Нельзя использовать iOS-зависимости

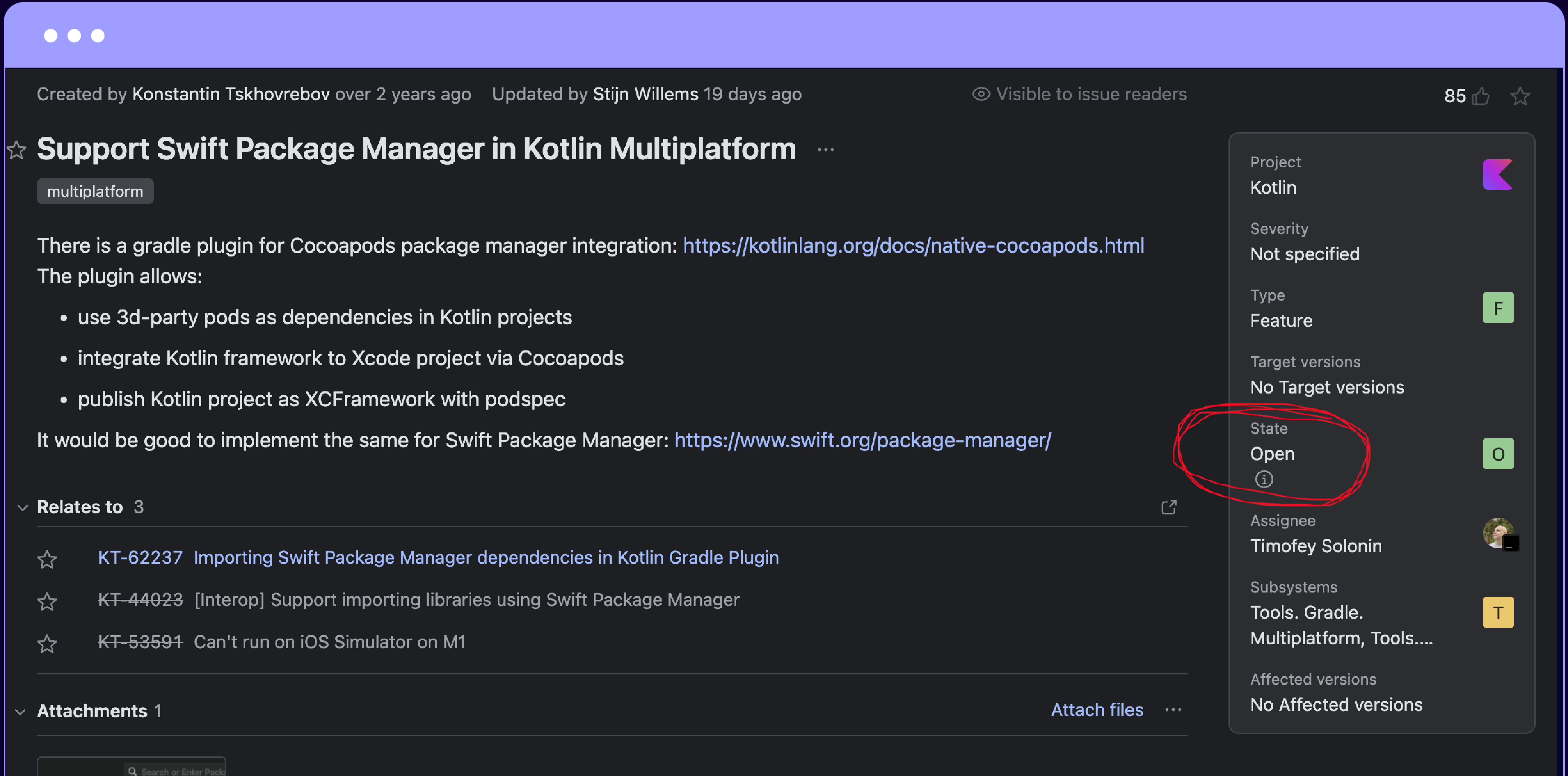
Увеличивается время сборки

SPM/Cocoapods Integration

Виды интеграций KMP в iOS

	No package manager	SPM	CocoaPods
Local	Direct Integration	SPM Local	CocoaPods Local
Remote	—	SPM Remote	CocoaPodsRemote

KT-53877 - SPM Support



Created by **Konstantin Tskhovrebov** over 2 years ago   Updated by **Stijn Willems** 19 days ago   Visible to issue readers   85

Support Swift Package Manager in Kotlin Multiplatform

multiplatform

There is a gradle plugin for Cocoapods package manager integration: https://kotlinlang.org/docs/native-cocoapods.html
The plugin allows:

- use 3d-party pods as dependencies in Kotlin projects
- integrate Kotlin framework to Xcode project via Cocoapods
- publish Kotlin project as XCFramework with podspec

It would be good to implement the same for Swift Package Manager: https://www.swift.org/package-manager/

Relates to 3

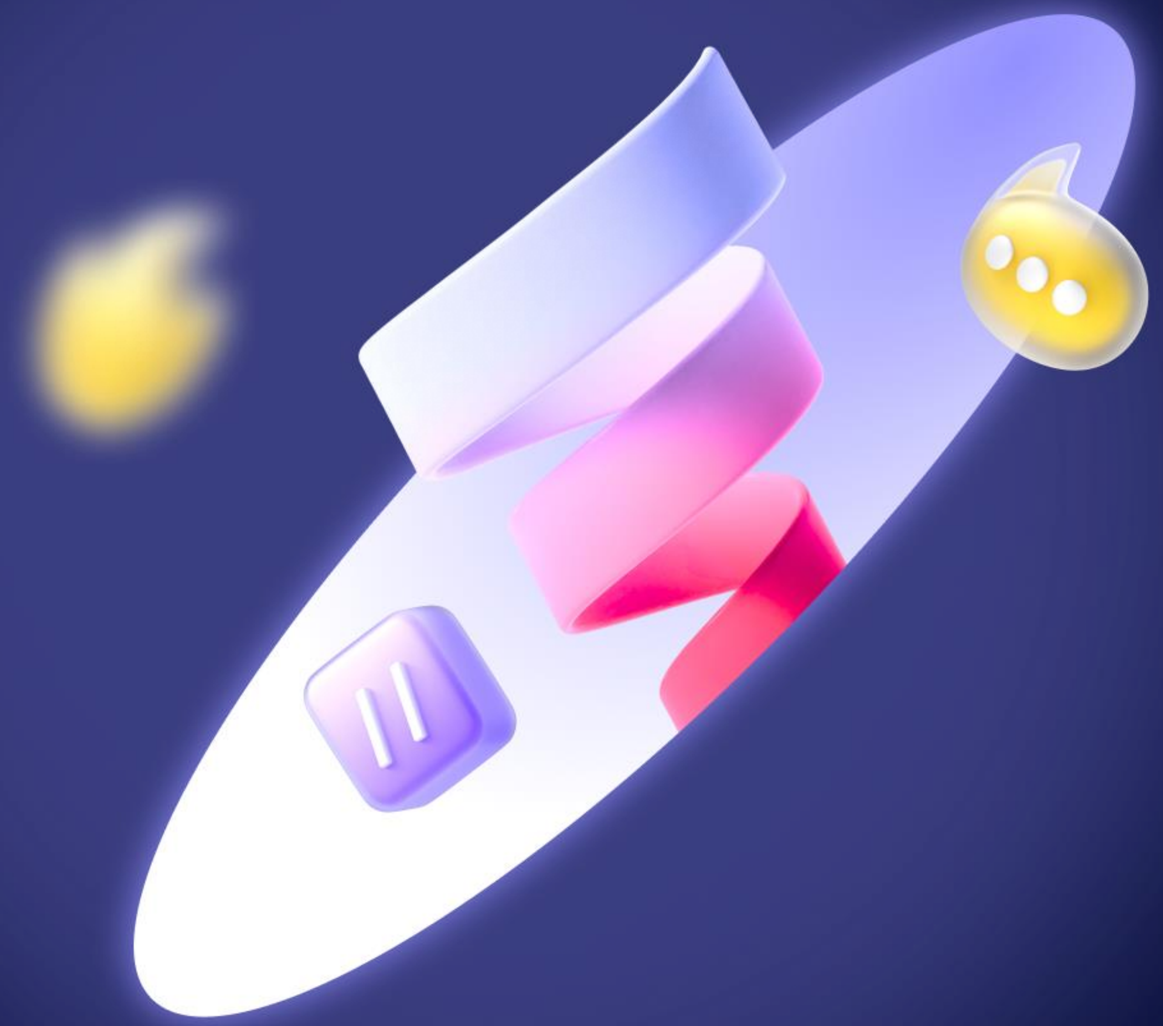
- KT-62237 Importing Swift Package Manager dependencies in Kotlin Gradle Plugin
- ~~KT-44023~~ [Interop] Support importing libraries using Swift Package Manager
- ~~KT-53591~~ Can't run on iOS Simulator on M1

Attachments 1   Attach files

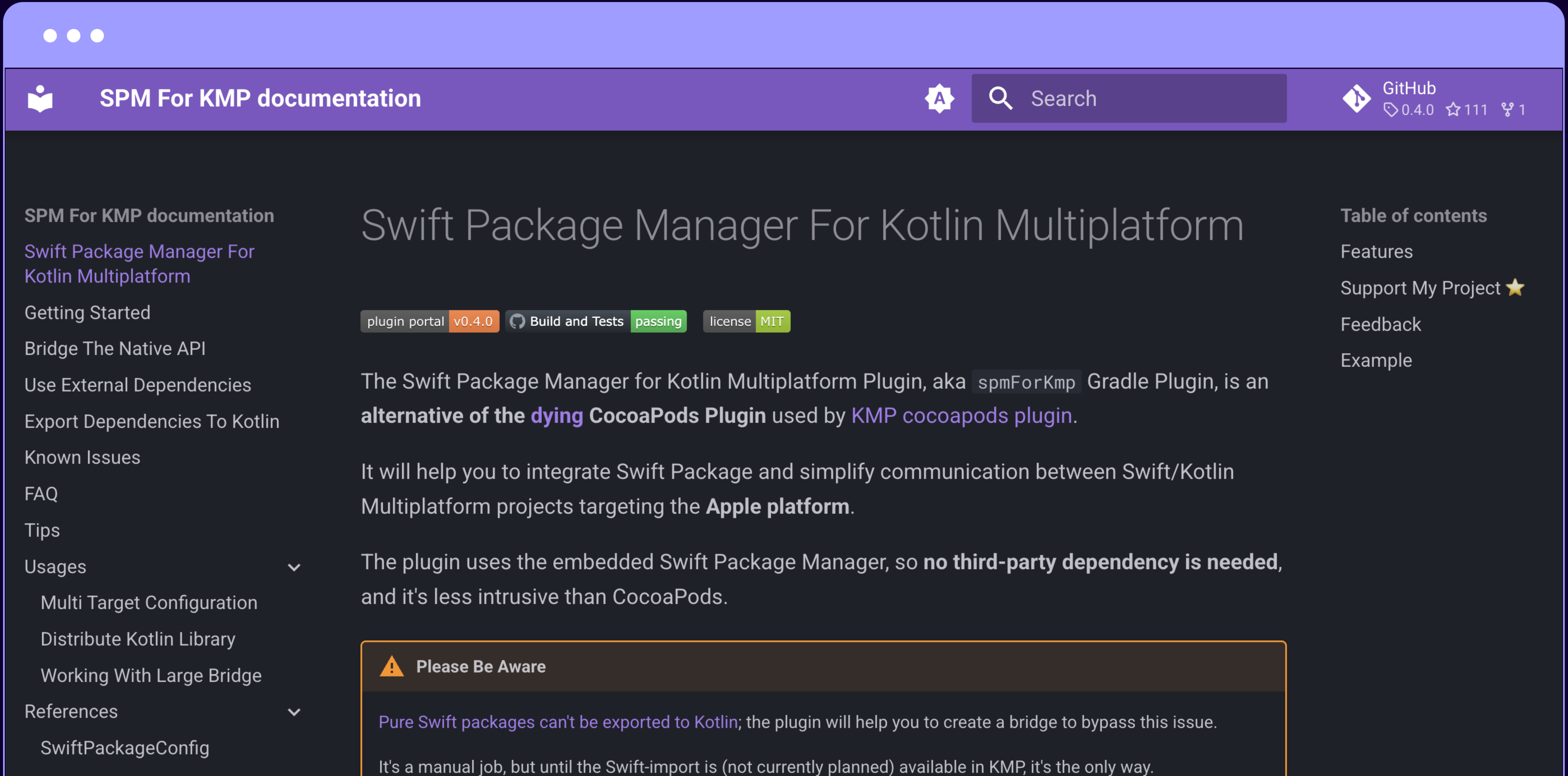
Project	Kotlin
Severity	Not specified
Type	Feature
Target versions	No Target versions
State	Open
Assignee	Timofey Solonin
Subsystems	Tools. Gradle. Multiplatform, Tools....
Affected versions	No Affected versions

Подключение iOS-библиотеки через SPM

1. Подтянуть библиотеку (фреймворк)
2. Построить и подключить биндинги — cinterop
3. Настроить линковку — build.gradle.kts
4. Сделать Package.swift
5. Собрать и задеплоить



SPM For KMP


SPM For KMP documentation
Search
GitHub
0.4.0
111
1
SPM For KMP documentation
Swift Package Manager For Kotlin Multiplatform
Getting Started
Bridge The Native API
Use External Dependencies
Export Dependencies To Kotlin
Known Issues
FAQ
Tips
Usages
Multi Target Configuration
Distribute Kotlin Library
Working With Large Bridge
References
SwiftPackageConfig
Table of contents
Features
Support My Project ⭐
Feedback
Example
plugin portal v0.4.0
Build and Tests passing
license MIT

Swift Package Manager For Kotlin Multiplatform

The Swift Package Manager for Kotlin Multiplatform Plugin, aka `spmForKmp` Gradle Plugin, is an **alternative of the dying CocoaPods Plugin** used by KMP cocoapods plugin.

It will help you to integrate Swift Package and simplify communication between Swift/Kotlin Multiplatform projects targeting the **Apple platform**.

The plugin uses the embedded Swift Package Manager, so **no third-party dependency is needed**, and it's less intrusive than CocoaPods.

Please Be Aware

Pure Swift packages can't be exported to Kotlin; the plugin will help you to create a bridge to bypass this issue.

It's a manual job, but until the Swift-import is (not currently planned) available in KMP, it's the only way.

Интеграция через пакетные менеджеры

Cocoapods Local/Remote

- Будет заморожен в декабре 2026 года.

SPM Local

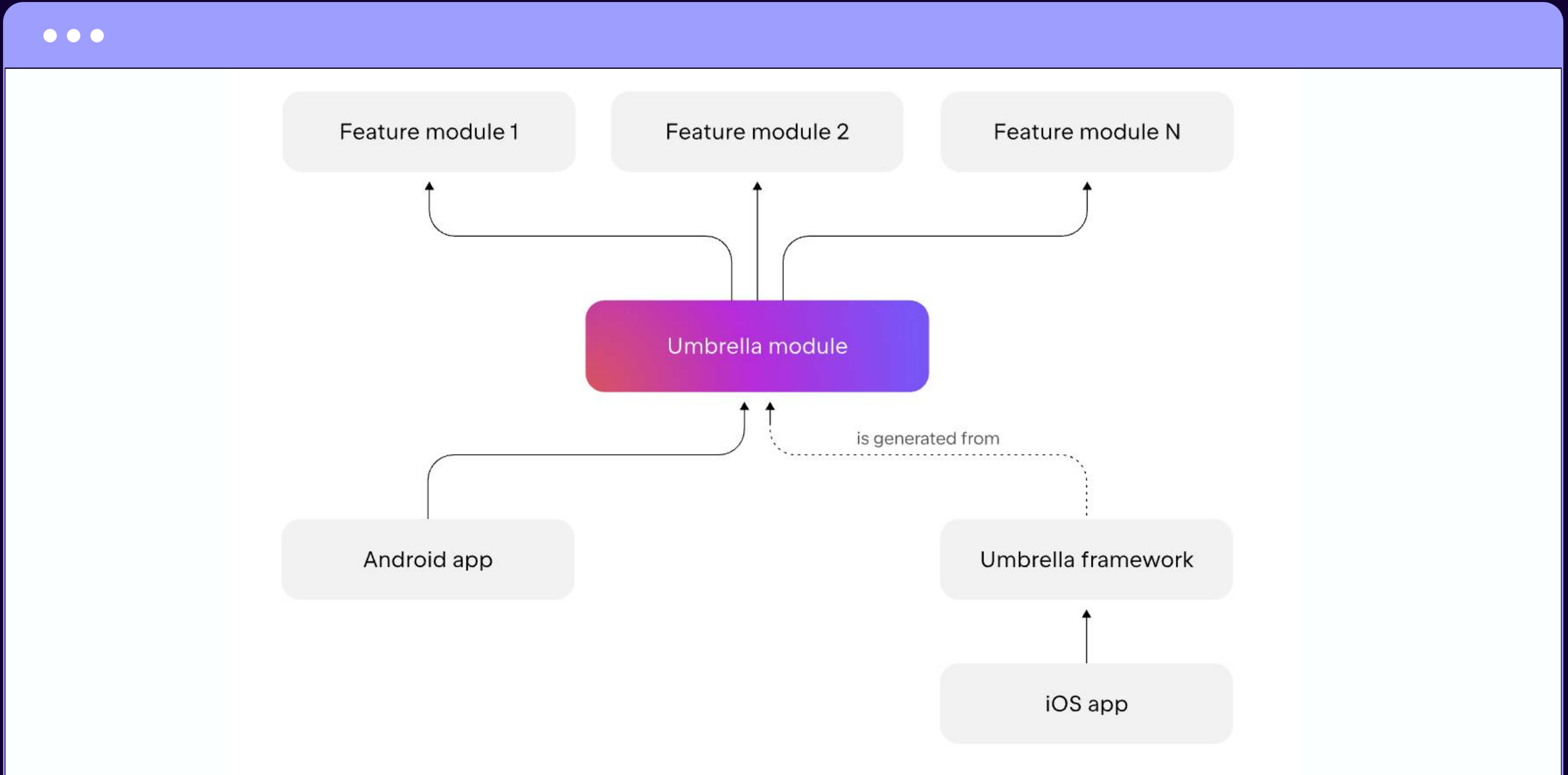
- Увеличивает время сборки и размер DerivedData.

SPM Remote

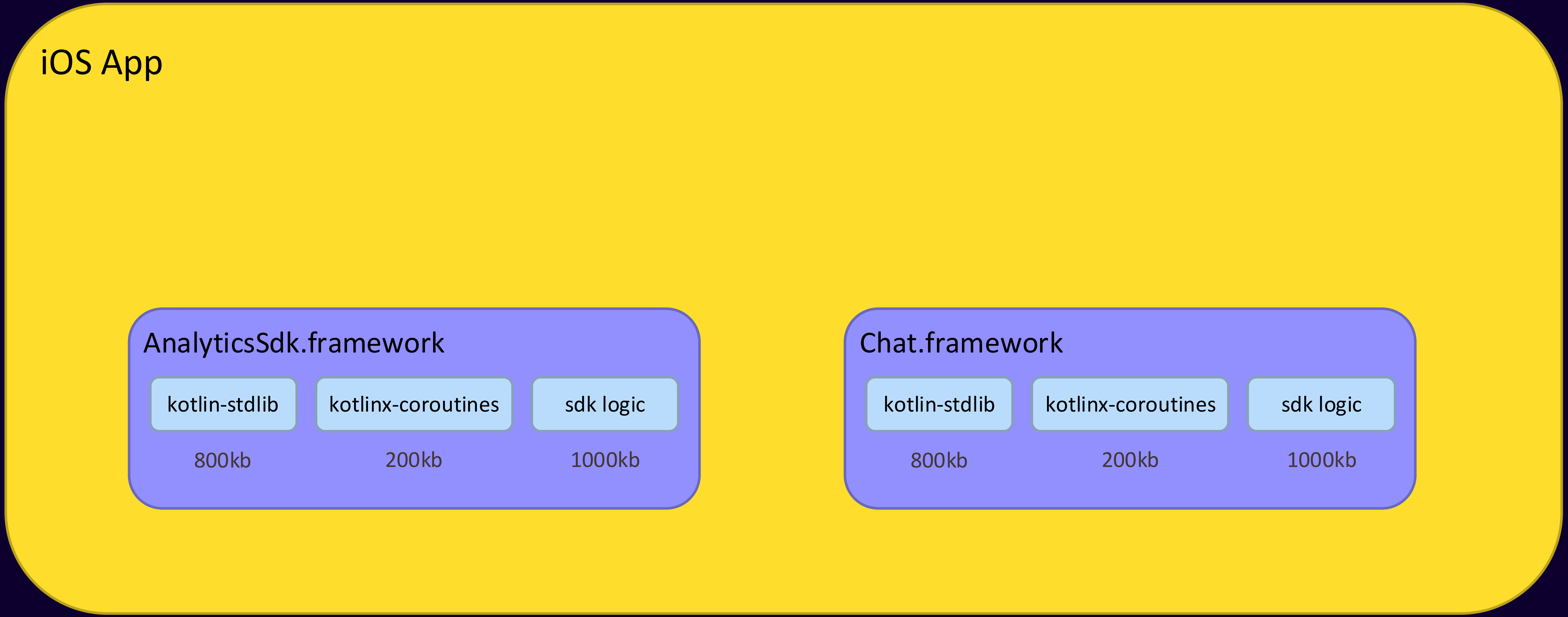
- Позволяет использовать iOS-зависимости.
- Не влияет на время сборки, т.к. не встраивается в пайплайн билда.

Проблема 2: Что делать с ограничением umbrella-модуля?

Umbrella-модуль



Дублирование зависимостей



iOS App

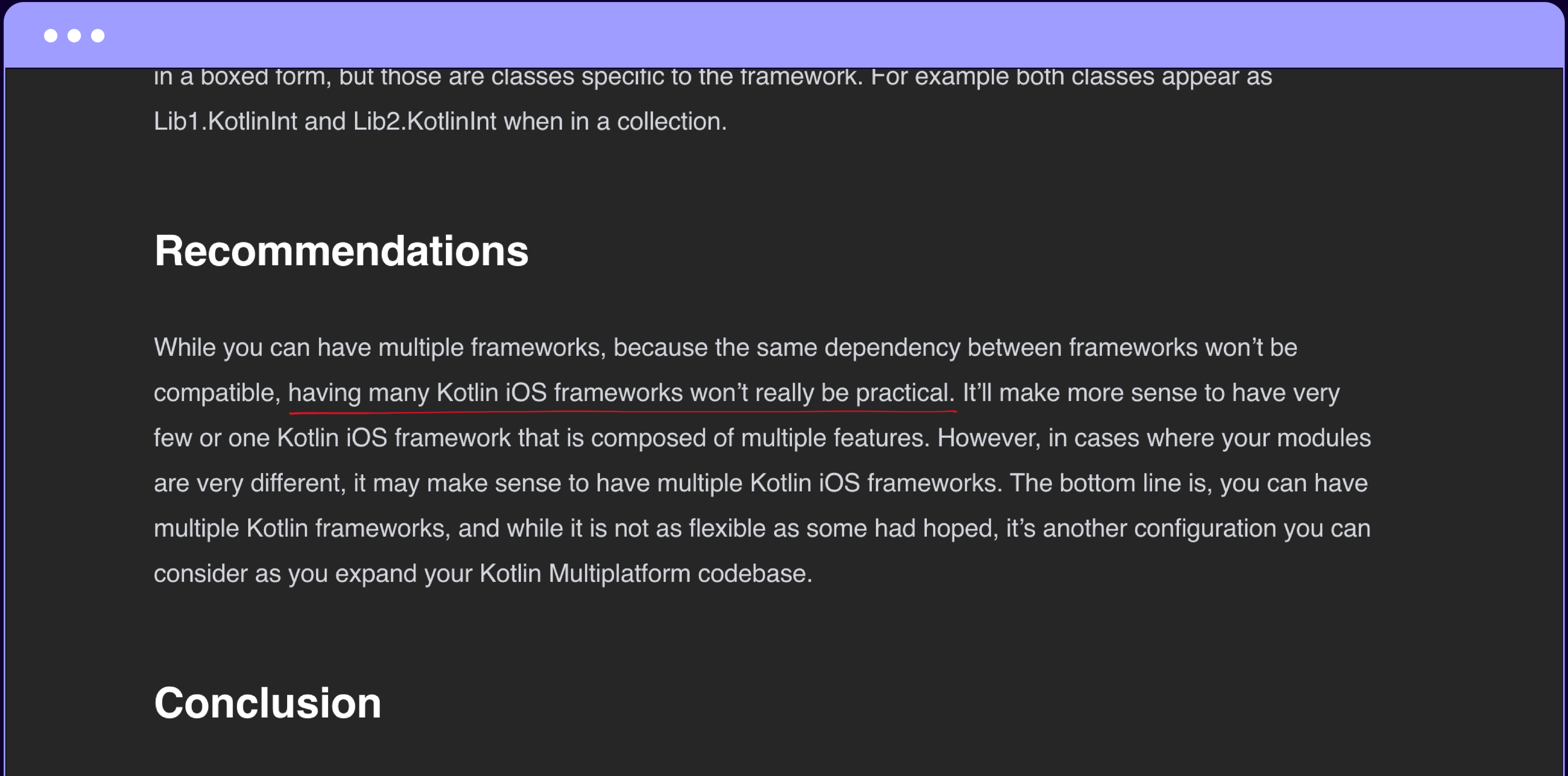
AnalyticsSdk.framework

kotlin-stdlib	kotlinx-coroutines	sdk logic
800kb	200kb	1000kb

Chat.framework

kotlin-stdlib	kotlinx-coroutines	sdk logic
800kb	200kb	1000kb

Мнение Touchlab



in a boxed form, but those are classes specific to the framework. For example both classes appear as Lib1.KotlinInt and Lib2.KotlinInt when in a collection.

Recommendations

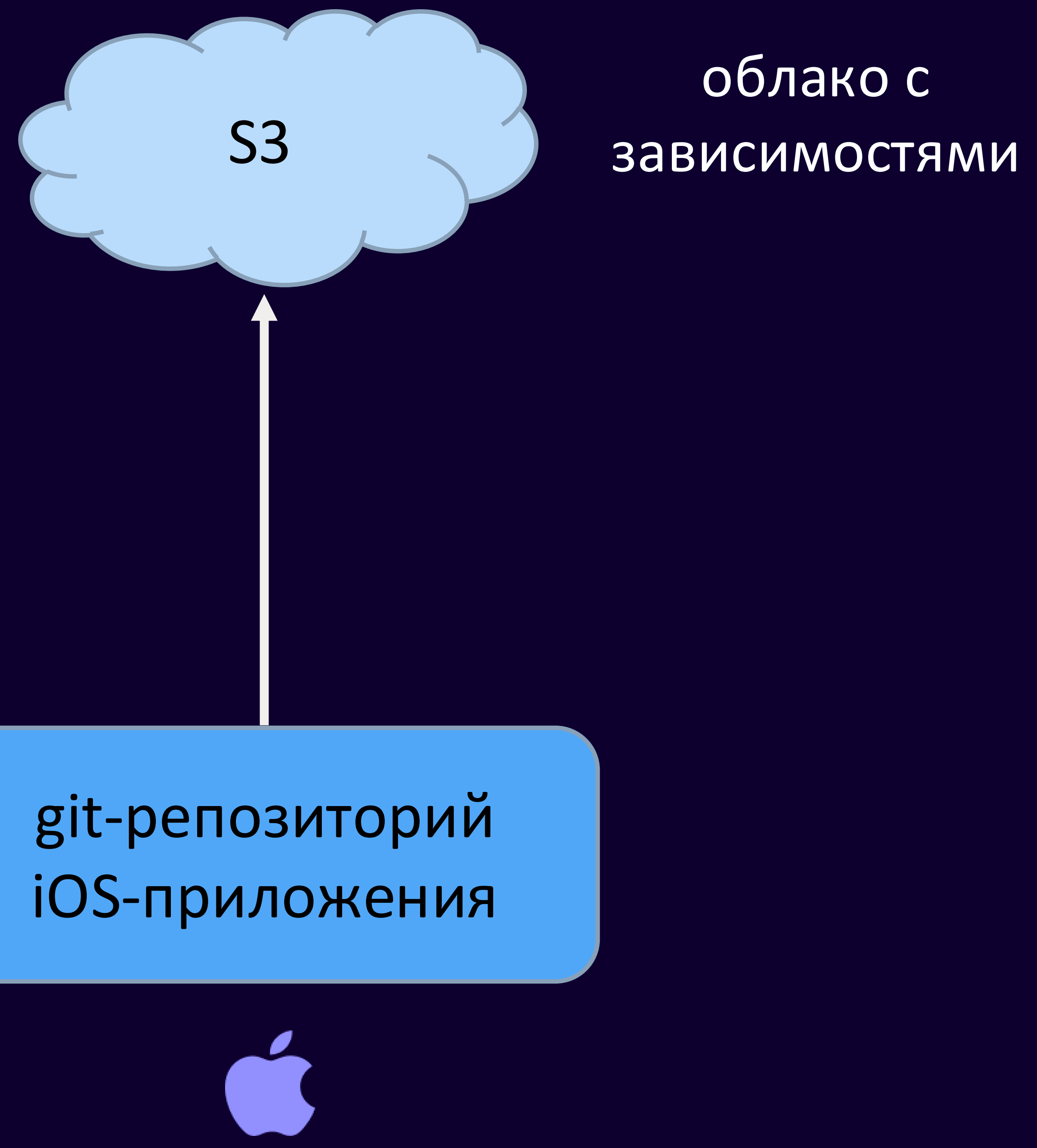
While you can have multiple frameworks, because the same dependency between frameworks won't be compatible, having many Kotlin iOS frameworks won't really be practical. It'll make more sense to have very few or one Kotlin iOS framework that is composed of multiple features. However, in cases where your modules are very different, it may make sense to have multiple Kotlin iOS frameworks. The bottom line is, you can have multiple Kotlin frameworks, and while it is not as flexible as some had hoped, it's another configuration you can consider as you expand your Kotlin Multiplatform codebase.

Conclusion

Как организовать подключение KMP к iOS **одним фреймворком**?

Делать **КМР-Платформу**

KMP-Платформа — Схема

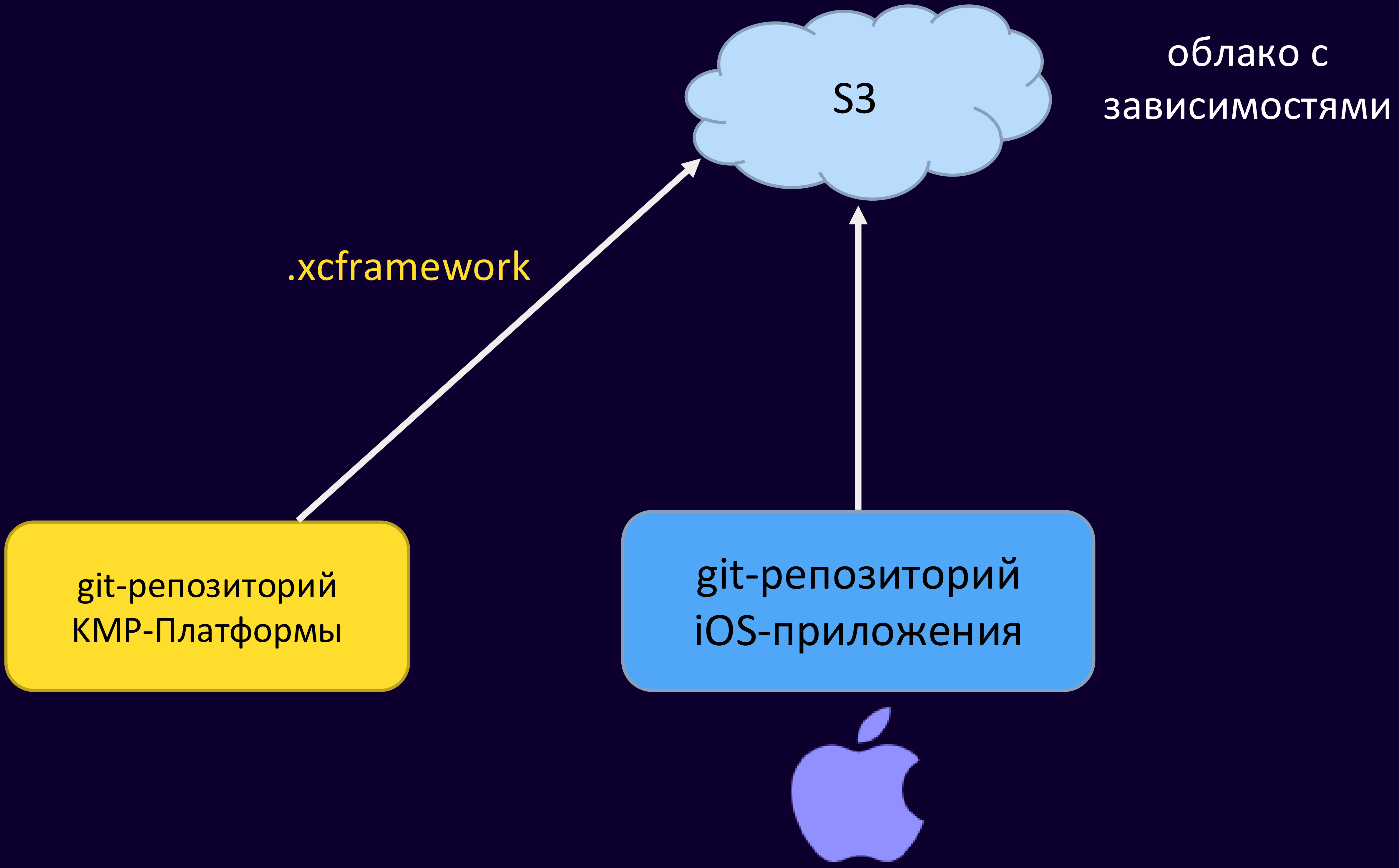


S3

облако с зависимостями

git-репозиторий iOS-приложения

KMP-Платформа — Схема



S3

облако с
зависимостями

.xcframework

git-репозиторий
KMP-Платформы

git-репозиторий
iOS-приложения

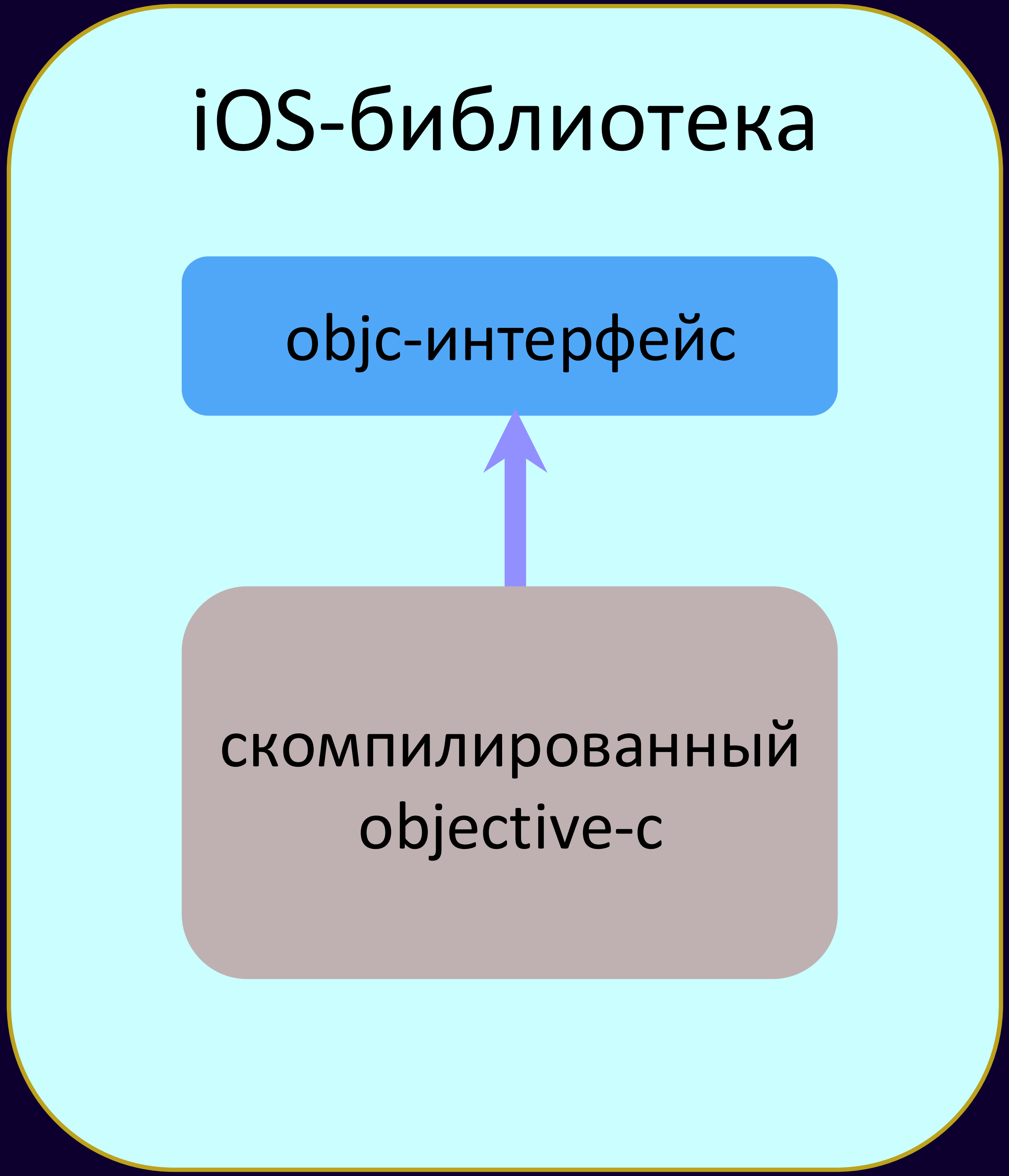
Решение проблемы Umbrella-модуля

- ✓ Remote SPM
- ✓ KMP-Платформа

Проблема 3: Что делать с неудобным API?

Добавить Swift-обертку?

Устройство ObjC-фреймворка

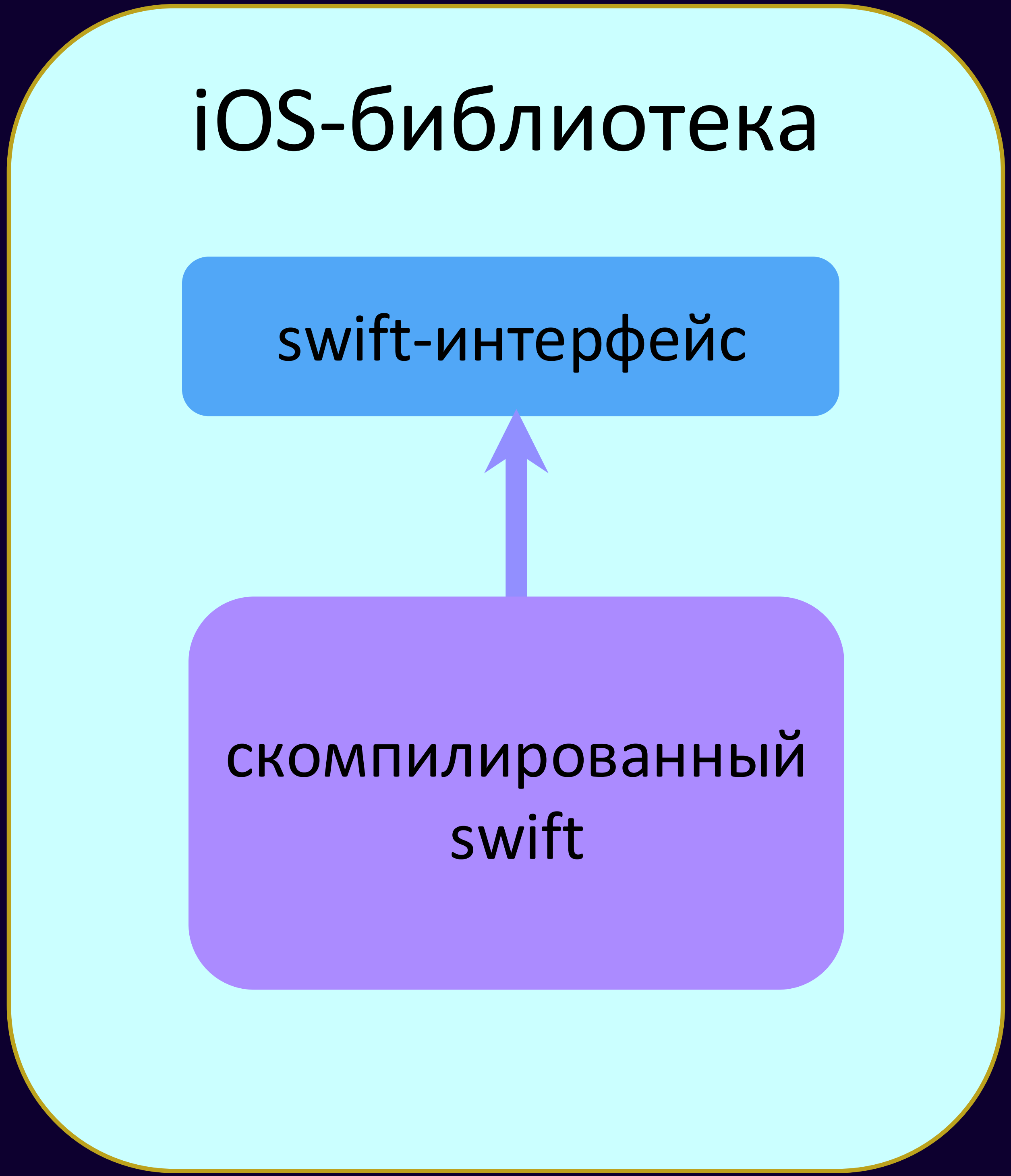


iOS-библиотека

objc-интерфейс

скомпилированный objective-c

Устройство Swift-фреймворка

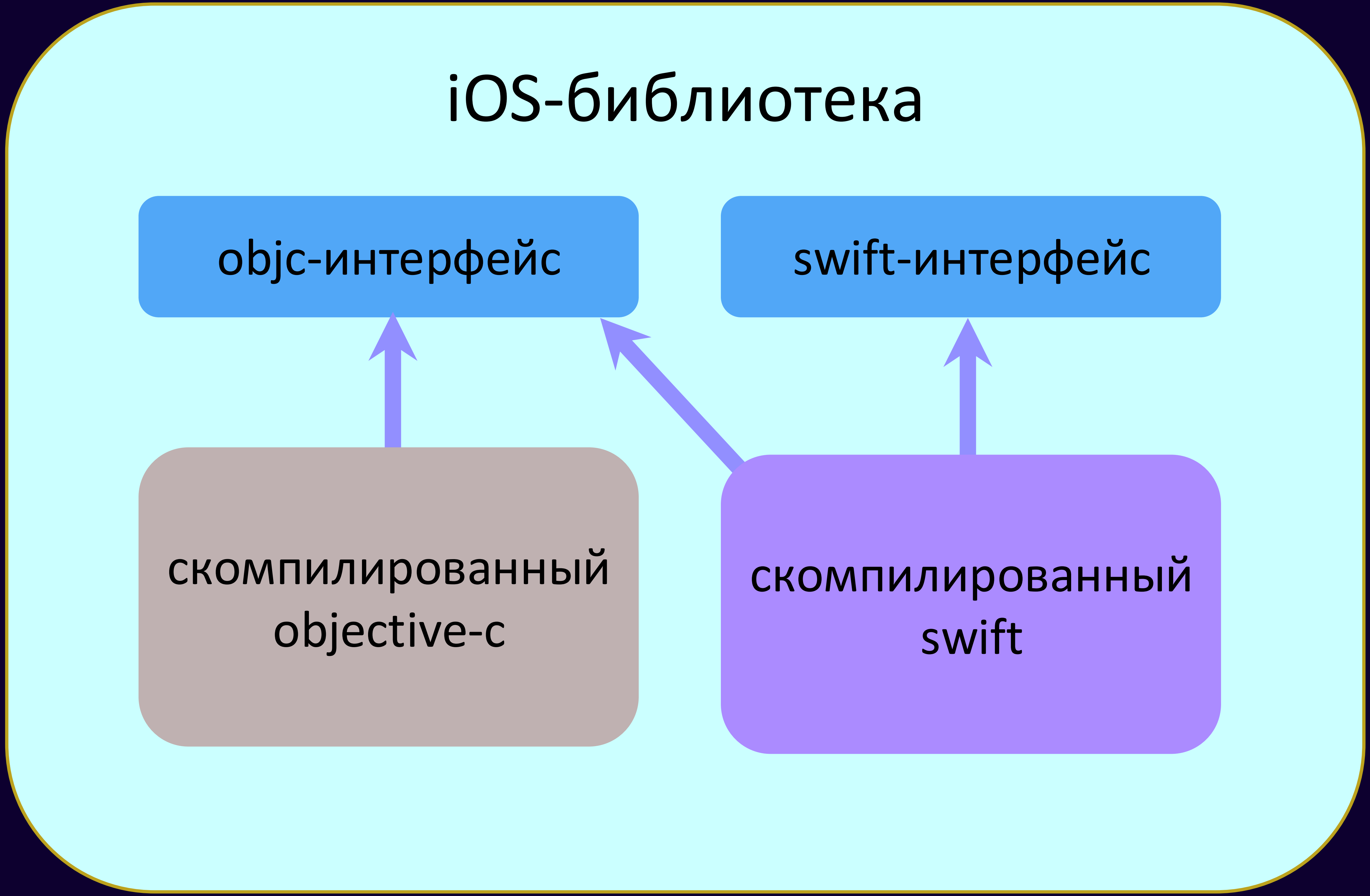


iOS-библиотека

swift-интерфейс

скомпилированный swift

Устройство ObjC+Swift фреймворка



iOS-библиотека

objc-интерфейс

swift-интерфейс

скомпилированный objective-c

скомпилированный swift

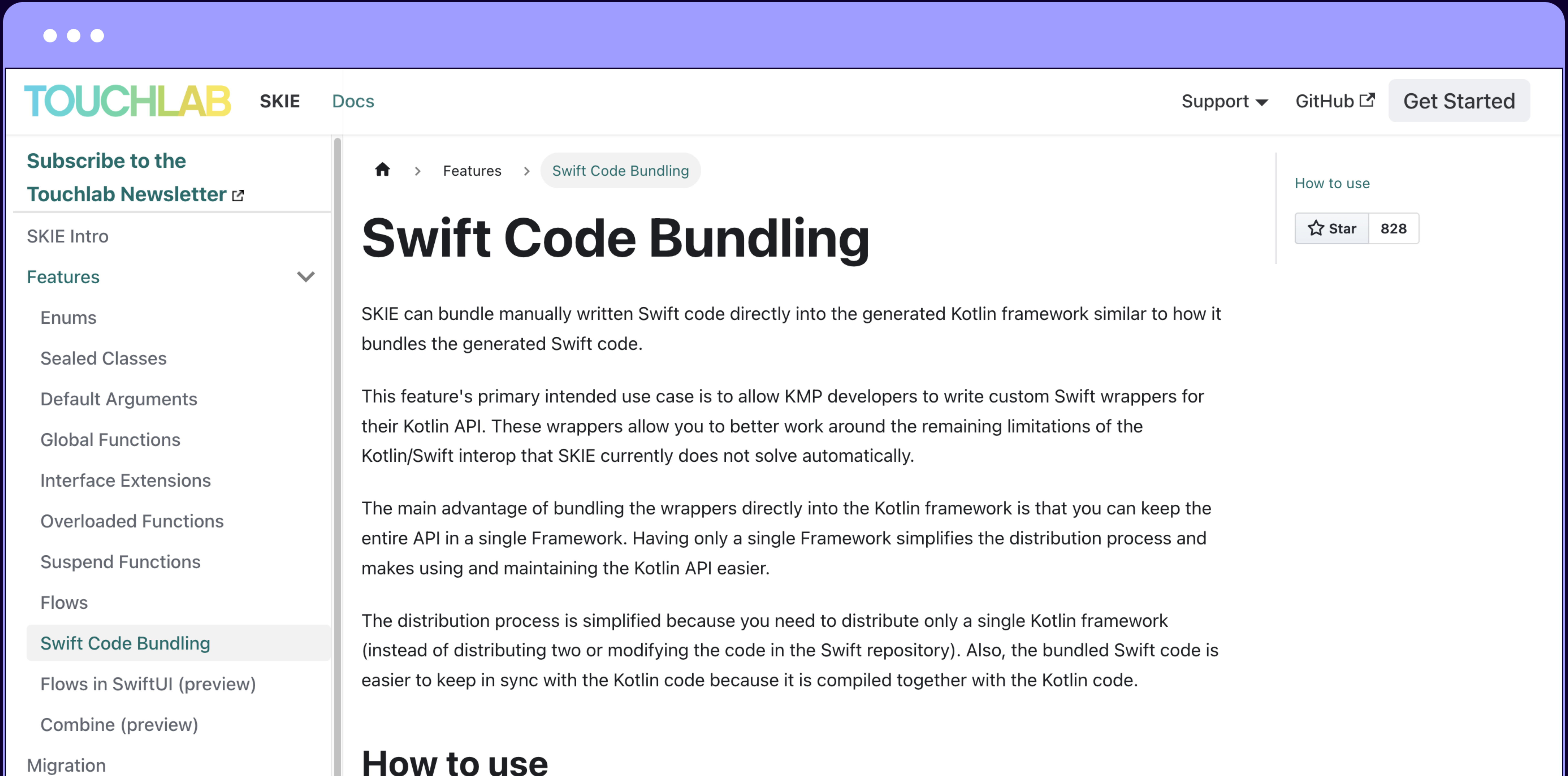
Алгоритм бандлинга Swift

1. собрать KMP-framework с objc-модулем
2. сгенерировать для него Bridging Headers
   `shared-Swift.h`
3. пересобрать модуль
   `swiftc -import-underlying-module -emit-module …`

Изменение после бандлинга

```
shared.framework/
    Headers/
        shared.h
    Modules/
        module.modulemap
    Info.plist
    shared
```

```
shared.framework/
    Headers/
        shared.h
        shared.apinotes
        shared-Swift.h
    Modules/
        shared.swiftmodule/
            arm64-apple-ios-simulator.private.swiftinterface
            arm64-apple-ios-simulator.swiftdoc
            arm64-apple-ios-simulator.swiftinterface
            arm64-apple-ios-simulator.swiftmodule
            arm64-apple-ios-simulator.swiftsourceinfo
        module.modulemap
    Info.plist
    shared
```



Swift Code Bundling

SKIE can bundle manually written Swift code directly into the generated Kotlin framework similar to how it bundles the generated Swift code.

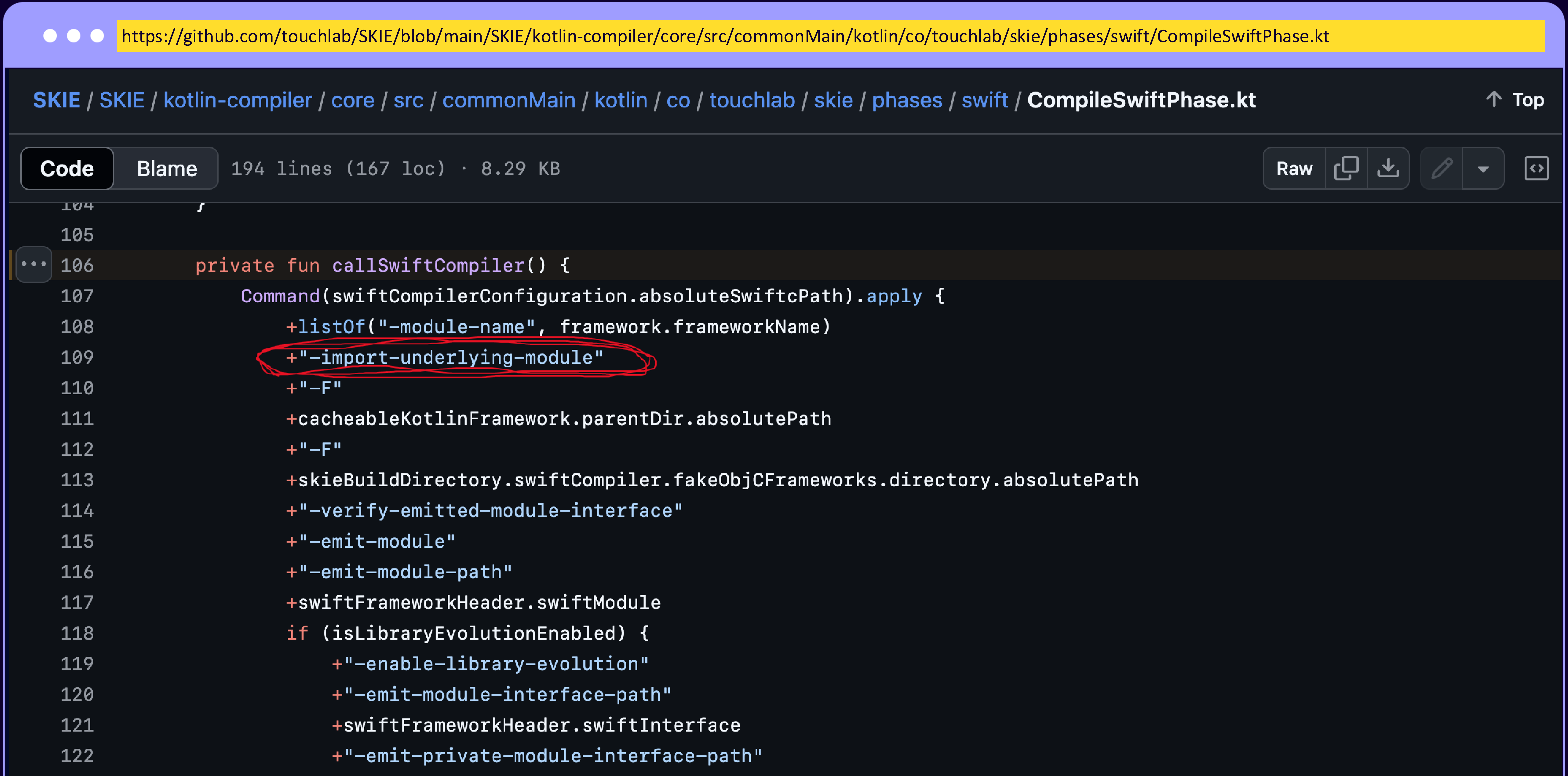
This feature's primary intended use case is to allow KMP developers to write custom Swift wrappers for their Kotlin API. These wrappers allow you to better work around the remaining limitations of the Kotlin/Swift interop that SKIE currently does not solve automatically.

The main advantage of bundling the wrappers directly into the Kotlin framework is that you can keep the entire API in a single Framework. Having only a single Framework simplifies the distribution process and makes using and maintaining the Kotlin API easier.

The distribution process is simplified because you need to distribute only a single Kotlin framework (instead of distributing two or modifying the code in the Swift repository). Also, the bundled Swift code is easier to keep in sync with the Kotlin code because it is compiled together with the Kotlin code.

How to use

Исходники Swift-бандлера



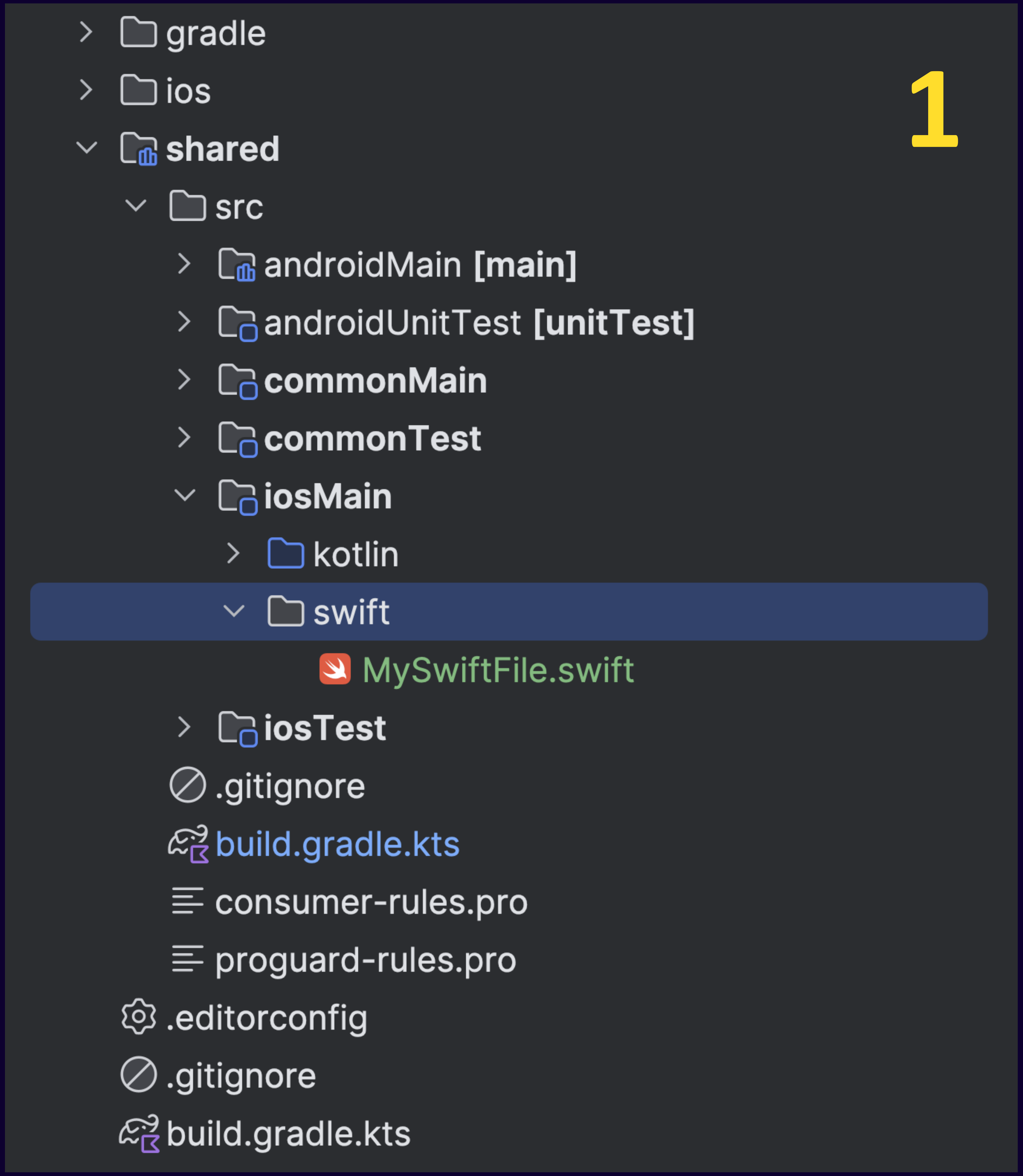
https://github.com/touchlab/SKIE/blob/main/SKIE/kotlin-compiler/core/src/commonMain/kotlin/co/touchlab/skie/phases/swift/CompileSwiftPhase.kt

SKIE / SKIE / kotlin-compiler / core / src / commonMain / kotlin / co / touchlab / skie / phases / swift / **CompileSwiftPhase.kt**

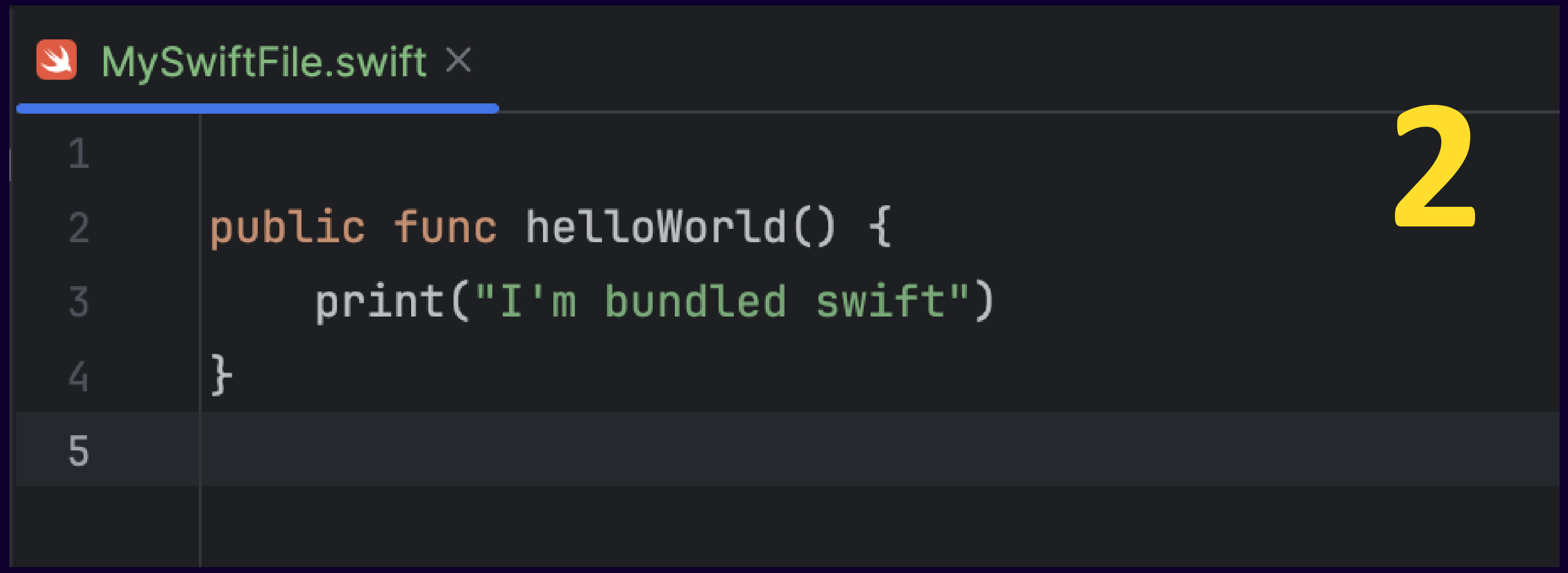
194 lines (167 loc) · 8.29 KB

```
    }

    private fun callSwiftCompiler() {
        Command(swiftCompilerConfiguration.absoluteSwiftcPath).apply {
            +listOf("-module-name", framework.frameworkName)
            +"-import-underlying-module"
            +"-F"
            +cacheableKotlinFramework.parentDir.absolutePath
            +"-F"
            +skieBuildDirectory.swiftCompiler.fakeObjCFrameworks.directory.absolutePath
            +"-verify-emitted-module-interface"
            +"-emit-module"
            +"-emit-module-path"
            +swiftFrameworkHeader.swiftModule
            if (isLibraryEvolutionEnabled) {
                +"-enable-library-evolution"
                +"-emit-module-interface-path"
                +swiftFrameworkHeader.swiftInterface
                +"-emit-private-module-interface-path"
```

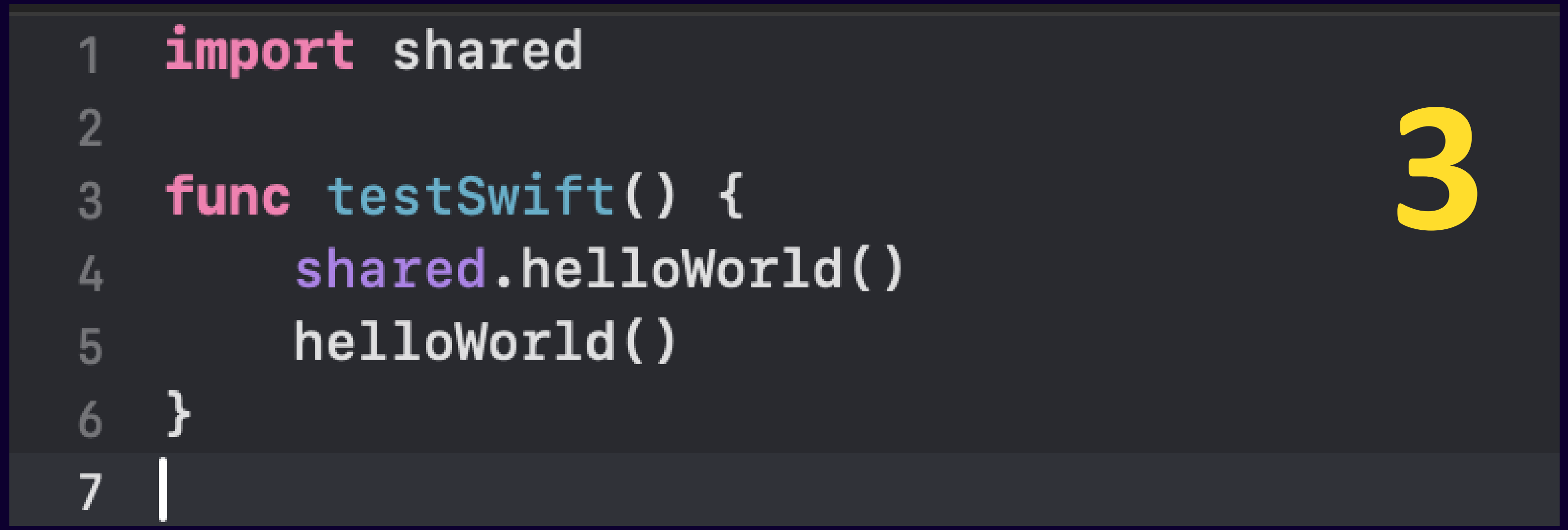
Пример бандлинга в коде



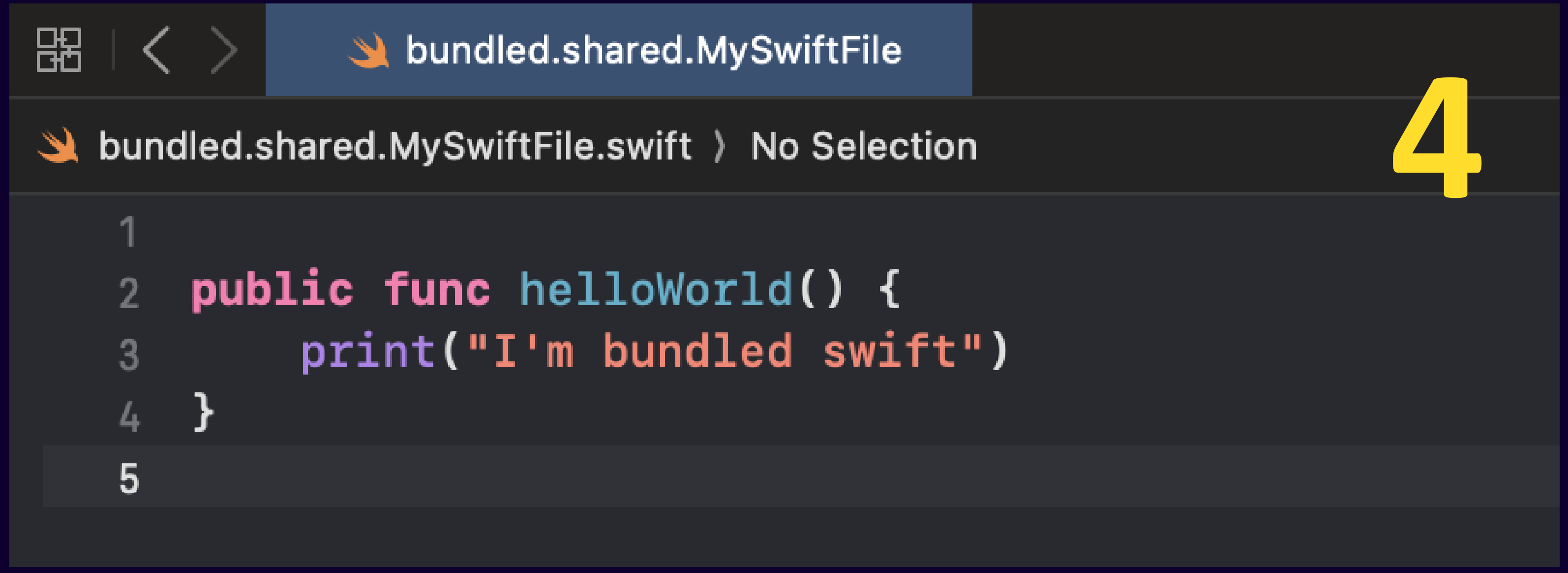
- gradle
- ios
- shared
  - src
    - androidMain [main]
    - androidUnitTest [unitTest]
    - commonMain
    - commonTest
    - iosMain
      - kotlin
      - swift
        - MySwiftFile.swift
    - iosTest
  - .gitignore
  - build.gradle.kts
  - consumer-rules.pro
  - proguard-rules.pro
- .editorconfig
- .gitignore
- build.gradle.kts



```swift

public func helloWorld() {
    print("I'm bundled swift")
}

```



```swift
import shared

func testSwift() {
    shared.helloWorld()
    helloWorld()
}

```

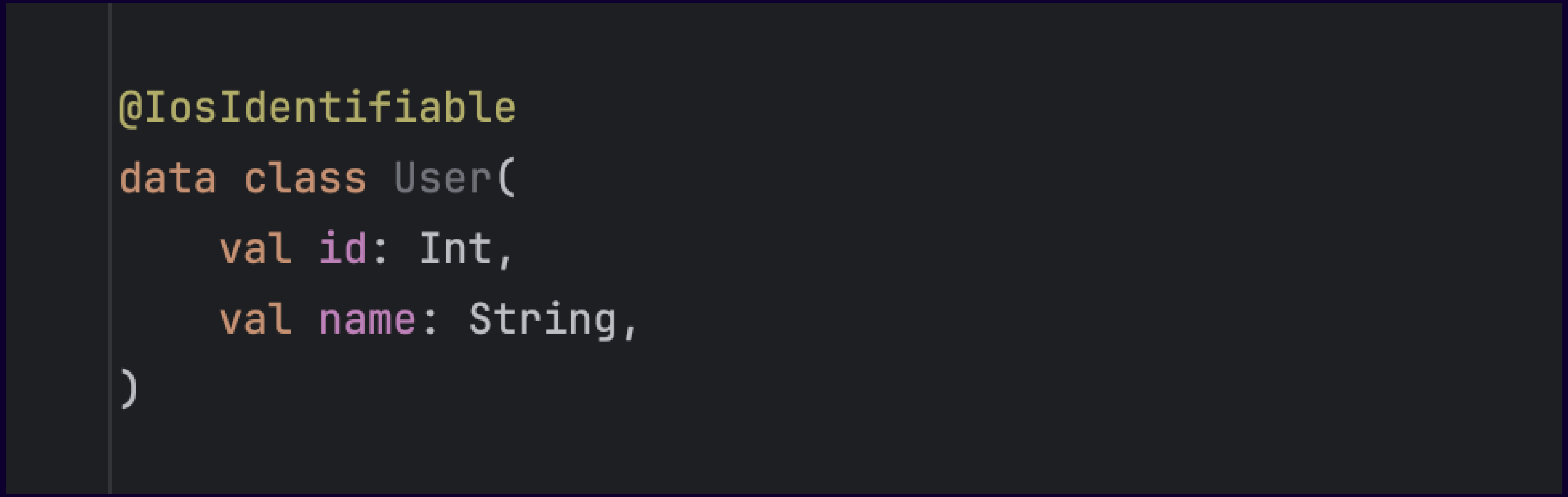

bundled.shared.MySwiftFile
bundled.shared.MySwiftFile.swift 〉 No Selection
4

```swift

public func helloWorld() {
    print("I'm bundled swift")
}

```

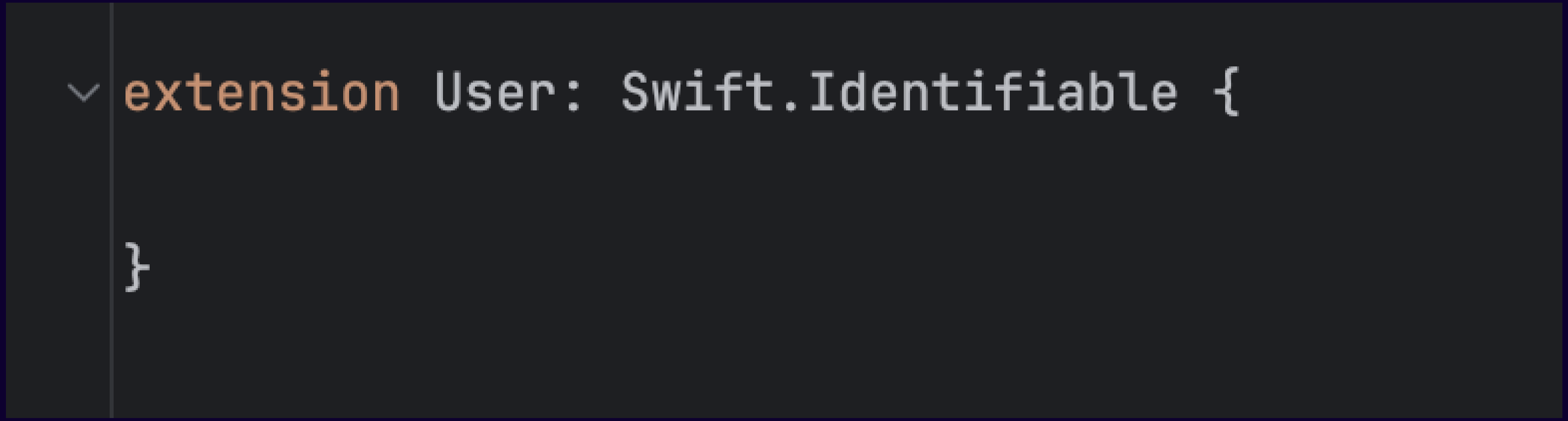
Еще про **бандлинг**

- можно доставлять документацию
- есть compile-проверка

Добавление Swift-extension из commonMain



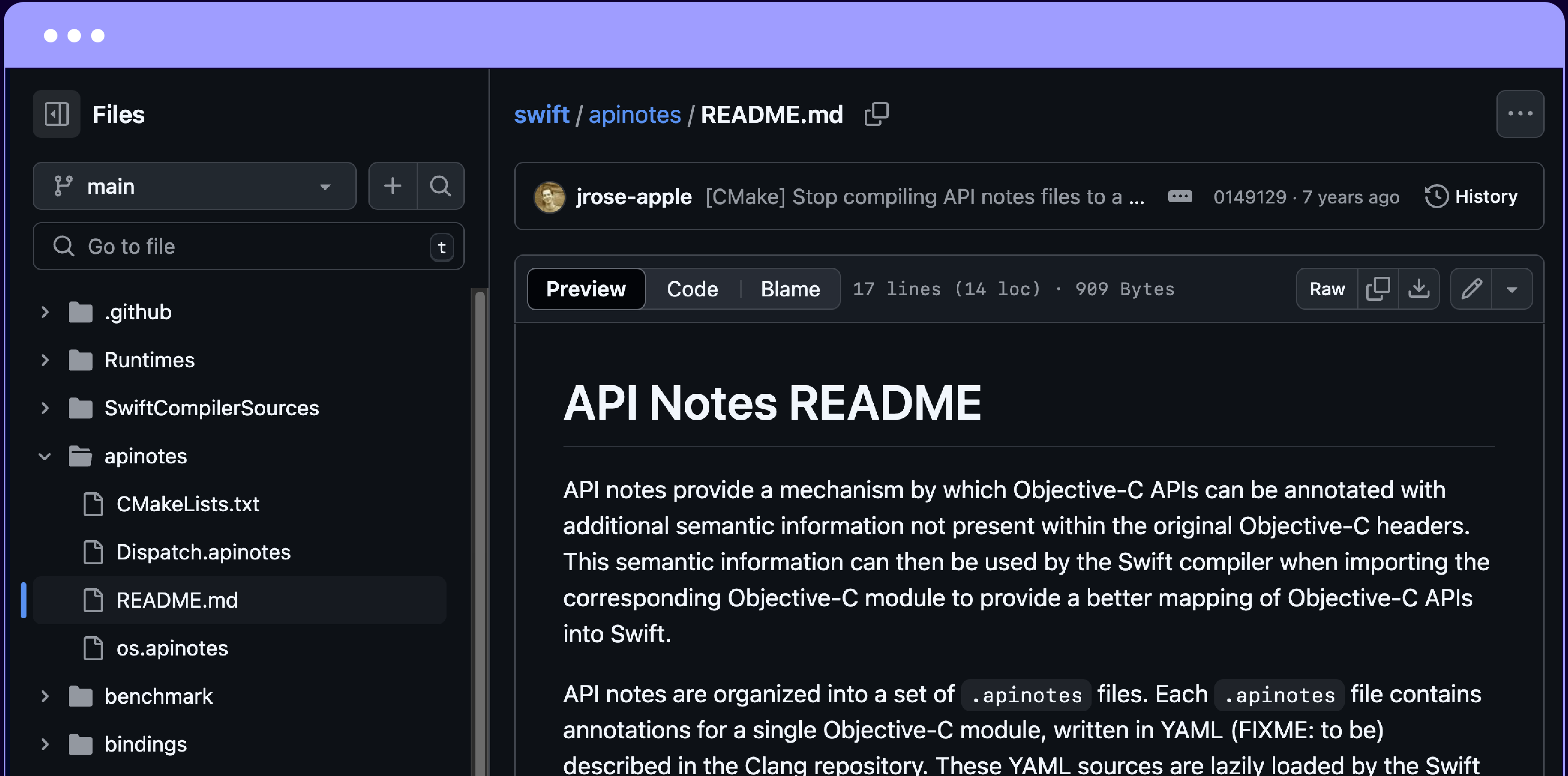
```
@IosIdentifiable
data class User(
    val id: Int,
    val name: String,
)
```



```
extension User: Swift.Identifiable {

}
```

SKIE

API Notes

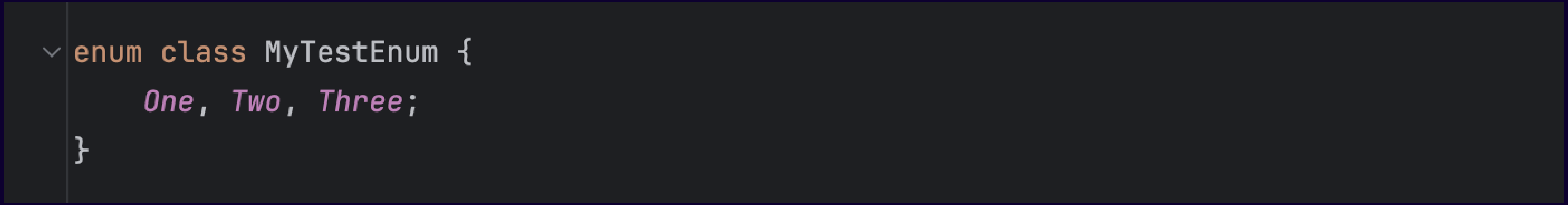


API Notes README

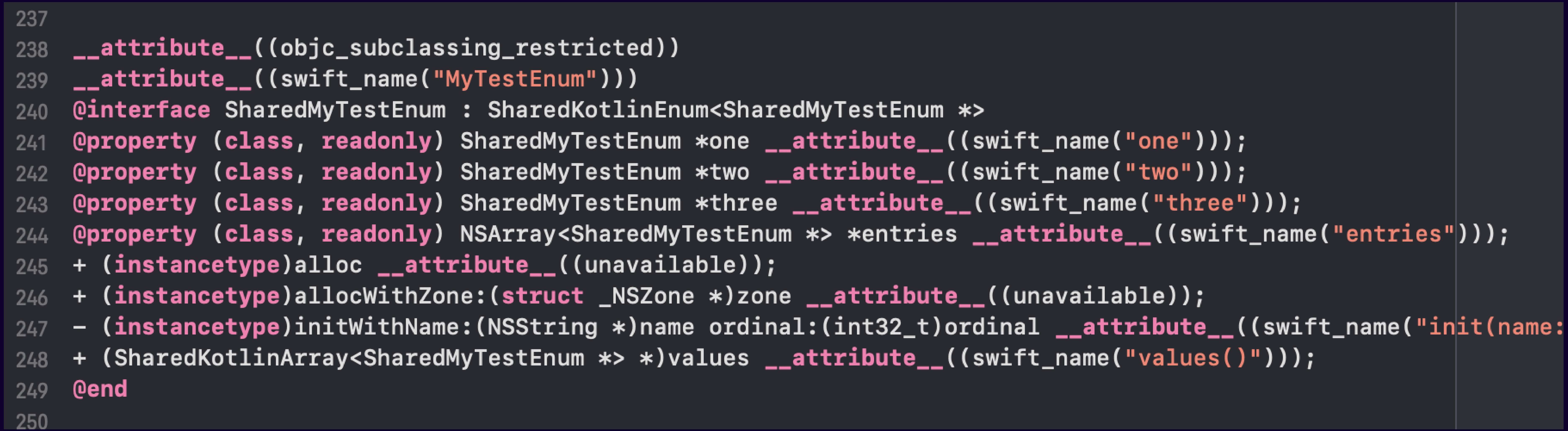
API notes provide a mechanism by which Objective-C APIs can be annotated with additional semantic information not present within the original Objective-C headers. This semantic information can then be used by the Swift compiler when importing the corresponding Objective-C module to provide a better mapping of Objective-C APIs into Swift.

API notes are organized into a set of `.apinotes` files. Each `.apinotes` file contains annotations for a single Objective-C module, written in YAML (FIXME: to be) described in the Clang repository. These YAML sources are lazily loaded by the Swift

Enum в KMP из коробки

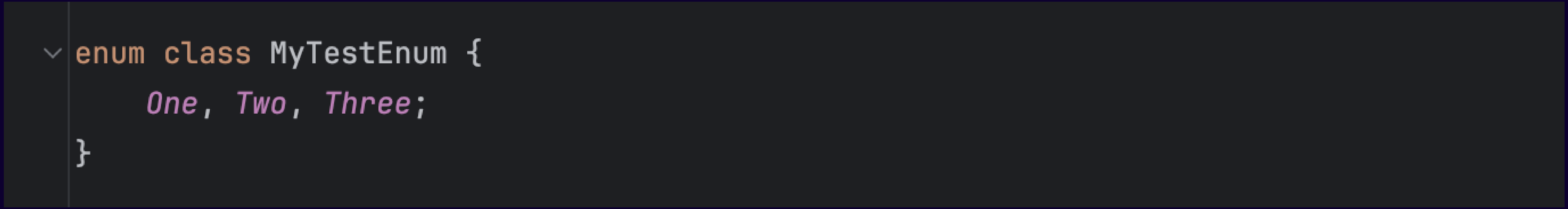


```kotlin
enum class MyTestEnum {
    One, Two, Three;
}
```

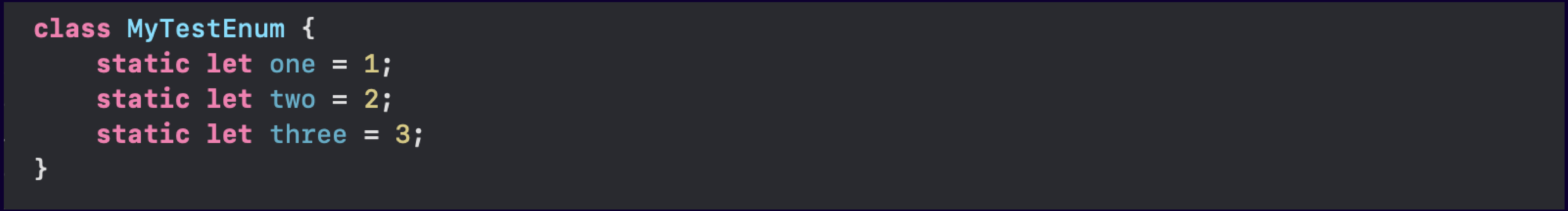


```objc

__attribute__((objc_subclassing_restricted))
__attribute__((swift_name("MyTestEnum")))
@interface SharedMyTestEnum : SharedKotlinEnum<SharedMyTestEnum *>
@property (class, readonly) SharedMyTestEnum *one __attribute__((swift_name("one")));
@property (class, readonly) SharedMyTestEnum *two __attribute__((swift_name("two")));
@property (class, readonly) SharedMyTestEnum *three __attribute__((swift_name("three")));
@property (class, readonly) NSArray<SharedMyTestEnum *> *entries __attribute__((swift_name("entries")));
+ (instancetype)alloc __attribute__((unavailable));
+ (instancetype)allocWithZone:(struct _NSZone *)zone __attribute__((unavailable));
- (instancetype)initWithName:(NSString *)name ordinal:(int32_t)ordinal __attribute__((swift_name("init(name:
+ (SharedKotlinArray<SharedMyTestEnum *> *)values __attribute__((swift_name("values()")));
@end

```

Enum в KMP из коробки

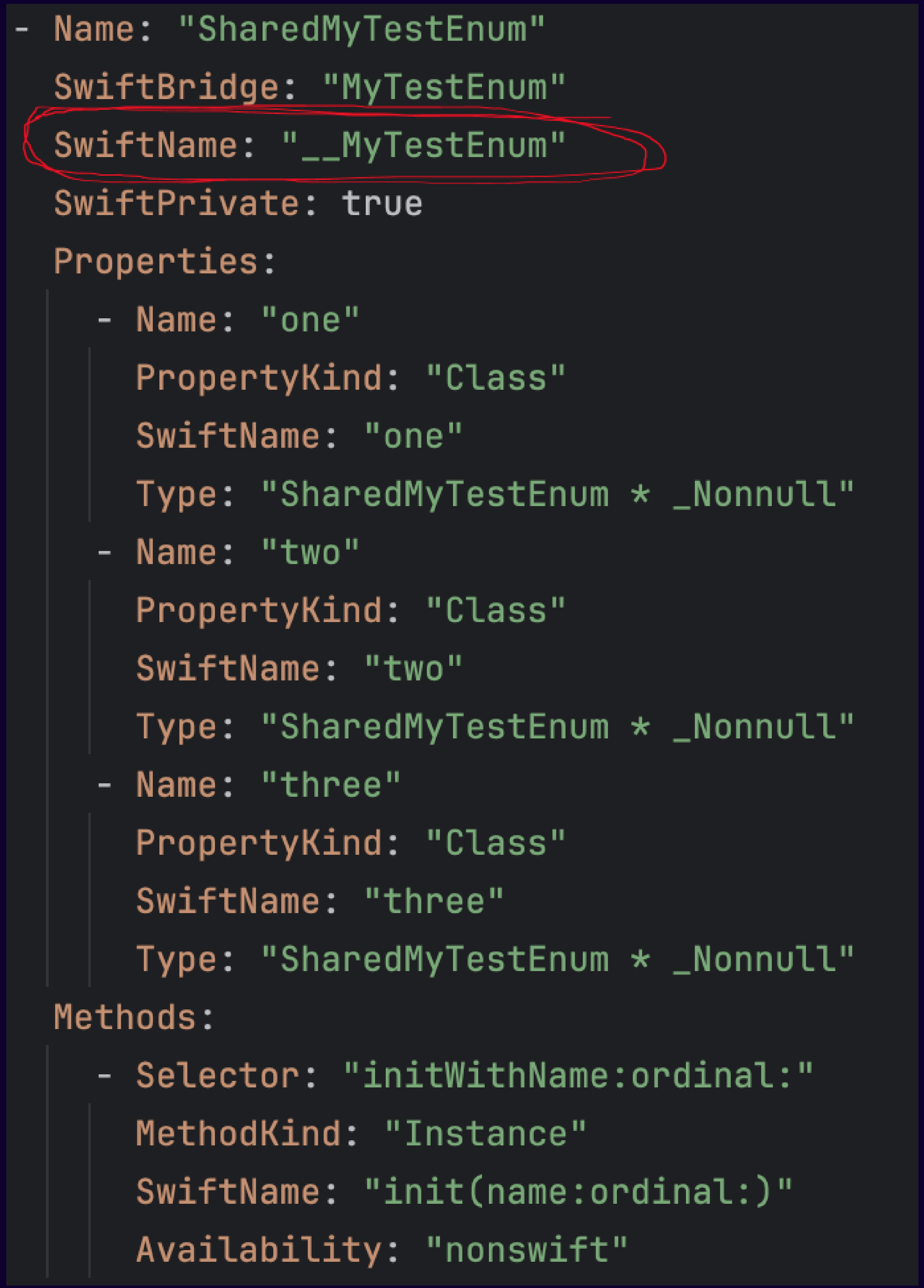


```
enum class MyTestEnum {
    One, Two, Three;
}
```



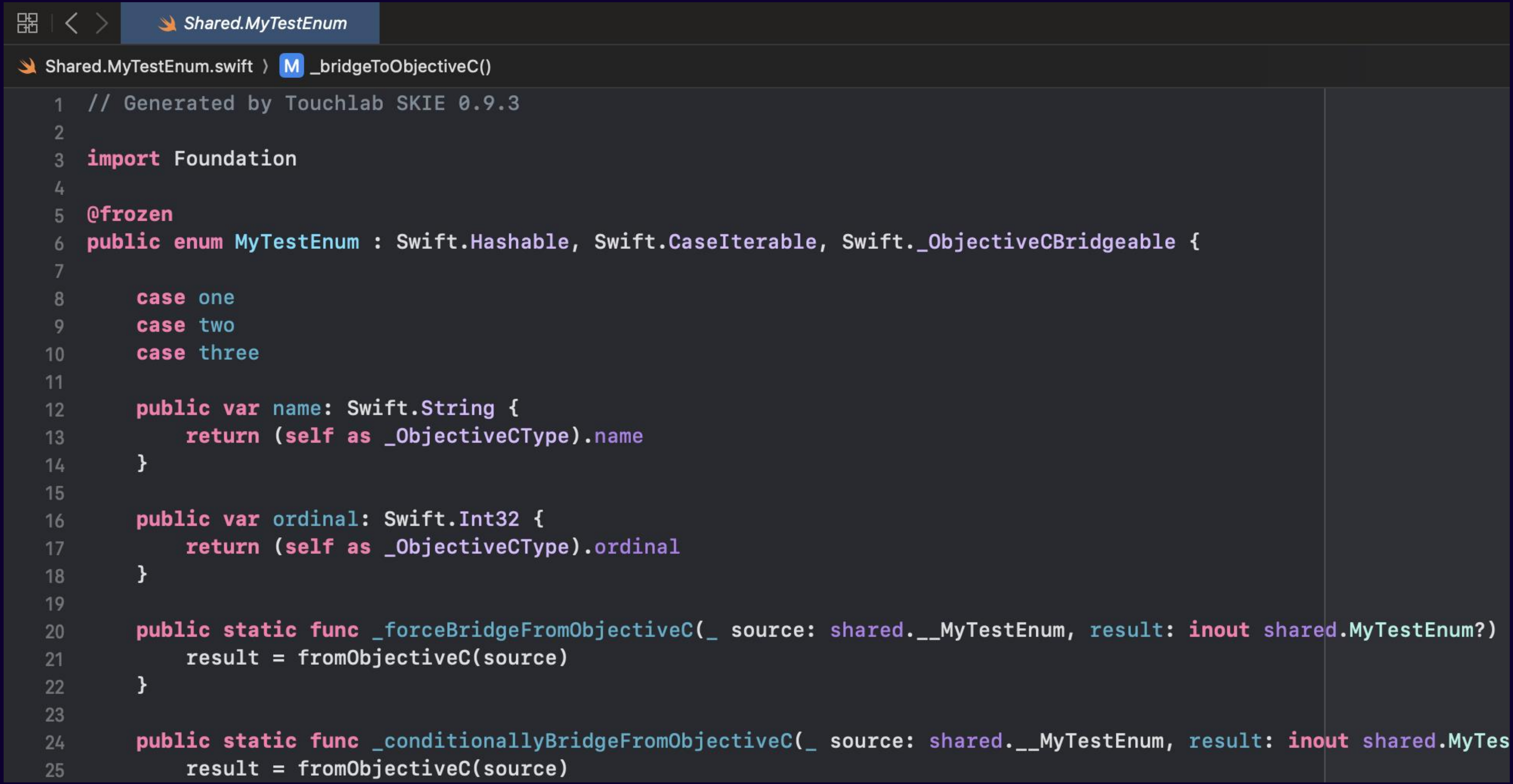
```
class MyTestEnum {
    static let one = 1;
    static let two = 2;
    static let three = 3;
}
```

Добавляем shared.apinotes



```
- Name: "SharedMyTestEnum"
  SwiftBridge: "MyTestEnum"
  SwiftName: "__MyTestEnum"
  SwiftPrivate: true
  Properties:
    - Name: "one"
      PropertyKind: "Class"
      SwiftName: "one"
      Type: "SharedMyTestEnum * _Nonnull"
    - Name: "two"
      PropertyKind: "Class"
      SwiftName: "two"
      Type: "SharedMyTestEnum * _Nonnull"
    - Name: "three"
      PropertyKind: "Class"
      SwiftName: "three"
      Type: "SharedMyTestEnum * _Nonnull"
  Methods:
    - Selector: "initWithName:ordinal:"
      MethodKind: "Instance"
      SwiftName: "init(name:ordinal:)"
      Availability: "nonswift"
```

Полноценный Swift enum



```swift
// Generated by Touchlab SKIE 0.9.3

import Foundation

@frozen
public enum MyTestEnum : Swift.Hashable, Swift.CaseIterable, Swift._ObjectiveCBridgeable {

    case one
    case two
    case three

    public var name: Swift.String {
        return (self as _ObjectiveCType).name
    }

    public var ordinal: Swift.Int32 {
        return (self as _ObjectiveCType).ordinal
    }

    public static func _forceBridgeFromObjectiveC(_ source: shared.__MyTestEnum, result: inout shared.MyTestEnum?)
        result = fromObjectiveC(source)
    }

    public static func _conditionallyBridgeFromObjectiveC(_ source: shared.__MyTestEnum, result: inout shared.MyTes
        result = fromObjectiveC(source)
```

Swift Bundling +
API Notes + Кодогенерация

Минусы SKIE

- еще одна точка отказа
- увеличенный размер приложения
- когда появится Swift-interop станет не нужен

Решение проблемы неудобного API

- Ничего не делать и ждать Swift-interop
- Использовать SKIE

Выводы

Проблемы с Kotlin Multiplatform в iOS

1. Какую интеграцию выбрать?

2. Что делать с ограничением umbrella-модуля?

3. Что делать с неудобным API?

Проблемы с Kotlin Multiplatform в iOS

1. Какую интеграцию выбрать?

✓ **Remote SPM**

2. Что делать с ограничением umbrella-модуля?

✓ **KMP-Платформа**

3. Что делать с неудобным API?

✓ **SKIE**

Финальное резюме

Спасибо!

Павел Шорохов

@pshorokhov