

Яндекс Карты

Kotlin Multiplatform в мобильных Яндекс.Картах с позиции iOS разработчика

Михаил Куренков, iOS разработчик

В этом докладе не будет:

В этом докладе не будет:

- › Пошаговой инструкции, как собрать прототип на Kotlin MPP
- › Сборки кода на Kotlin под Linux, RaspberryPi или холодильник
- › Рассказов о том, как прекрасен Kotlin MPP

**Будет опыт внедрения Kotlin MPP
в большой проект**

Зачем?

Зачем?

Ускорить разработку

Писать код один раз

Писать код один раз

Изменять код в одном месте

Писать код один раз

Изменять код в одном месте

Меньше итераций тестирования

План

Что такое Kotlin Multiplatform?

Интероп Kotlin - Swift

Нативные библиотеки в Kotlin

Настройка процессов

Результат применения Kotlin Multiplatform

Что такое Kotlin Multiplatform?

С чем нам предстоит работать?

Kotlin Multiplatform - набор инструментов

Kotlin Multiplatform - набор инструментов

Компиляторы

Kotlin/JVM

Kotlin/Native

Gradle Plugins

org.jetbrains.kotlin.multiplatform
org.jetbrains.kotlin.native.cocoapods

Plugins for IDE

Kotlin Multiplatform Mobile
for Android Studio

Kotlin/Native
for AppCode

**Собирается Kotlin Multiplatform
с помощью Gradle**

Собирается Kotlin Multiplatform с помощью Gradle

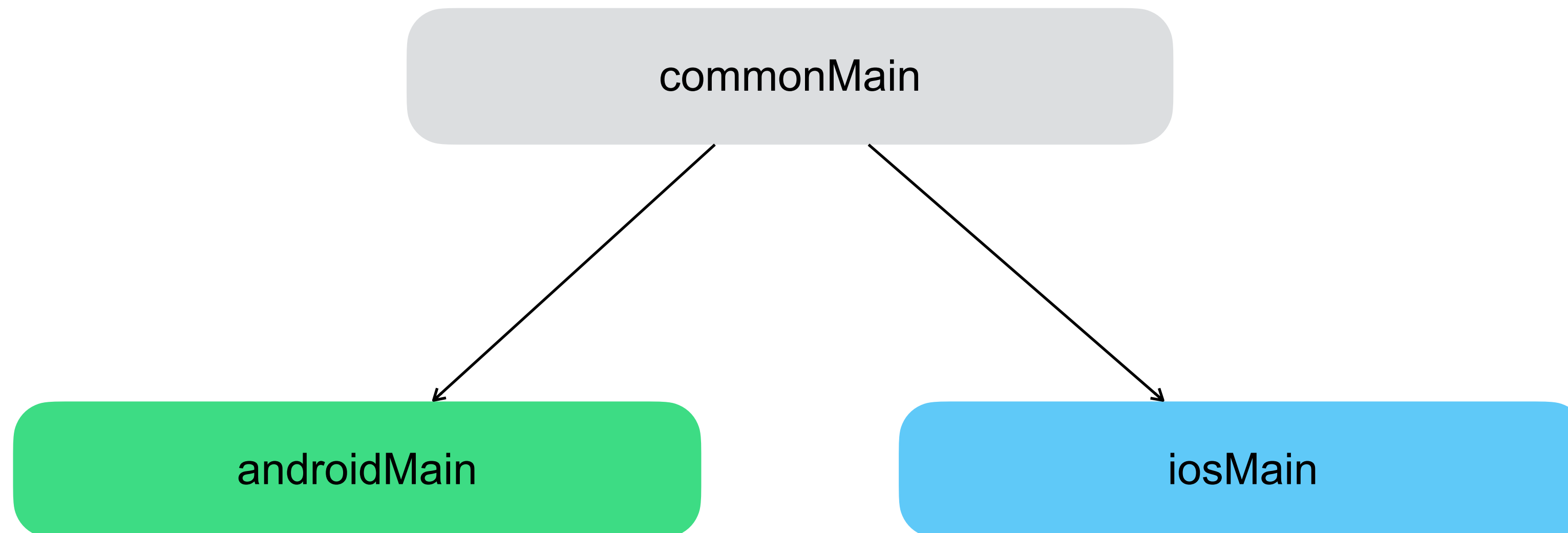
Нужно будет подружиться с Gradle

Собирается Kotlin Multiplatform с помощью Gradle

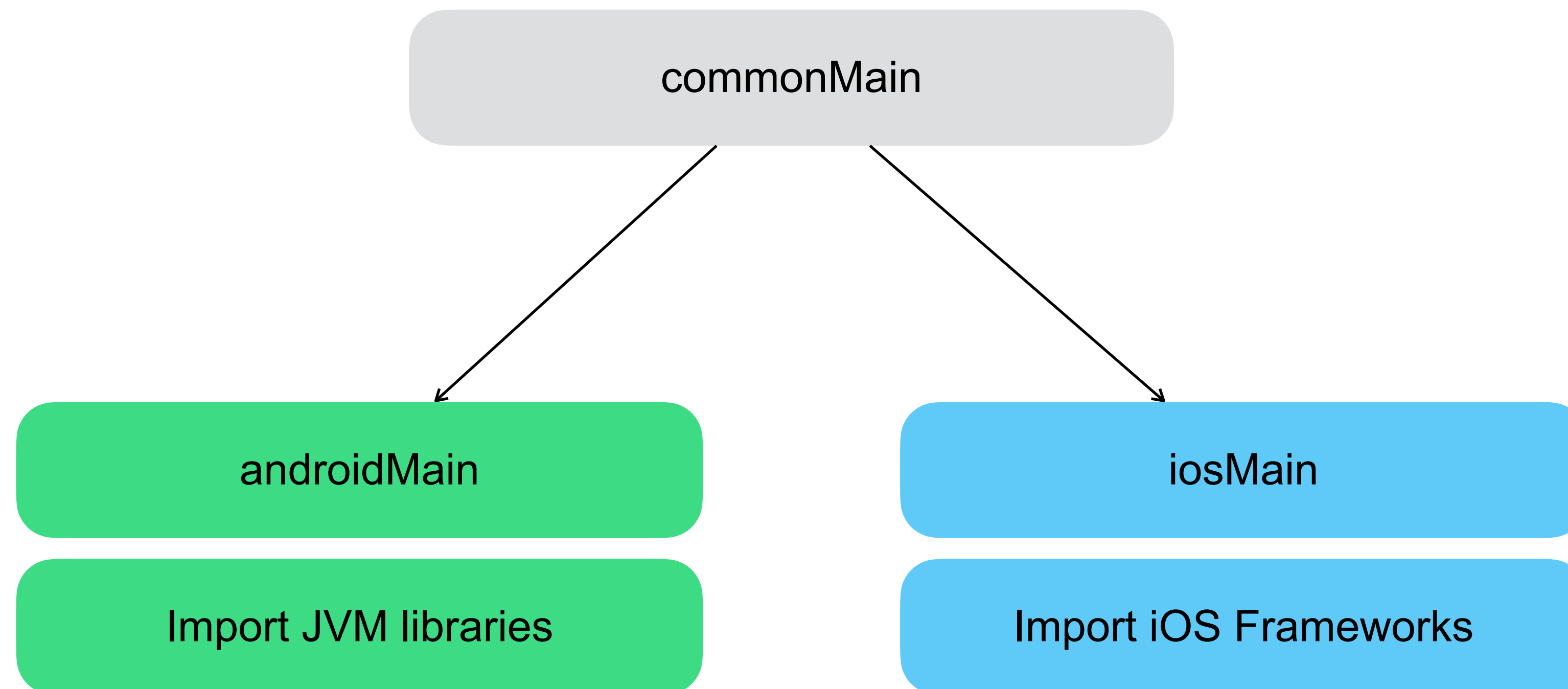
Нужно будет подружиться с Gradle

... но лучше с Android разработчиком,
который шарит в Gradle

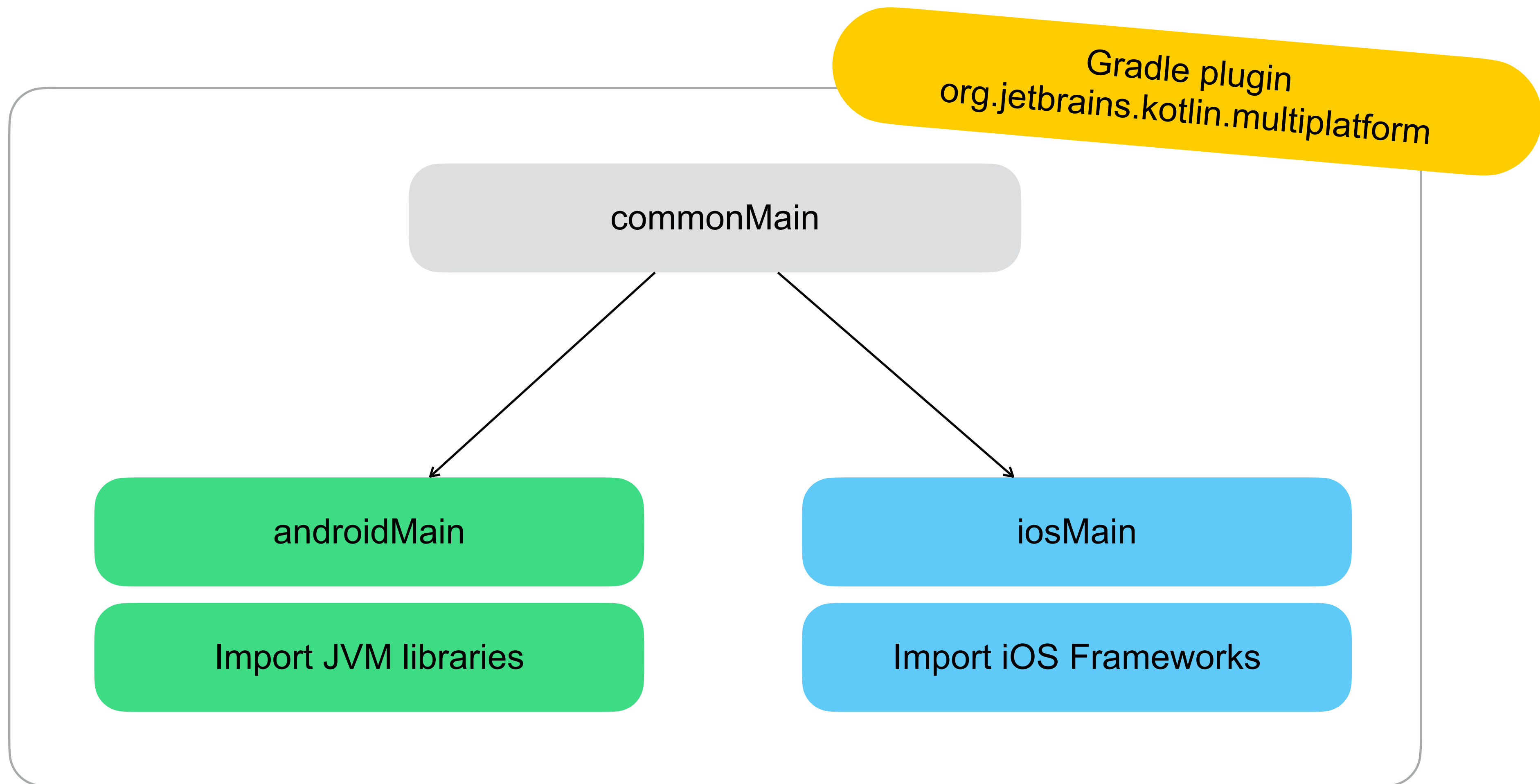
Структура Kotlin Multiplatform библиотеки



Структура Kotlin Multiplatform библиотеки



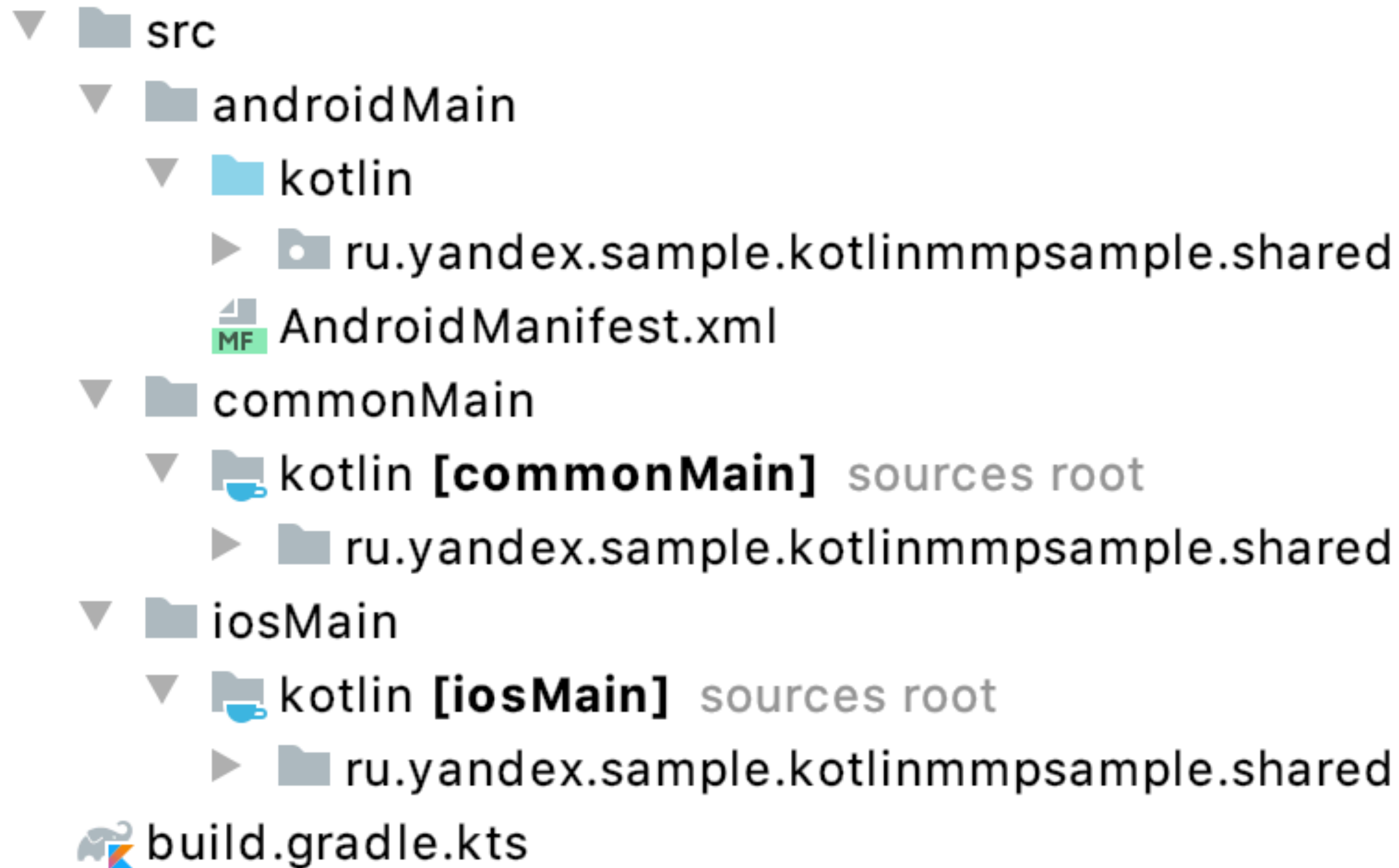
Структура Kotlin Multiplatform библиотеки



build.gradle.kts

```
plugins {  
    id("org.jetbrains.kotlin.multiplatform").version("1.4.0")  
}  
  
kotlin {  
    android()  
    ios()  
}
```

Структура Kotlin Multiplatform библиотеки



expect & actual

```
// commonMain  
expect class A {  
    expect fun f()  
}
```

expect & actual

```
// commonMain  
expect class A {  
    expect fun f()  
}
```

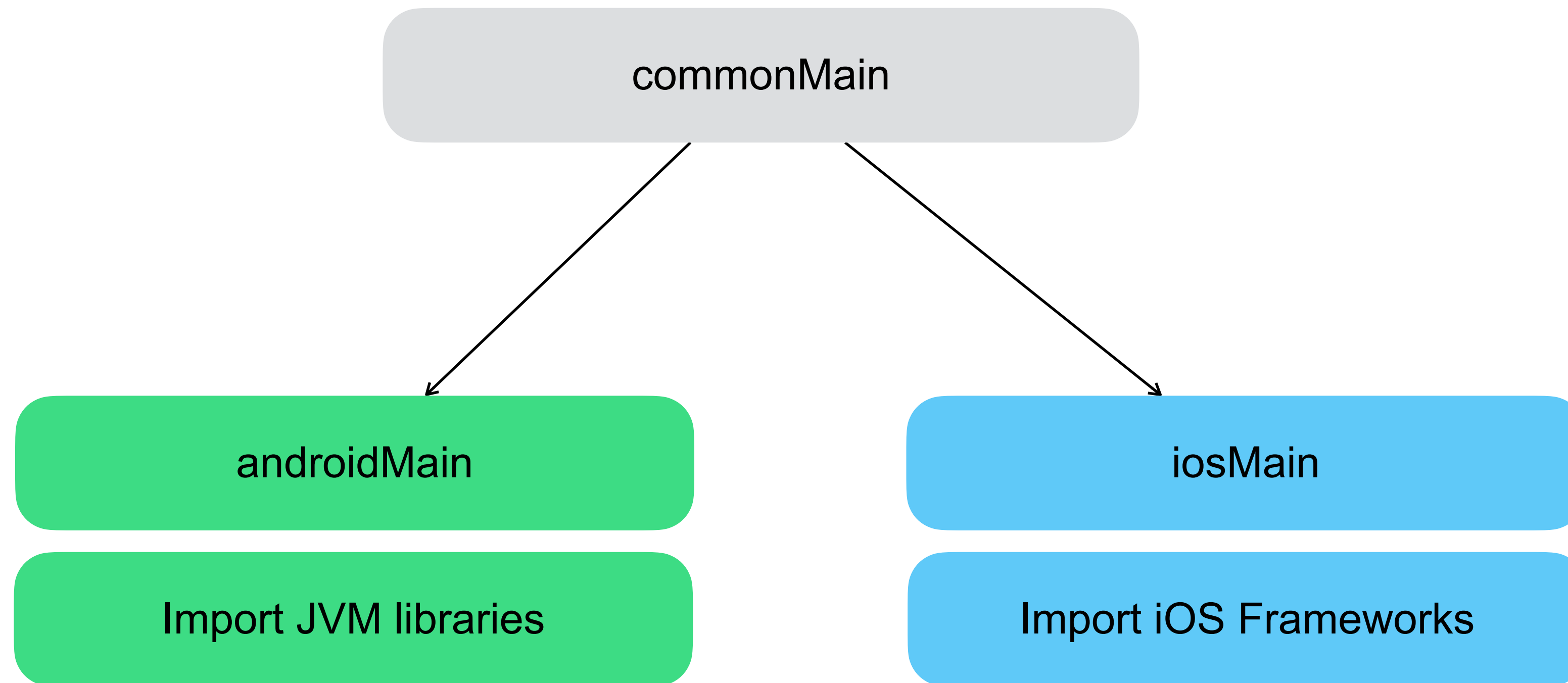
// iosMain

```
actual class A {  
    actual fun f() = println("iOS")  
    fun g() = doSome()  
}
```

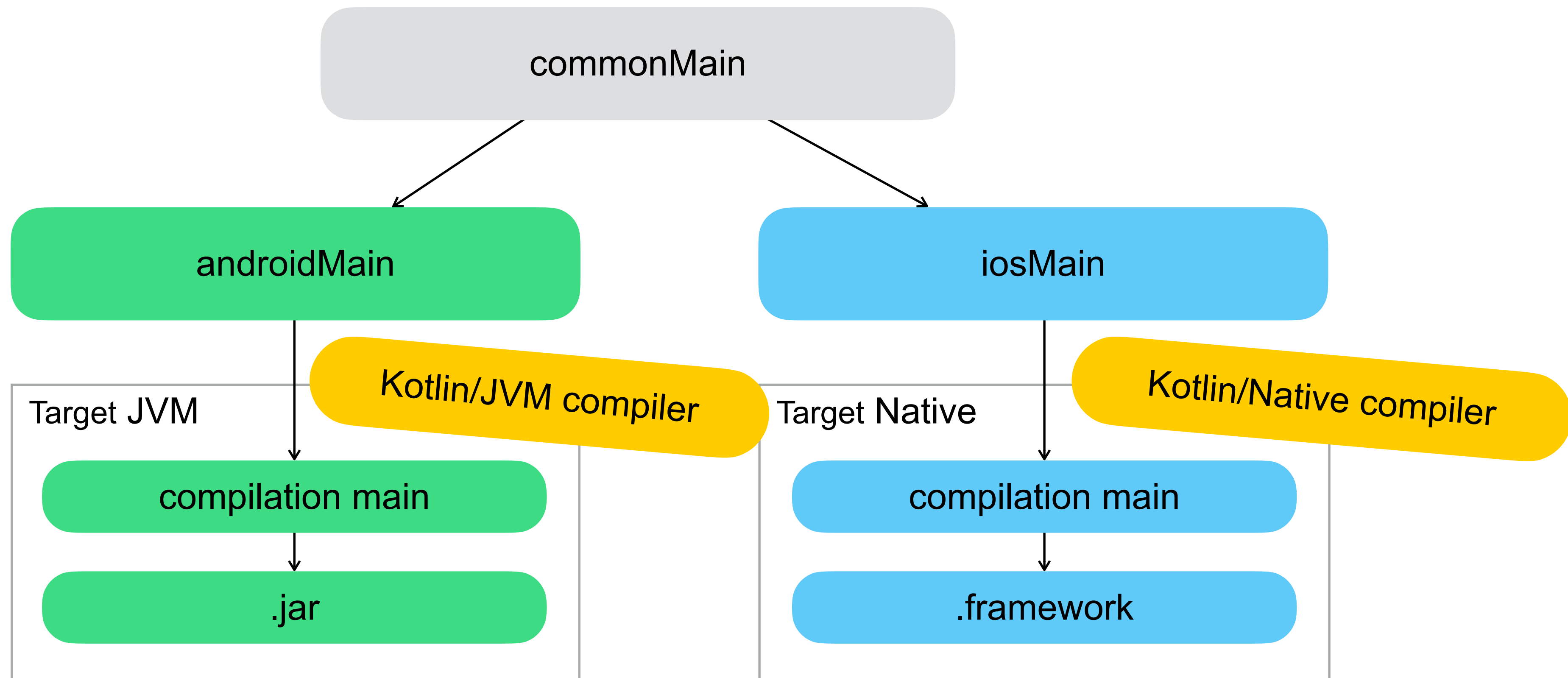
// androidMain

```
actual class A {  
    actual fun f() = println("Android")  
}
```

Сборка



Сборка



Подключение

Подключение

на Android

› Через Gradle

Подключение

на iOS

- › Xcode Target с вызовом Gradle в кастомной фазе сборки
- › Через CocoaPods и плагин `org.jetbrains.kotlin.native.cocoapods`

на Android

- › Через Gradle

build.gradle.kts

```
plugins {  
    id("org.jetbrains.kotlin.multiplatform").version("1.4.0")  
    id("org.jetbrains.kotlin.native.cocoapods").version("1.4.0")  
}  
  
kotlin {  
    android()  
    ios()  
    cocoapods {  
        summary = "Kotlin CocoaPods library"  
        homepage = "https://..."  
    }  
}
```

Сгенерированный podspec

```
Pod::Spec.new do |spec|
  spec.static_framework      = true
  spec.vendored_frameworks   = "../multiplatform.framework"
  spec.libraries              = "c++"
  ...
  spec.script_phases = [
    {
      :name => 'Build shared',
      :execution_position => :before_compile,
      :shell_path => '/bin/sh',
      :script => <<-SCRIPT
        set -ev
        REPO_ROOT="$PODS_TARGET_SRCROOT"
        "$REPO_ROOT/../gradlew" -p "$REPO_ROOT" :shared:syncFramework \
          -Pkotlin.native.cocoapods.target=$KOTLIN_TARGET \
          -Pkotlin.native.cocoapods.configuration=$CONFIGURATION \
          -Pkotlin.native.cocoapods.cflags="$OTHER_CFLAGS" \
          -Pkotlin.native.cocoapods.paths.headers="$HEADER_SEARCH_PATHS" \
          -Pkotlin.native.cocoapods.paths.frameworks="$FRAMEWORK_SEARCH_PATHS"
      SCRIPT
    }
  ]
end
```

Эволюция Kotlin CocoaPods интеграции

Этап 1

Возможности:

1. генерация def
2. генерация podspec
3. cinterop
4. frameworks

Ограничения:

1. Требовалось наличие xcproject, Podfile
2. Ручной вызов *pod install* и запуск сборки в Xcode
3. Только pod'ы из репо Cocoapods

Эволюция Kotlin CocoaPods интеграции

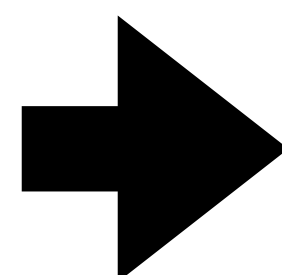
Этап 1

Возможности:

1. генерация def
2. генерация podspec
3. cinterop
4. frameworks

Ограничения:

1. Требовалось наличие xcproject, Podfile
2. Ручной вызов *pod install* и запуск сборки в Xcode
3. Только pod'ы из репо Cocoapods



Эволюция Kotlin CocoaPods интеграции

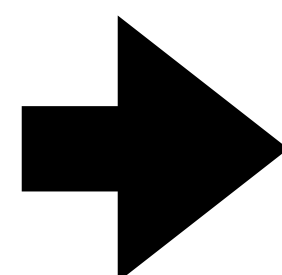
Этап 1

Возможности:

1. генерация def
2. генерация podspec
3. cinterop
4. frameworks

Ограничения:

1. Требовалось наличие xcproject, Podfile
2. Ручной вызов *pod install* и запуск сборки в Xcode
3. Только pod'ы из репо Cocoapods



Этап 2

Эволюция Kotlin CocoaPods интеграции

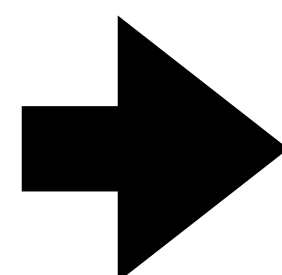
Этап 1

Возможности:

1. генерация def
2. генерация podspec
3. cinterop
4. frameworks

Ограничения:

1. Требовалось наличие xcproject, Podfile
2. Ручной вызов *pod install* и запуск сборки в Xcode
3. Только pod'ы из репо Cocoapods



Этап 2

Возможности:

1. Работа без xcproject, Podfile
2. Поддержка локальных pod'ов
3. Улучшенная поддержка в IDE при импорте проекта

Ограничения:

1. Избыточная рекомпиляция pod'ов
2. Покрыт не весь DSL для pod-зависимостей

Эволюция Kotlin CocoaPods интеграции

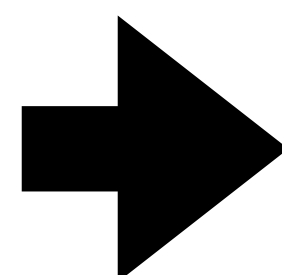
Этап 1

Возможности:

1. генерация def
2. генерация podspec
3. cinterop
4. frameworks

Ограничения:

1. Требовалось наличие xcproject, Podfile
2. Ручной вызов *pod install* и запуск сборки в Xcode
3. Только pod'ы из репо Cocoapods



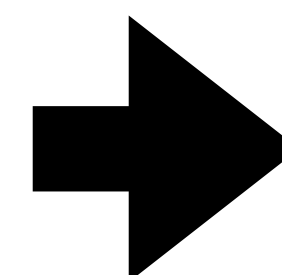
Этап 2

Возможности:

1. Работа без xcproject, Podfile
2. Поддержка локальных pod'ов
3. Улучшенная поддержка в IDE при импорте проекта

Ограничения:

1. Избыточная рекомпиляция pod'ов
2. Покрыт не весь DSL для pod-зависимостей



Эволюция Kotlin CocoaPods интеграции

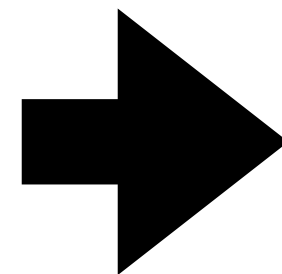
Этап 1

Возможности:

1. генерация def
2. генерация podspec
3. cinterops
4. frameworks

Ограничения:

1. Требовалось наличие xcproject, Podfile
2. Ручной вызов *pod install* и запуск сборки в Xcode
3. Только pod'ы из репо Cocoapods



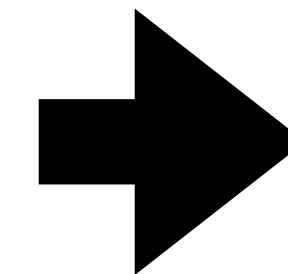
Этап 2

Возможности:

1. Работа без xcproject, Podfile
2. Поддержка локальных pod'ов
3. Улучшенная поддержка в IDE при импорте проекта

Ограничения:

1. Избыточная рекомпиляция pod'ов
2. Покрыт не весь DSL для pod-зависимостей



Этап 3

Эволюция Kotlin CocoaPods интеграции

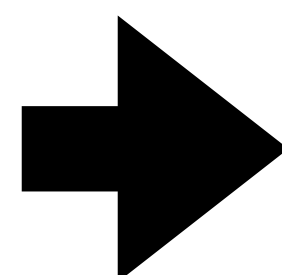
Этап 1

Возможности:

1. генерация def
2. генерация podspec
3. cinterops
4. frameworks

Ограничения:

1. Требовалось наличие xcproject, Podfile
2. Ручной вызов *pod install* и запуск сборки в Xcode
3. Только pod'ы из репо Cocoapods



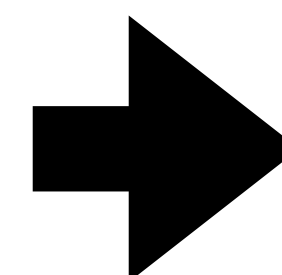
Этап 2

Возможности:

1. Работа без xcproject, Podfile
2. Поддержка локальных pod'ов
3. Улучшенная поддержка в IDE при импорте проекта

Ограничения:

1. Избыточная recompilation pod'ов
2. Покрыт не весь DSL для pod-зависимостей



Этап 3

Возможности:

1. Добавление pod'а из git repo, пользовательского spec repo, архива по http
2. Оптимизация Up-To-Date проверок
3. Улучшения в стабильности

Ограничения:

1. Отсутствие транзитивности cocoapods-зависимостей в Kotlin
2. Подключение не более 1 Kotlin-пода в Xcode проект
3. Невозможность публикации Kotlin-пода

IDE

Android Studio

Позволяет собирать
Android и iOS

Позволяет разрабатывать
только на Kotlin

AppCode

Позволяет собирать
только iOS

Позволяет разрабатывать
на Kotlin и Swift

Kotlin Multiplatform - набор инструментов

Kotlin Multiplatform - набор инструментов

› Компиляторы Kotlin/JVM и Kotlin/Native

Kotlin Multiplatform - набор инструментов

- › Компиляторы Kotlin/JVM и Kotlin/Native
- › Сборка с помощью Gradle

Kotlin Multiplatform - набор инструментов

- › Компиляторы Kotlin/JVM и Kotlin/Native
- › Сборка с помощью Gradle
- › Плагин для подключения через CocoaPods

Kotlin Multiplatform - набор инструментов

- › Компиляторы Kotlin/JVM и Kotlin/Native
- › Сборка с помощью Gradle
- › Плагин для подключения через CocoaPods
- › Поддержка в Android Studio и AppCode

Интероп Kotlin - Swift

Как взаимодействовать с уже написанным кодом

**Kotlin / Native framework
предоставляет
Objective-C интерфейс**

Интероп из Kotlin в Swift

В Swift недоступны

- › Abstract classes
- › Inline classes

Частично недоступны

- › Sealed Classes и Enums
- › Generics

Kotlin Sealed Class - аналог Swift Enum

// Kotlin

```
sealed class SealedClass {  
    object Object: SealedClass()  
    data class DataClass(val value: Double): SealedClass()  
    class ClassWithValue(val value: Double): SealedClass()  
}
```

// Swift

```
enum EnumWrapperForSealedClass {  
    case Object  
    case DataClass(Double)  
    case ClassWithValue(Double)  
}
```

Sealed Class и Enum

ВЫГЛЯДИТ как иерархия классов

```
__attribute__((swift_name("SealedClass")))  
@interface KNFSealedClass : KNFBase  
@end;  
  
__attribute__((swift_name("SealedClass.FirstChild")))  
@interface KNFSealedClassFirstChild : KNFSealedClass  
...  
@end;  
  
__attribute__((swift_name("SealedClass.SecondChild")))  
@interface KNFSealedClassSecondChild : KNFSealedClass  
...  
@end;
```

Sealed Class и Enum

ВЫГЛЯДИТ как иерархия классов

```
switch sealedClass {  
  
    case let firstChild as SealedClass.FirstChild:  
        print(firstChild.value)  
  
    case let secondChild as SealedClass.SecondChild:  
        print(secondChild.value)  
  
    default:  
        // Необходим, даже когда мы обрабатываем все case-ы.  
        break  
  
}
```

Generics в Kotlin и Swift разные

Generics в Kotlin и Swift разные

**При интеропе Kotlin generics
превратятся в Objective-C generics**

Пропадают generics в протоколах

Пропадают generics в протоколах

**Из аргумента generic пропадают
optionals, variance annotation, star projection**

Исправляем интерфейсы в actual-ax

// commonMain

```
fun getGenericData(): GenericData<String?> {  
    return GenericData("Hello from Kotlin!")  
}
```

// iosMain

```
class OptionalWrapper<T : Any>(val value: T?)
```

```
public fun getGenericDataIos(): GenericData<OptionalWrapper<String>> {  
    val value = getGenericData()  
    return GenericData(OptionalWrapper(value.data))  
}
```

Objective-C representation

```
@interface OptionalWrapper<T> : KotlinBase
```

```
@property (readonly) T _Nullable value;
```

```
@end;
```

```
+ (GenericData<NSString *> *)getGenericData;
```

```
+ (GenericData<OptionalWrapper<NSString *> *> *)getGenericDataLos;
```

Objective-C representation

```
@interface OptionalWrapper<T> : KotlinBase
@property (readonly) T _Nullable value;
@end;
```

Optional пропал из аргумента шаблона

```
+ (GenericData<NSString *> *)getGenericData;
```

```
+ (GenericData<OptionalWrapper<NSString *> *> *)getGenericDataLos;
```

Objective-C representation

```
@interface OptionalWrapper<T> : KotlinBase
@property (readonly) T _Nullable value;
@end;
```

Optional сохраняется,
благодаря аннотации **_Nullable**

```
+ (GenericData<NSString *> *)getGenericData;
```

Optional пропал из аргумента шаблона

```
+ (GenericData<OptionalWrapper<NSString *> *> *)getGenericDataLos;
```

Для комфортного использования Kotlin из Swift

**Не используем Sealed Classes и Enums
в качестве возвращаемых значений**

**Не используем Sealed Classes и Enums
в качестве возвращаемых значений**

Не используем generics в interfaces

**Не используем Sealed Classes и Enums
в качестве возвращаемых значений**

Не используем generics в interfaces

Оборачиваем generics

**Не используем Sealed Classes и Enums
в качестве возвращаемых значений**

Не используем generics в interfaces

Оборачиваем generics

Оборачиваем вызовы suspend functions

Нативные библиотеки в Kotlin

**Kotlin/Native обеспечивает
совместимость с Objective-C**

**Kotlin/Native обеспечивает
совместимость с Objective-C**

Но не с Swift

**Kotlin/Native обеспечивает
совместимость с Objective-C**

**Из библиотеки на Swift будет доступен
только API экспортированный в Objective-C**

**Kotlin/Native обеспечивает
совместимость с Objective-C**

**Из библиотеки на Swift будет доступен
только API экспортированный в Objective-C
с помощью аннотации @objc**

iOS SDK доступен в Kotlin Multiplatform

iOS SDK доступен в Kotlin Multiplatform

› Сложности с наследованием

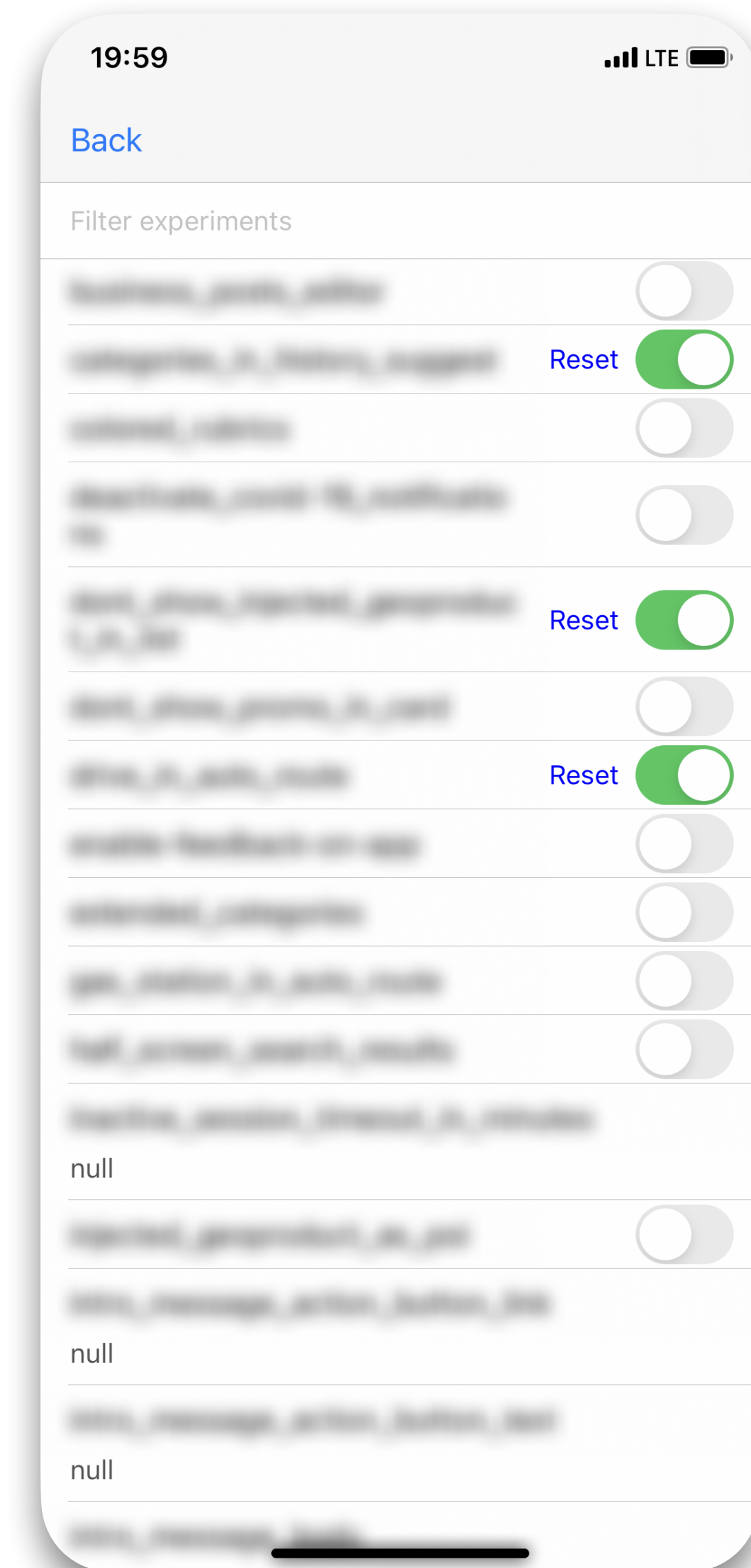
iOS SDK доступен в Kotlin Multiplatform

- › Сложности с наследованием
- › Нужно использовать специальные аннотации для настройки взаимодействия Kotlin с Objective-C runtime

iOS SDK доступен в Kotlin Multiplatform

- › Сложности с наследованием
- › Нужно использовать специальные аннотации для настройки взаимодействия Kotlin с Objective-C runtime
- › Отсутствуют некоторые ключевые слова и макросы, такие как *#selector*, *If @available*, etc

- › UINavigationController с UITableView
- › Переиспользование ячеек
- › Обработка нажатия на кнопку



Сложности с наследованием

```
class MyViewController:  
    UIViewController,  
    UITableViewDataSourceProtocol,  
    UITableViewDelegateProtocol  
{  
    . . .  
}
```

Class 'MyViewController' is not abstract and does not implement abstract member
@ObjCMethod public abstract fun `class`() : ObjCClass?
defined in platform.UIKit.UITableViewDelegateProtocol

Сложности с наследованием

```
class MyViewController: UIViewController {  
    val tableViewDataSource = MyDataSource()  
    val tableViewDelegate = MyTableViewDelegate()  
}
```

```
class MyDataSource: NSObject(), UITableViewDataSourceProtocol {  
}
```

```
class MyTableViewDelegate: NSObject(), UITableViewDelegateProtocol {  
}
```

Сложности с наследованием

```
class MyTableViewDelegate: NSObject(), UITableViewDelegateProtocol {  
  
    override fun tableView(tableView: UITableView,  
                           didSelectRowAtIndexPath: NSIndexPath): Unit { . . . }  
  
    override fun tableView(tableView: UITableView,  
                           heightForRowAtIndexPath: NSIndexPath): CGFloat { . . . }  
}
```

Conflicting overloads:

```
public open fun tableView(tableView: UITableView, didSelectRowAtIndexPath: NSIndexPath):  
Unit defined in MyTableViewDelegate,  
public open fun tableView(tableView: UITableView, heightForRowAtIndexPath: NSIndexPath):  
CGFloat /* = Double */ defined in MyTableViewDelegate
```

Сложности с наследованием

```
class MyTableViewDelegate: NSObject(), UITableViewDelegateProtocol {  
  
    override fun tableView(tableView: UITableView,  
        didSelectRowAtIndexPath: NSIndexPath): Unit { . . . }  
  
    override fun tableView(tableView: UITableView,  
        heightForRowAtIndexPath: NSIndexPath): CGFloat { . . . }  
}
```

Conflicting overloads:

```
public open fun tableView(tableView: UITableView, didSelectRowAtIndexPath: NSIndexPath):  
Unit defined in MyTableViewDelegate,  
public open fun tableView(tableView: UITableView, heightForRowAtIndexPath: NSIndexPath):  
CGFloat /* = Double */ defined in MyTableViewDelegate
```

Сложности с наследованием

```
class MyTableViewDelegate: NSObject(), UITableViewDelegateProtocol {  
  
    @Suppress("CONFLICTING_OVERLOADS")  
    override fun tableView(tableView: UITableView,  
                           didSelectRowAtIndexPath: NSIndexPath): Unit { . . . }  
  
    @Suppress("CONFLICTING_OVERLOADS")  
    override fun tableView(tableView: UITableView,  
                           heightForRowAtIndexPath: NSIndexPath): CGFloat { . . . }  
  
}
```

Сложности с наследованием

```
@ExportObjCClass
```

```
class MyTableViewCell : UITableViewCell {
```

```
    @ObjCObjectBase.OverrideInit
```

```
    constructor(style: UITableViewCellStyle, reuseIdentifier: String?):
```

```
        super(style, reuseIdentifier)
```

```
}
```

@ObjCObjectBase.OverrideInit - для переопределения Obj-C конструктора

Взаимодействие с Objective-C runtime

```
@ExportObjCClass
```

```
class MyTableViewCell : UITableViewCell {
```

```
    @ObjCObjectBase.OverrideInit
```

```
    constructor(style: UITableViewCellStyle, reuseIdentifier: String?):
```

```
        super(style, reuseIdentifier)
```

```
}
```

@ExportObjCClass - делает класс видимым для поиска во время выполнения

Взаимодействие с Objective-C runtime

```
@ExportObjCClass  
class MyTableViewCell : UITableViewCell {  
    companion object Meta : UITableViewCellMeta()  
}
```

```
tableView.registerClass(  
    cellClass = MyTableViewCell,  
    forCellReuseIdentifier = "MyTableViewCell"  
)
```

Взаимодействие с Objective-C runtime

@ObjCAction

private fun onTap() { . . . }

```
button.addTarget(  
    target = this,  
    action = NSSelectorFromString("onTapped"),  
    forControlEvents = UIControlEventTouchUpInside  
)
```

Отсутствует #selector

@IBAction

private fun onTap() { . . . }

```
button.addTarget(  
    target = this,  
    action = NSSelectorFromString("onTapped"),  
    forControlEvents = UIControlEventTouchUpInside  
)
```

Нет аналога If @available

```
val version = UIDevice.currentDevice().systemVersion.toDoubleOrNull()
if (version != null && version >= 13) {
    . . .
}
```

Использовать нативные библиотеки в Kotlin
ВОЗМОЖНО

Использовать нативные библиотеки в Kotlin
ВОЗМОЖНО

Сложные места есть, но для каждого из них
существует тривиальное решение

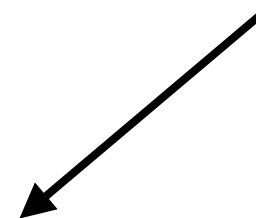
**Для нас UI на Kotlin/Native
остается экспериментом**

**Для нас UI на Kotlin/Native
остается экспериментом**

Из-за уже готовой нативной дизайн системы

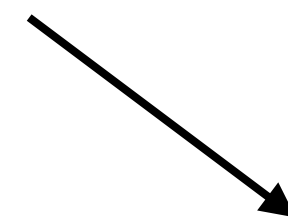
Как подключать сторонние библиотеки?

Как подключать сторонние библиотеки?



через

CocoaPods



с применением

cinterop

Подключение библиотек через Cocoapods

```
plugins {  
    id("org.jetbrains.kotlin.multiplatform").version("1.4.0")  
    id("org.jetbrains.kotlin.native.cocoapods").version("1.4.0")  
}  
  
kotlin {  
    cocoapods {  
        summary = "Kotlin CocoaPods library"  
        homepage = "https://..."  
  
        pod("AFNetworking", "~> 4.0.0")  
        pod("DevelopmentPod", "1.0", project.file("../Pod.podspec"))  
    }  
}
```

Подключение библиотек через CocoaPods

Это просто



CocoaPods
integration

Подключение библиотек через Cocoapods

Не работал
со статическими библиотеками

Исправлено в Kotlin 1.4.20

<https://youtrack.jetbrains.com/issue/KT-42531>



*CocoaPods
integration*

Подключение библиотек через Cocoapods

Не работал
со статическими библиотеками

Исправлено в Kotlin 1.4.20

<https://youtrack.jetbrains.com/issue/KT-42531>

В настоящее время не
реализована транзитивность



*CocoaPods
integration*

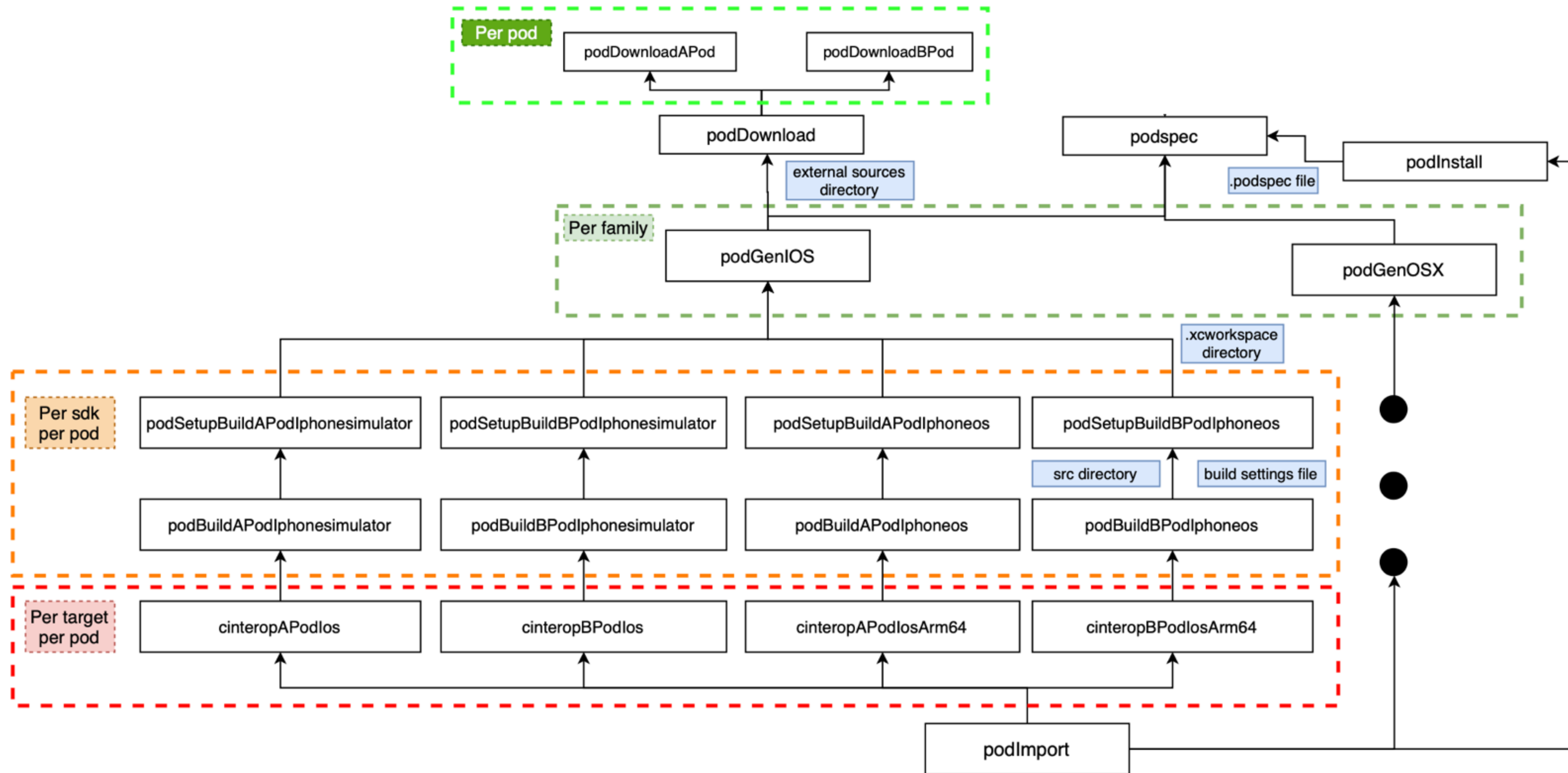
Подключение библиотек через cinterop

1. Создать .def file
с описанием бриджинга
2. Объявить зависимость от cinterop
в build.gradle



*Kotlin/Native
interoperability*

Kotlin CocoaPods Integration in depth



Использование нативных библиотек с идентичным API на разных платформах

Facebook SDK, Google Analytics, Yandex.MapKit, etc.

Два лица YandexMapKit

// iOS

```
@interface YMKLayer : NSObject
@property (readonly, getter=isValid) BOOL valid;
- (void)activateWithOn:(BOOL)on;
- (void)remove;
@end
```

// Android

```
public interface Layer {
    public boolean isValid();
    public void activate(boolean on);
    public void remove();
}
```

expect & actual

// commonMain

```
expect class Layer {  
    fun isValid(): Boolean  
    fun activateWithOn(on: Boolean)  
    fun remove()  
}
```

// iosMain

```
actual typealias Layer =  
    platform.YandexMapsMobile.YMKLayer
```

// androidMain

```
actual class Layer(val impl: com.yandex.mapkit.layers.Layer) {  
    actual fun isValid(): Boolean = impl.isValid()  
    actual fun activateWithOn(on: Boolean) = impl.activate(on)  
    actual fun remove() = impl.remove()  
}
```

// iosMain

```
actual typealias Layer =  
    platform.YandexMapsMobile.YMKLayer
```

// androidMain

```
actual class Layer(  
    val impl: com.yandex.mapkit.layers.Layer) {  
    actual fun isValid(): Boolean  
        = impl.isValid()  
    actual fun activateWithOn(on: Boolean)  
        = impl.activate(on)  
    actual fun remove()  
        = impl.remove()  
}
```

// androidMain

```
actual typealias Layer =  
    com.yandex.mapkit.layers.Layer
```

// iosMain

```
actual interface Layer {  
    actual fun isValid(): Boolean  
    actual fun activate(on: Boolean)  
    actual fun remove()  
}  
class LayerAdapter(val impl: YMKLayer): Layer {  
    fun isValid(): Boolean  
        = impl.isValid()  
    fun activate(on: Boolean)  
        = impl.activateWithOn(on)  
    fun remove()  
        = impl.remove()  
}
```

// iosMain

```
actual typealias Layer =  
    platform.YandexMapsMobile.YMKLayer
```

// androidMain

```
actual class Layer(  
    val impl: com.yandex.mapkit.layers.Layer) {  
    actual fun isValid(): Boolean  
        = impl.isValid()  
    actual fun activateWithOn(on: Boolean)  
        = impl.activate(on)  
    actual fun remove()  
        = impl.remove()  
}
```

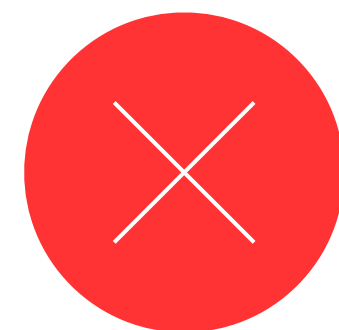


// androidMain

```
actual typealias Layer =  
    com.yandex.mapkit.layers.Layer
```

// iosMain

```
actual interface Layer {  
    actual fun isValid(): Boolean  
    actual fun activate(on: Boolean)  
    actual fun remove()  
}  
class LayerAdapter(val impl: YMKLayer): Layer {  
    fun isValid(): Boolean  
        = impl.isValid()  
    fun activate(on: Boolean)  
        = impl.activateWithOn(on)  
    fun remove()  
        = impl.remove()  
}
```



iOS SDK доступен в Kotlin Multiplatform

iOS SDK доступен в Kotlin Multiplatform

**Подключаем сторонние библиотеки
через плагин для CocoaPods или C-interop**

iOS SDK доступен в Kotlin Multiplatform

**Подключаем сторонние библиотеки
через плагин для CocoaPods или C-interop**

**Для использования в общем коде библиотек
с идентичным API на разных платформах
пишем обертки с помощью expect/actual**

Настройка процессов

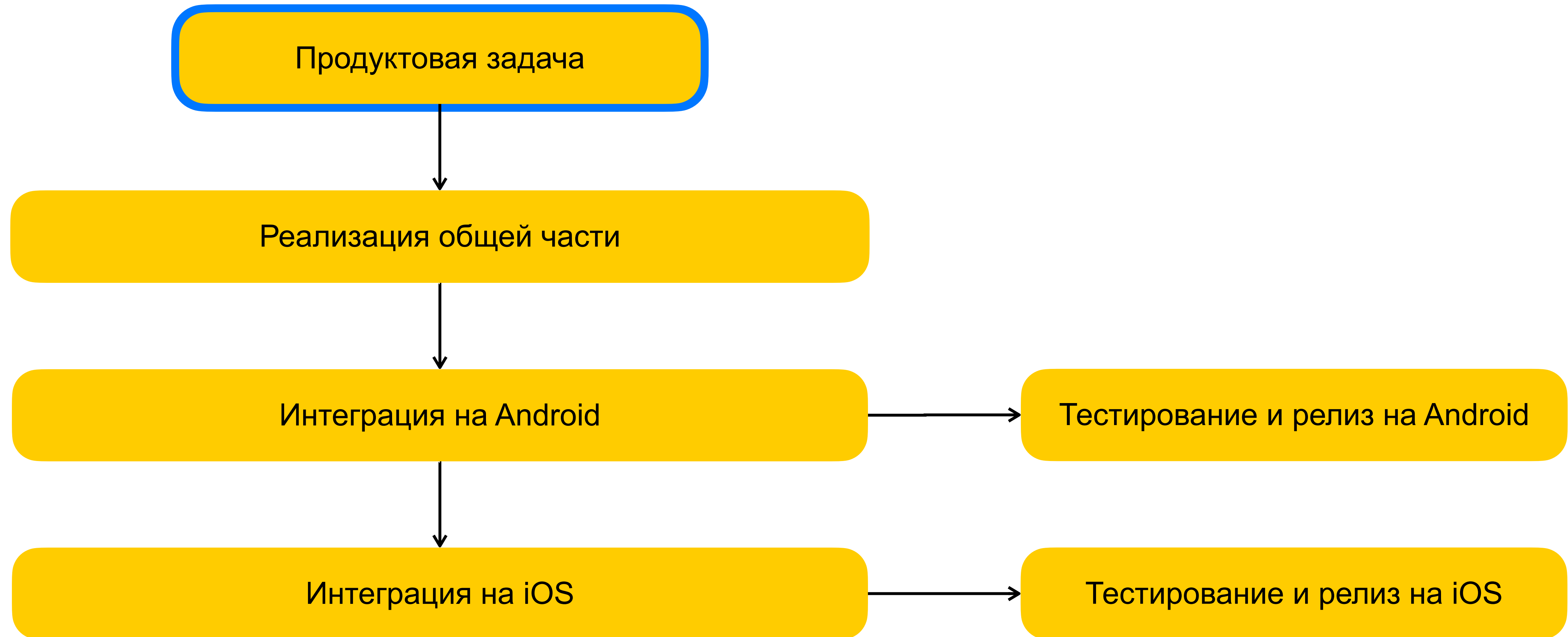
Рецепт успешного использования Kotlin MPP
в большом проекте

**Изменения в мультиплатформе
ломают и iOS, и Android**

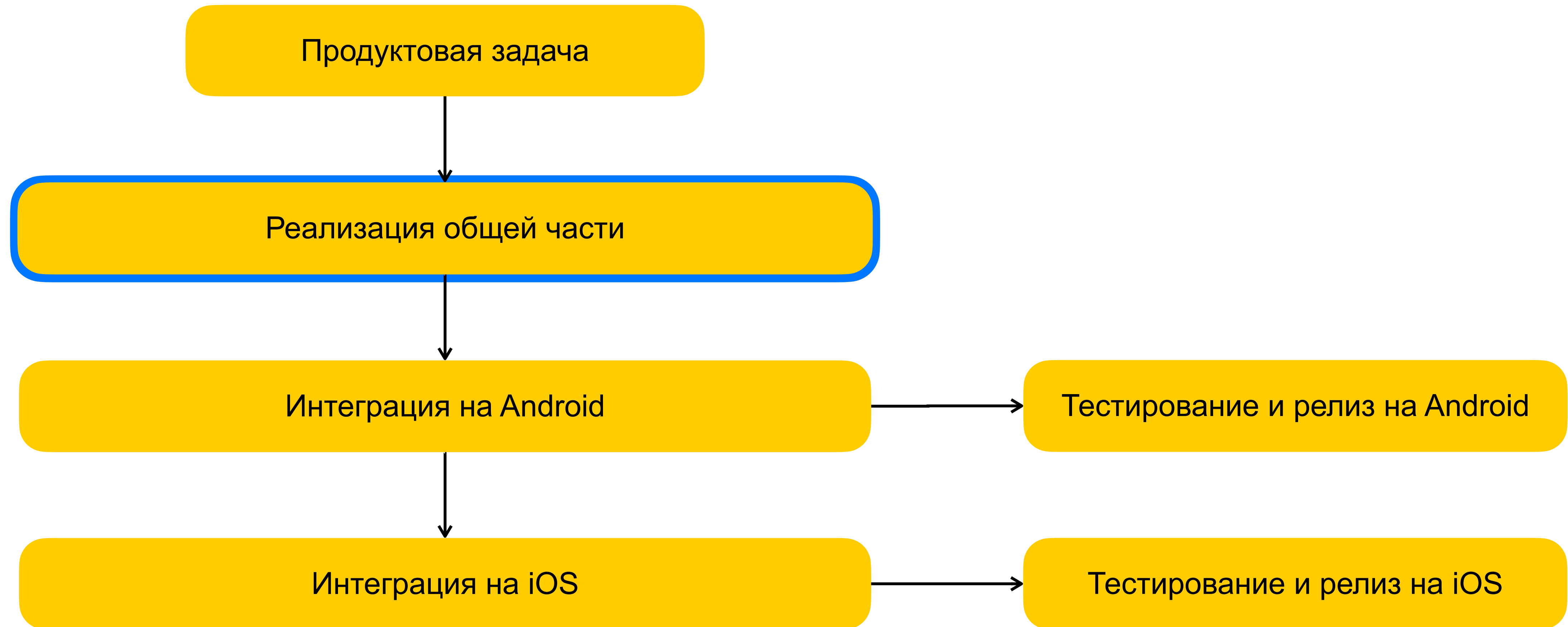
Простой способ построить процессы



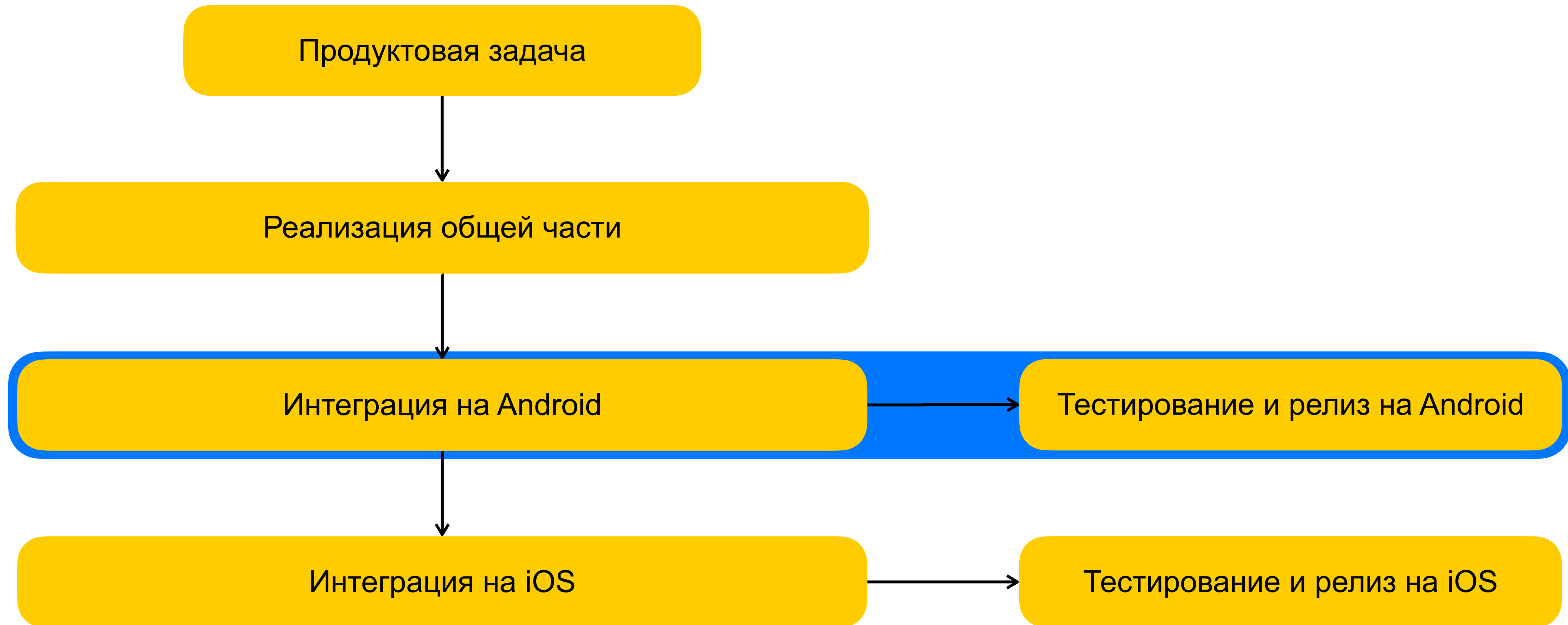
Простой способ построить процессы



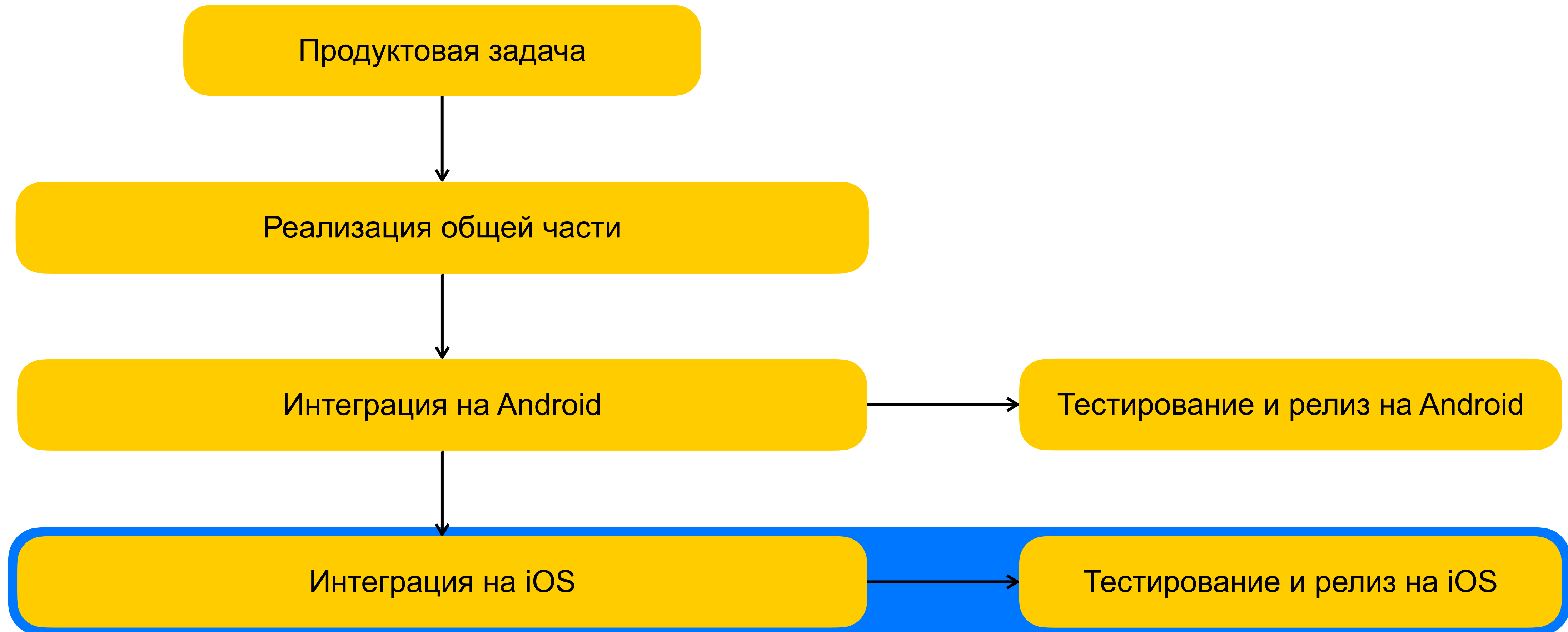
Простой способ построить процессы



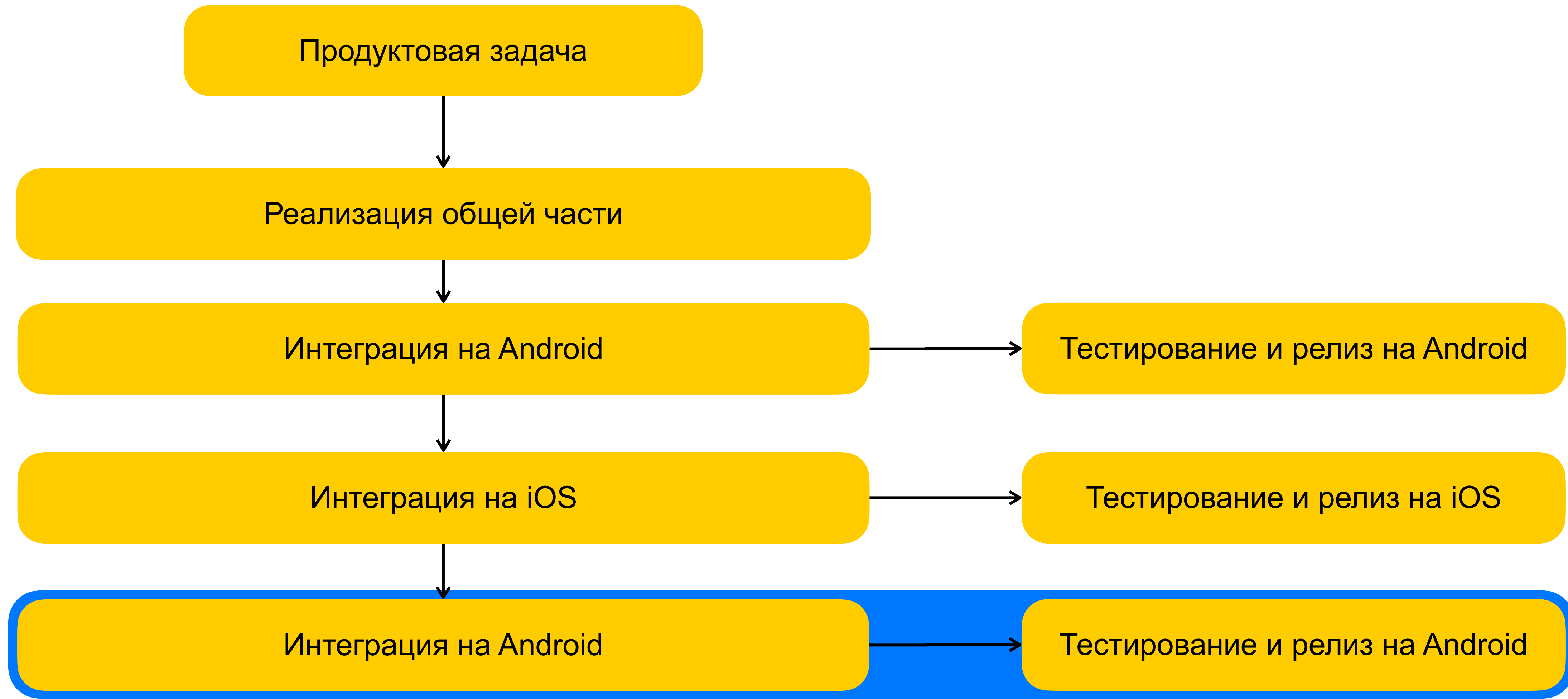
Простой способ построить процессы



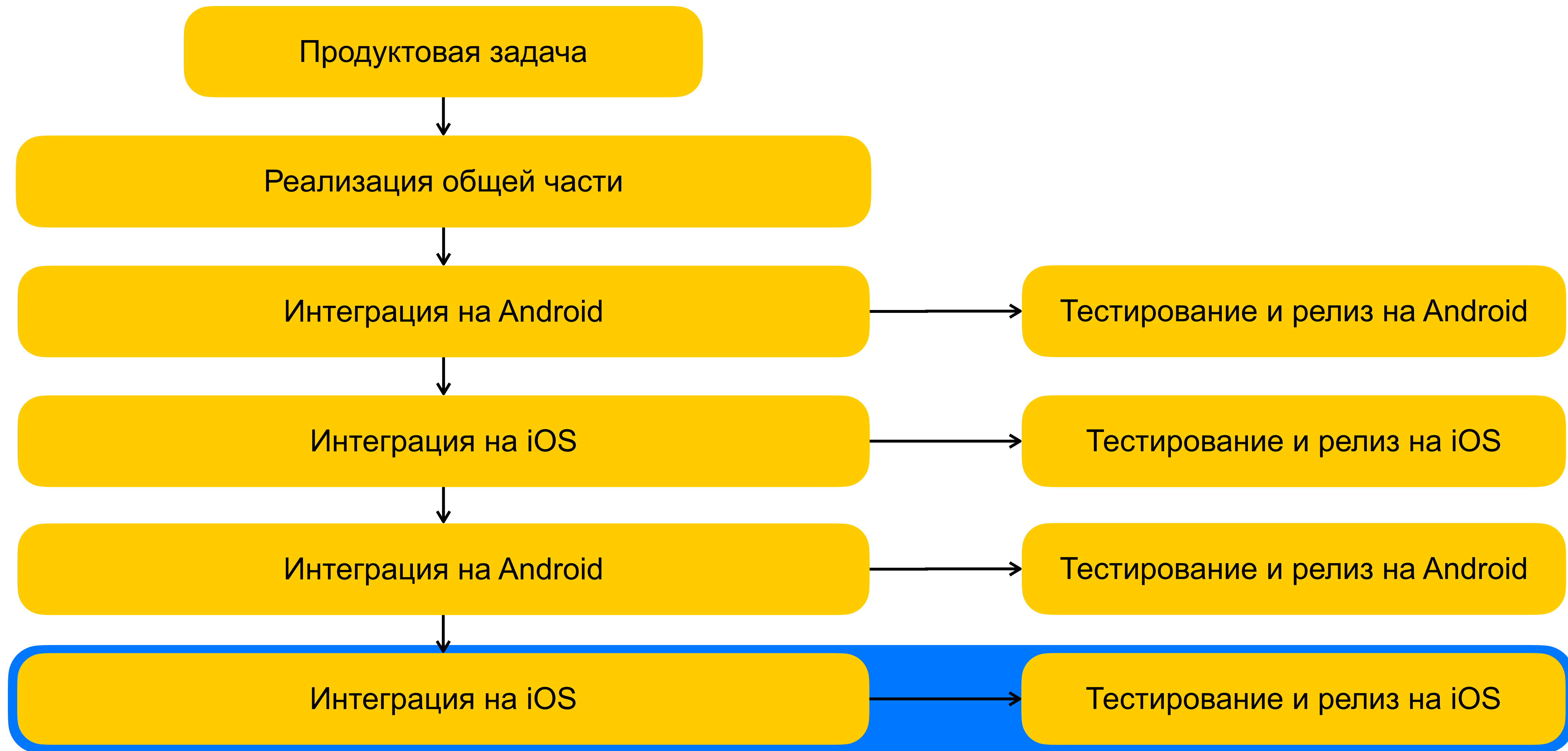
Простой способ построить процессы



Простой способ построить процессы



Простой способ построить процессы



Реализация проверки подписанных диплинков, когда мы только начинали

Реализация общей части	4 человеко-дня
Интеграция на Android	1 человеко-день
Интеграция на iOS	7 человеко-дней

Реализация проверки подписанных диплинков, когда мы только начинали

Реализация общей части	4 человеко-дня
Интеграция на Android	1 человеко-день
Интеграция на iOS	7 человеко-дней

Медленней в 2.5 раза

**Разрабатываем фичи
сразу под обе платформы**

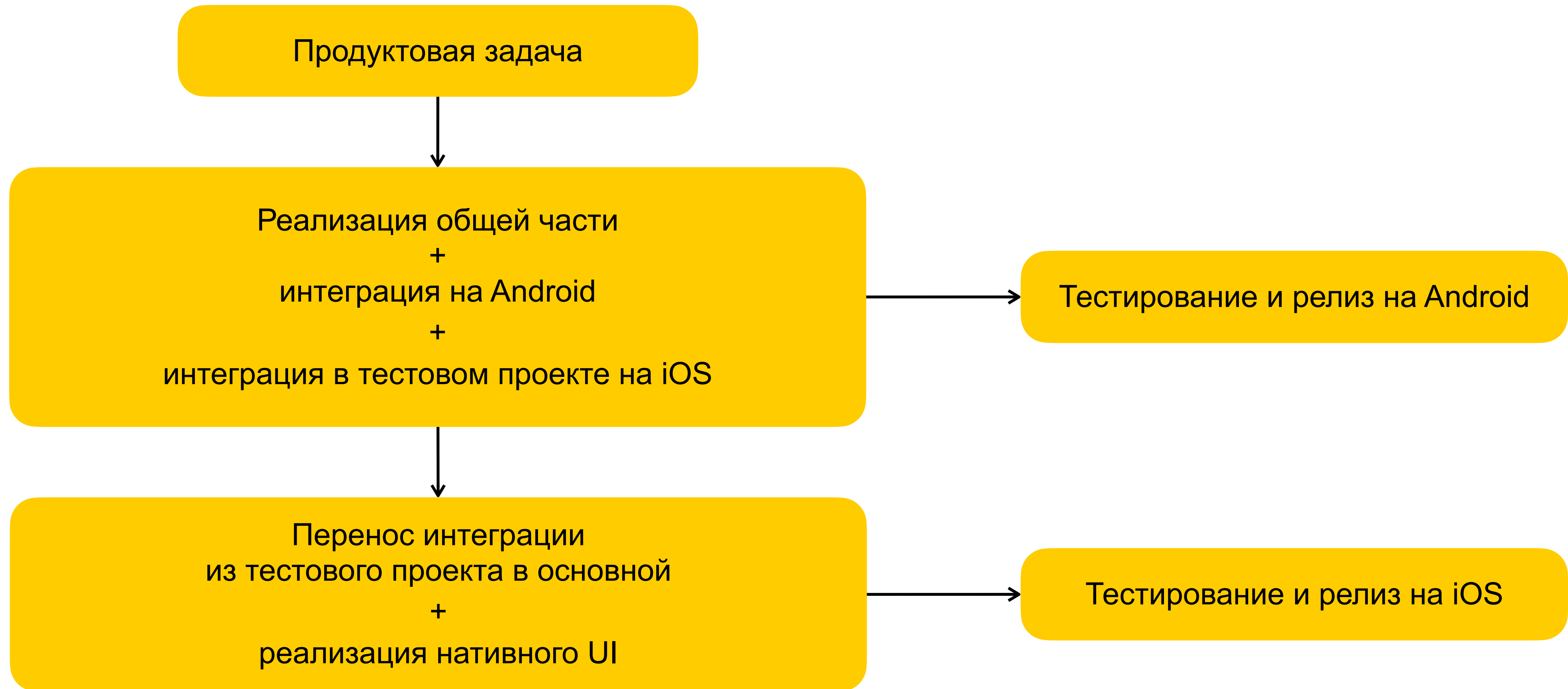
**Разрабатываем фичи
сразу под обе платформы**

**Значит нам нужны программисты,
которые хорошо владеют и iOS, и Android?**

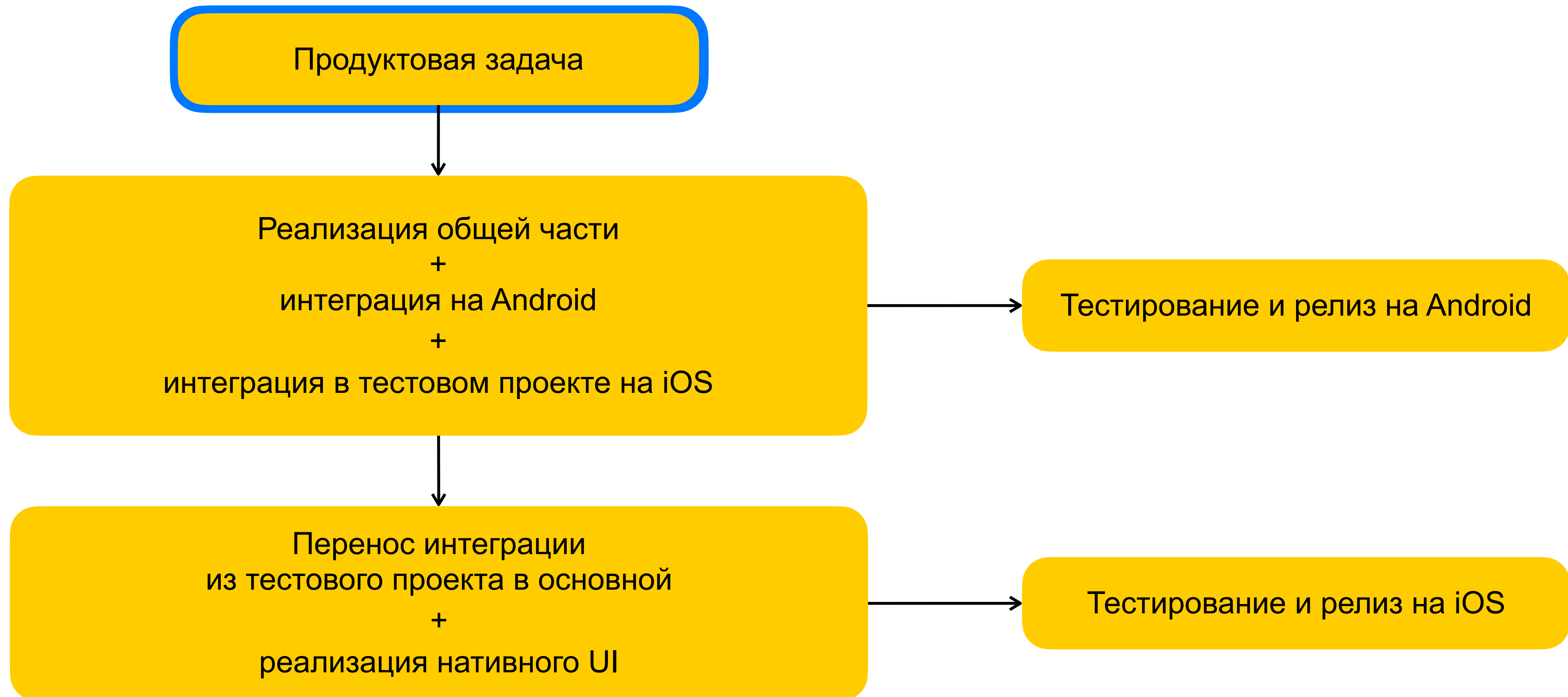
**Значит нам нужны программисты,
которые хорошо владеют и iOS, и Android?**

НЕТ

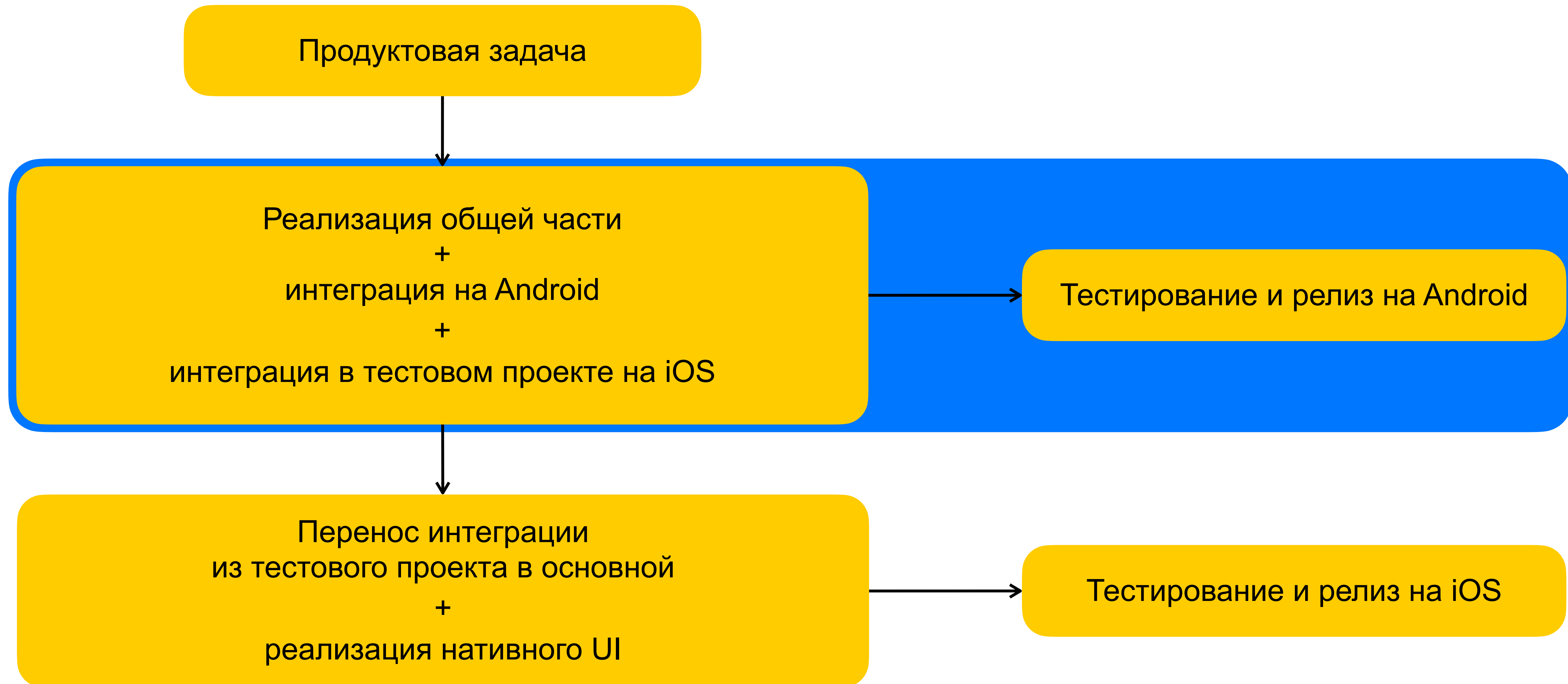
Как построить процессы



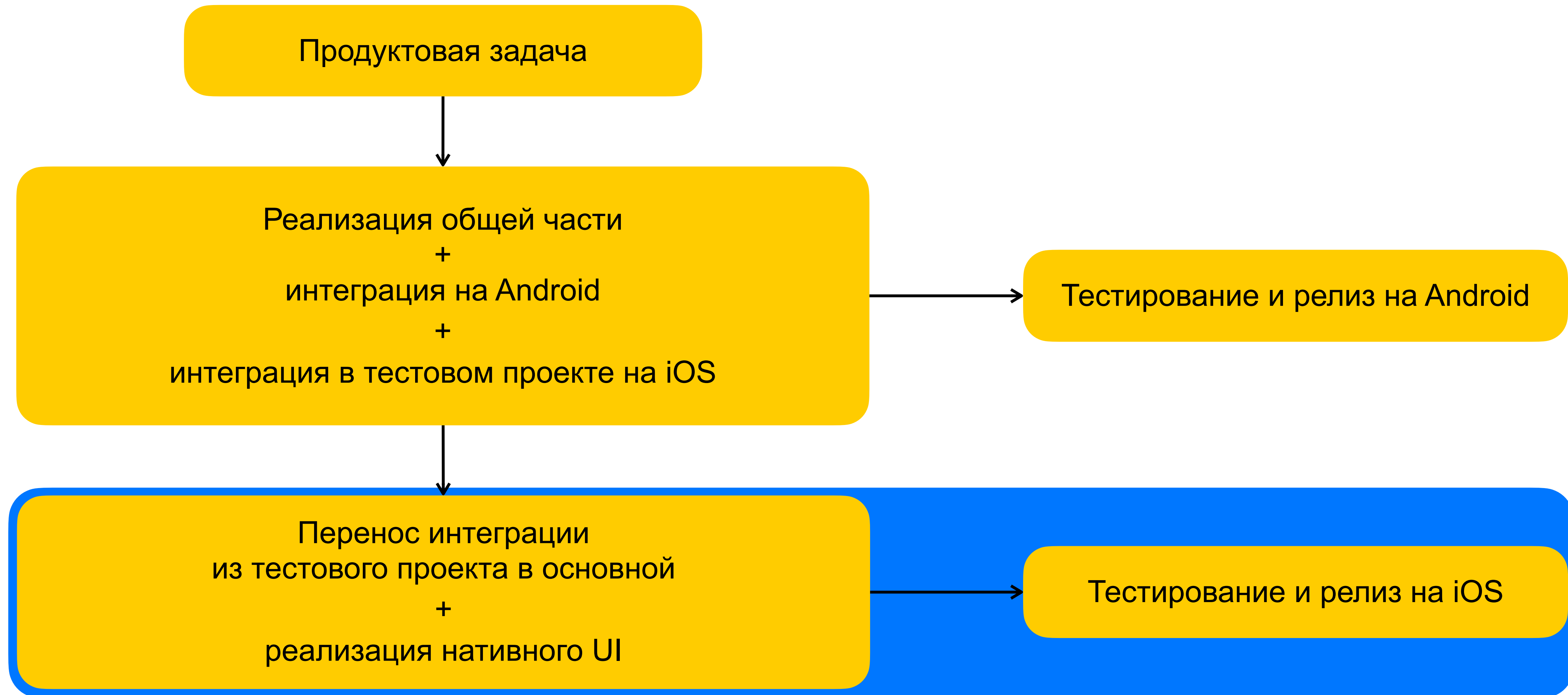
Как построить процессы



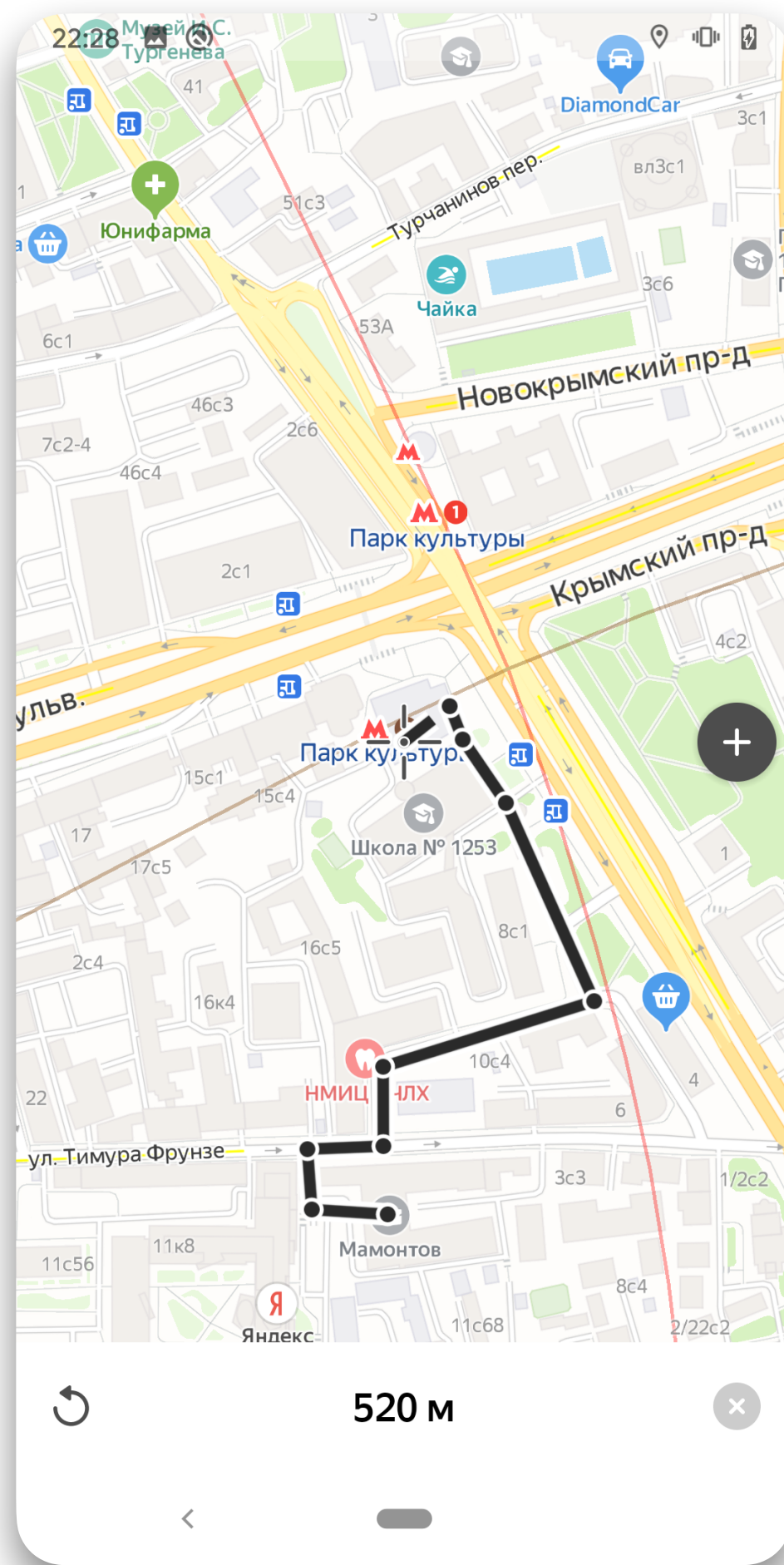
Как построить процессы



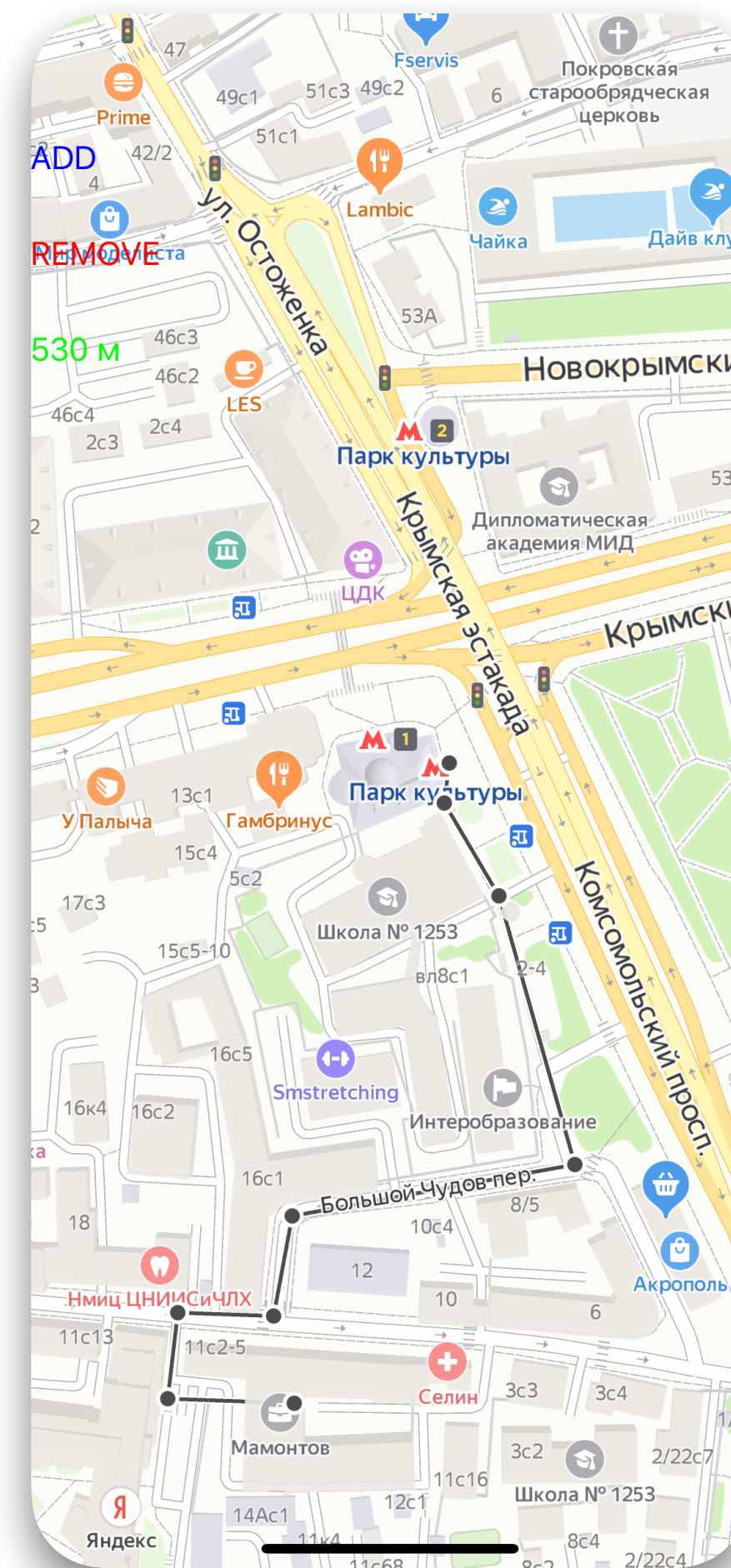
Как построить процессы



Тестовый проект “Рулетки”

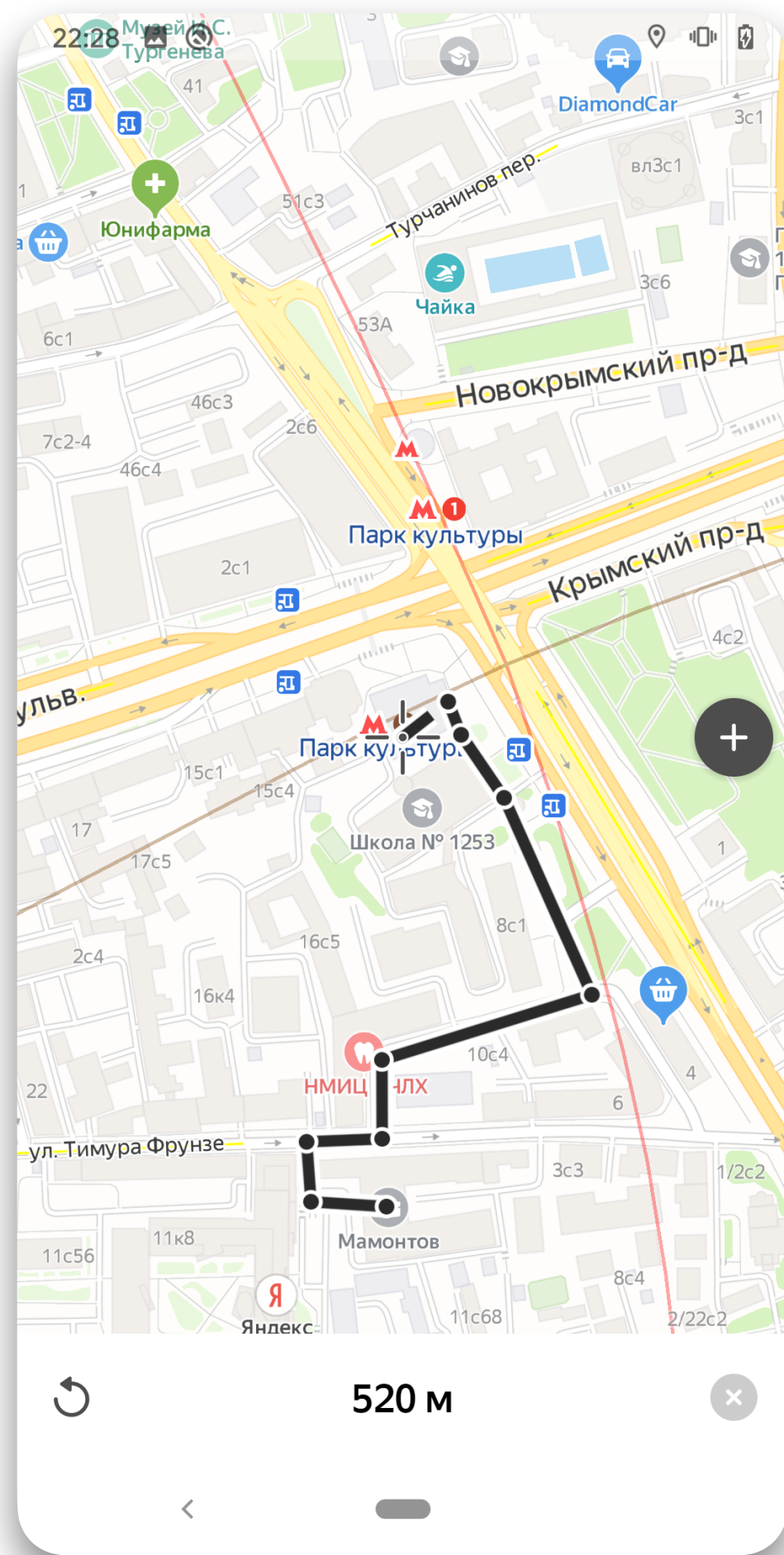


Android

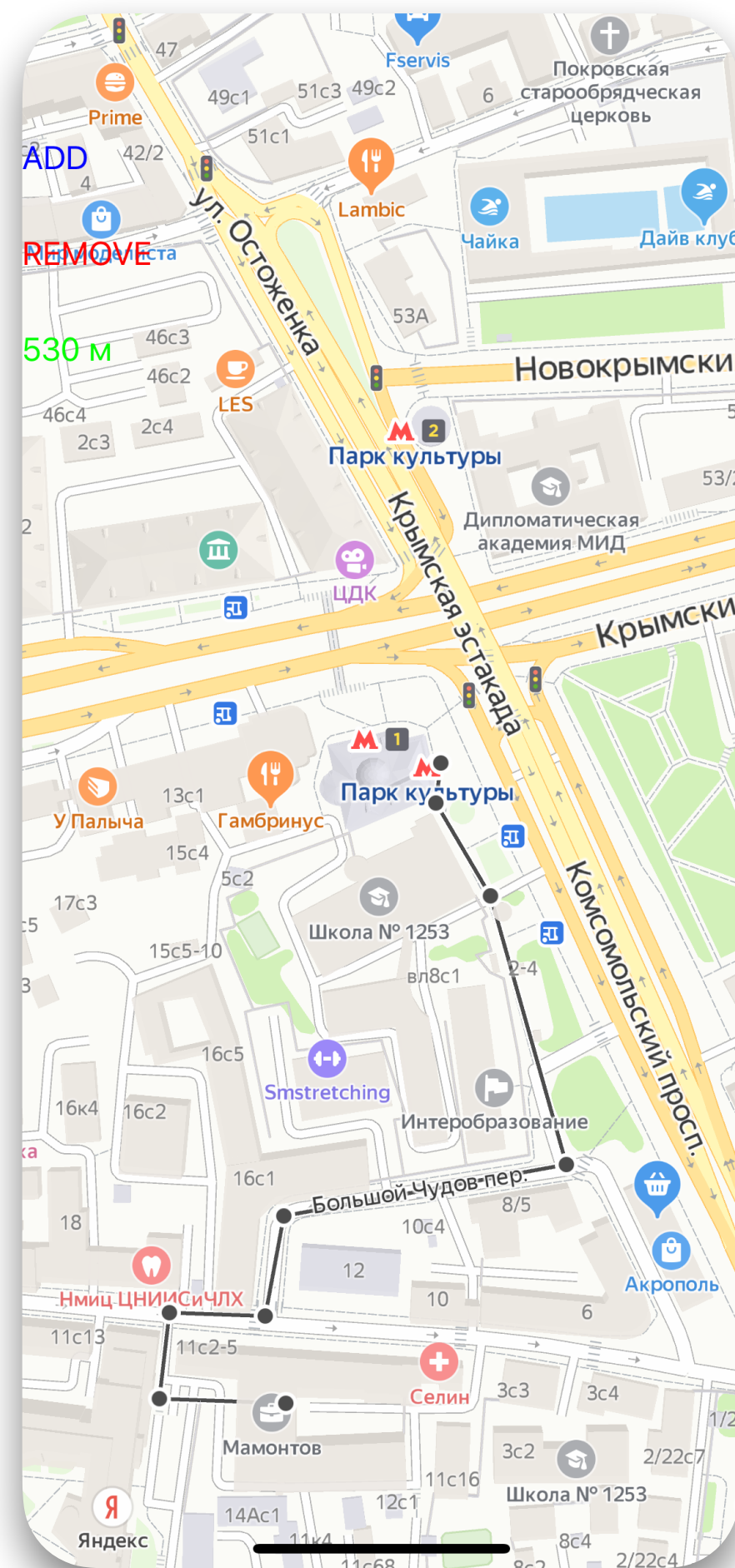


iOS Sample

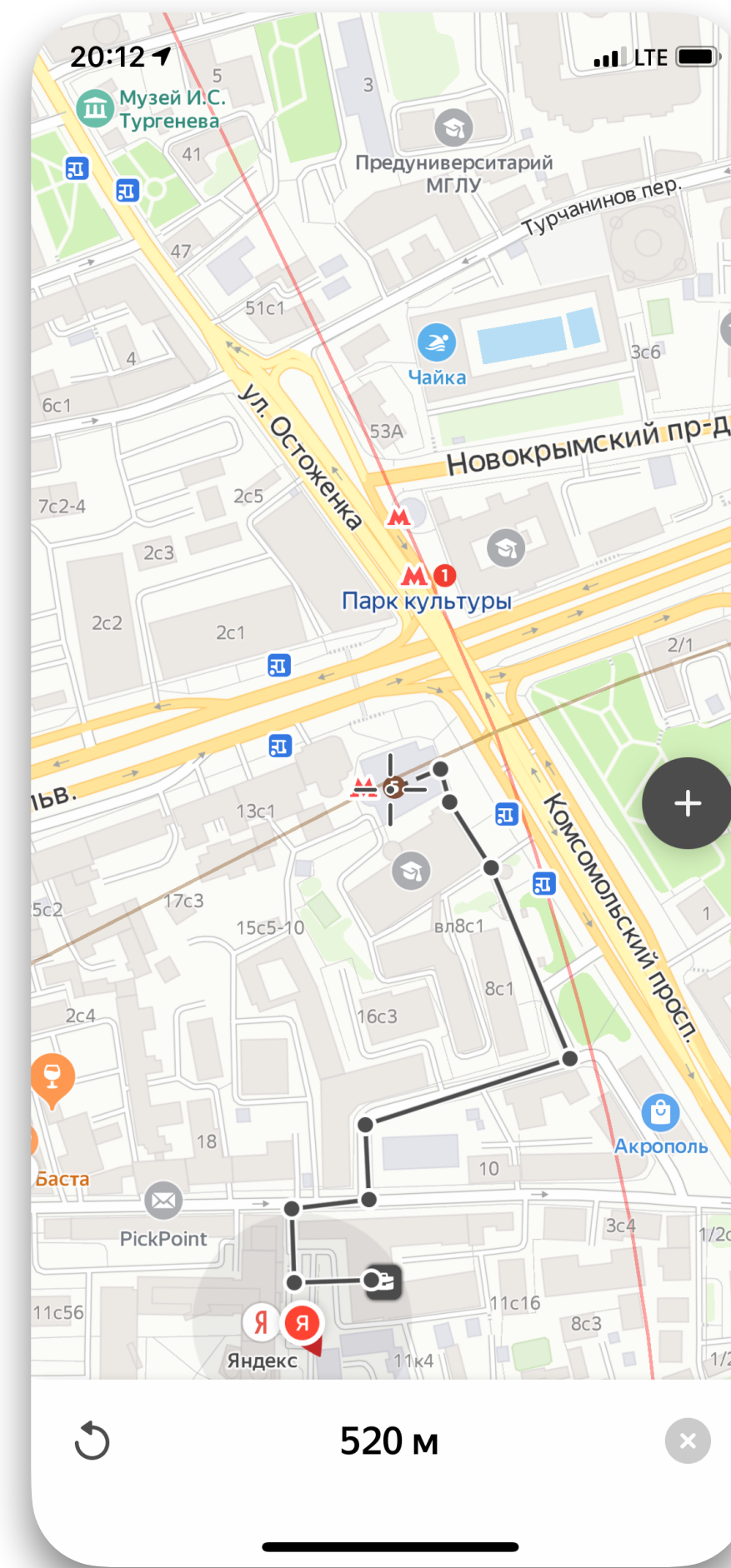
Тестовый проект “Рулетки”



Android



iOS Sample



iOS

Тестовый проект “Рулетки”

```
override fun viewDidLoad() {
    super.viewDidLoad()

    let addLabel = UILabel()
    addLabel.frame = CGRect(x: 0, y: 50, width: 200, height: 50)
    addLabel.text = "ADD"
    addLabel.addGestureRecognizer(UITapGestureRecognizer(target: self, action: #selector(addLandmark(sender:))))
    addLabel.textColor = UIColor.blue

    let removeLabel = UILabel()
    removeLabel.frame = CGRect(x: 0, y: 100, width: 200, height: 50)
    removeLabel.text = "REMOVE"
    removeLabel.addGestureRecognizer(UITapGestureRecognizer(target: self, action: #selector(removeLandmark(sender:))))
    removeLabel.textColor = UIColor.red

    let totalDistanceLabel = UILabel()
    totalDistanceLabel.frame = CGRect(x: 0, y: 150, width: 200, height: 50)
    totalDistanceLabel.textColor = UIColor.green

    view.addSubview(addLabel)
    view.addSubview(removeLabel)
    view.addSubview(totalDistanceLabel)

    rouletteInteractor.startRender()

    toRxSwift(rouletteInteractor.state())
        .bind(onNext: { [weak totalDistanceLabel] state in
            totalDistanceLabel?.text = state.formattedTotalDistance
        })
        .disposed(by: bag)
    }
}

@objc func addLandmark(sender: UITapGestureRecognizer) {
    rouletteInteractor.addLandmark()
}

@objc func removeLandmark(sender: UITapGestureRecognizer) {
    rouletteInteractor.removeLastLandmark()
}

private lazy var rouletteInteractor: KotlinNative.RouletteInteractor(...)
```

**Создаем тестовые проекты
одновременно под iOS и Android**

**Создаем тестовые проекты
одновременно под iOS и Android**

С помощью скрипта

Тестовые проекты

Настроены идентично основному проекту

Signing, BundleId, iOS target version, версия Swift, оптимизации...

Предоставляют необходимое окружение

YandexMapKit, AccountManager, базовая навигация

**Собираем тестовые проекты
и прогоняем тесты
при любом изменении в
мультиплатформенном коде**

- | | |
|--|-------------|
| ✓ Mobile / mobile-maps-client-android :: pull-requests-sb #65230 | 21 Oct 2020 |
| Tests passed: 573, ignored: 4 | ↺↻ |
| <hr/> | |
| ✓ Mobile / mobile-maps-client-ios / new / build-samples #29071 | 20 Oct 2020 |
| Success | ↺↻ |
| <hr/> | |
| ✓ Mobile / mobile-maps-client-ios / new / pull-requests #29074 | 21 Oct 2020 |
| Tests passed: 564 | ↺↻ |

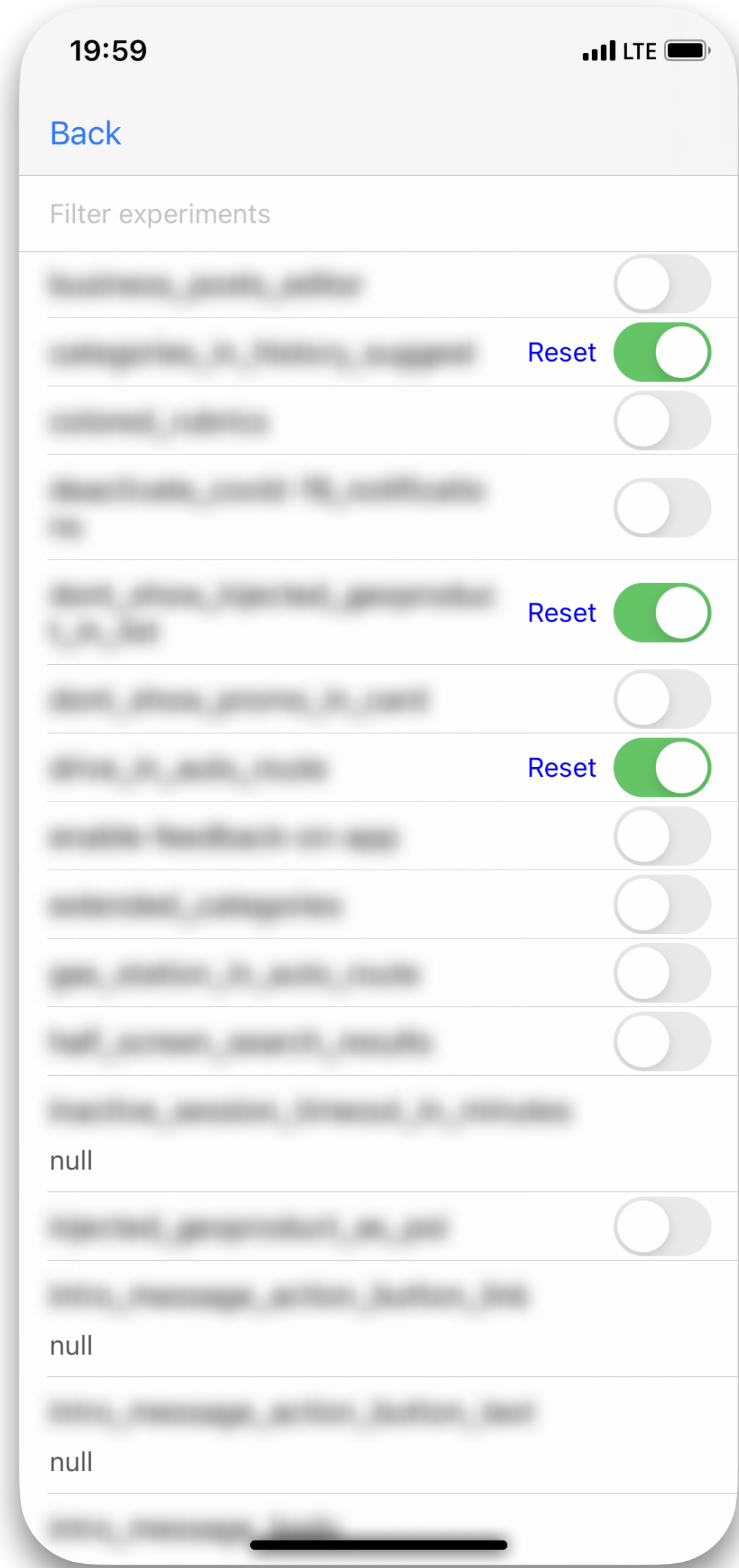
Реализация Дебаг-Панели

Реализация общей части 20 человеко-дней

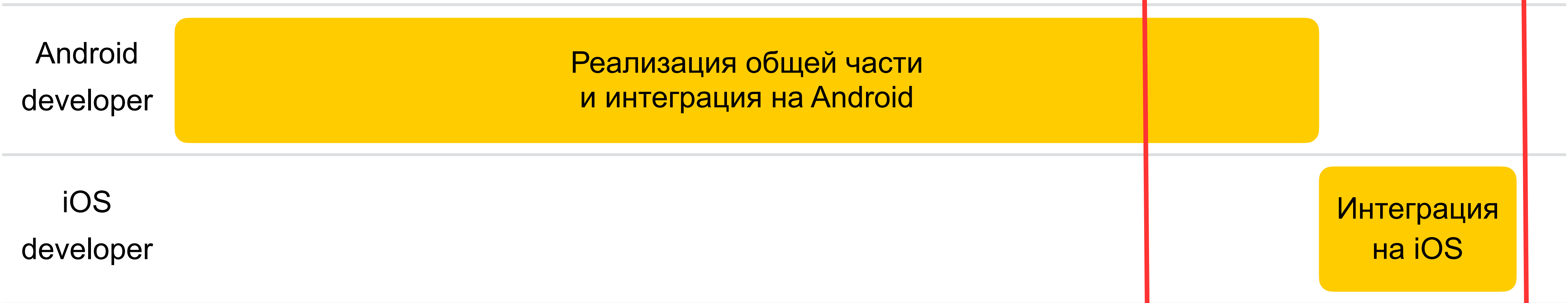
Реализация на Android 1 человеко-день

Интеграция на iOS 7 человеко-дня

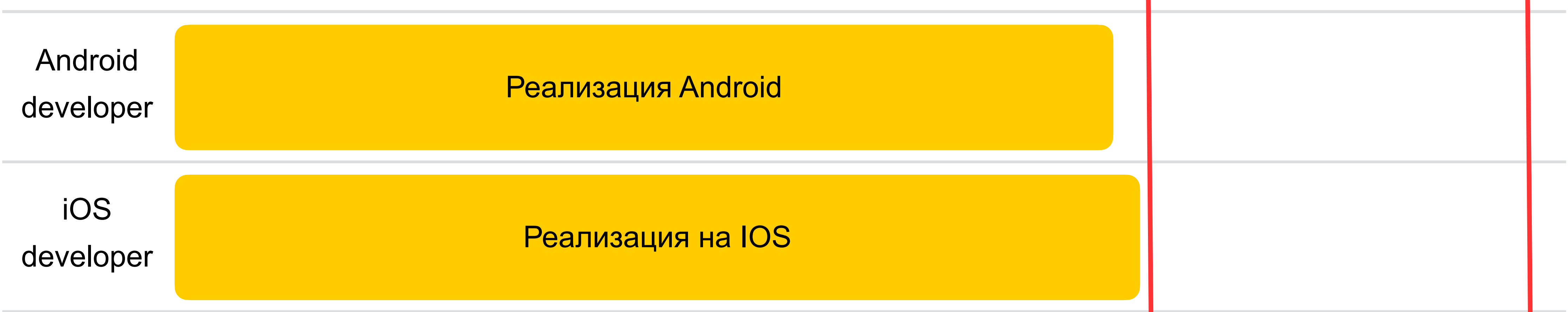
Быстрее на ~15%



Мультиплатформа



Натив



Мультиплатформа

Реализация общей части
и интеграция на Android

Интеграция
на iOS

Реализация Android

Реализация на IOS

Натив

Разрабатываем сразу под обе платформы

Разрабатываем сразу под обе платформы

Разрабатываем в тестовых проектах

Разрабатываем сразу под обе платформы

Разрабатываем в тестовых проектах

**Собираем прогоняем тесты
при любом изменении в мультиплатформе**

Результат применения Kotlin Multiplatform

Реализация проверки подписанных диплинков

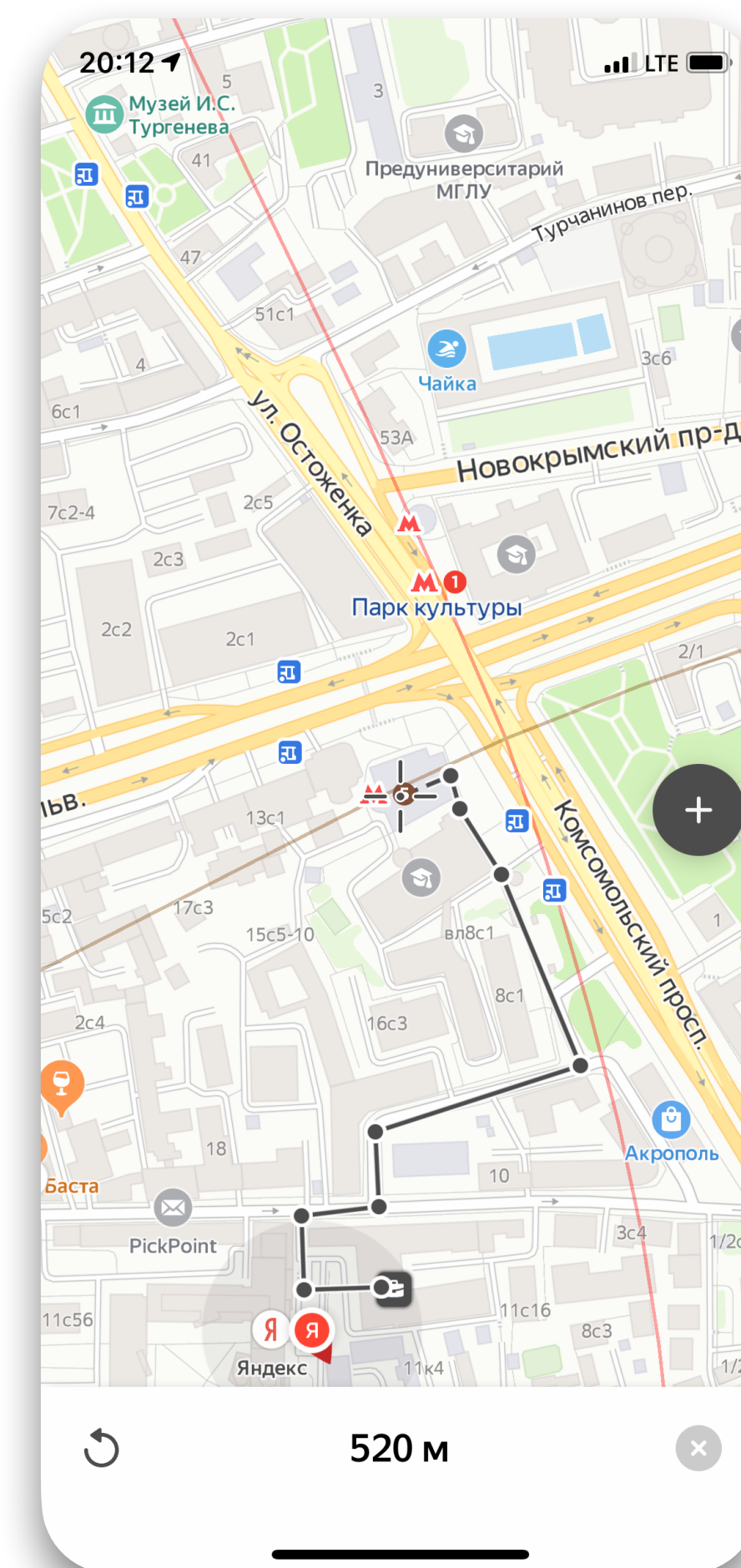
- › Мы не умели работать с мультиплатформой
- › Долго надстраивали инструменты
- › Много взаимодействия с платформенными библиотеками

Медленней на ~150%

Рулетка

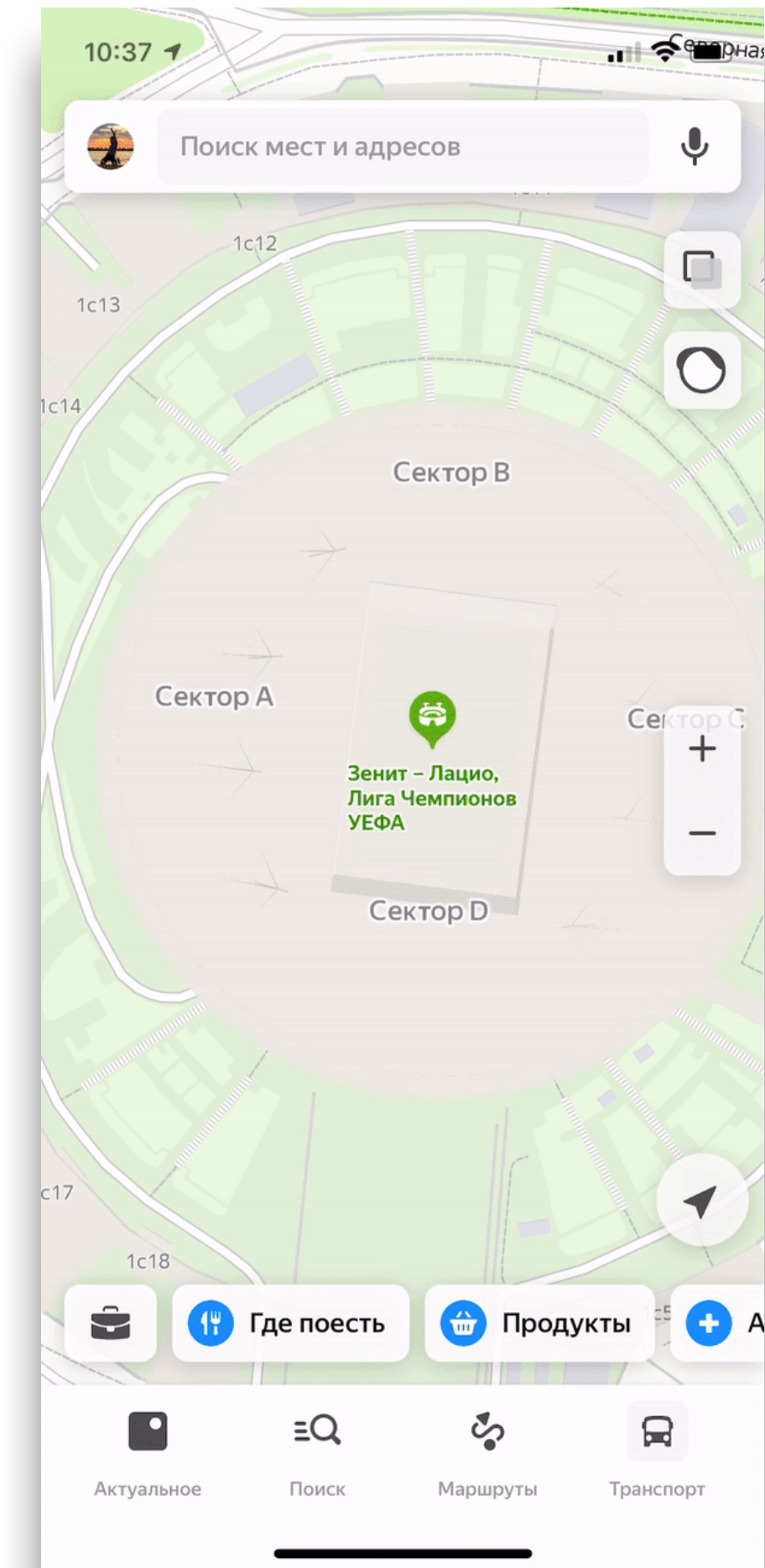
- › Починили процессы
- › Долго настраивали инструменты
- › Много взаимодействия с платформенными библиотеками

Медленней на ~20%



События на карте

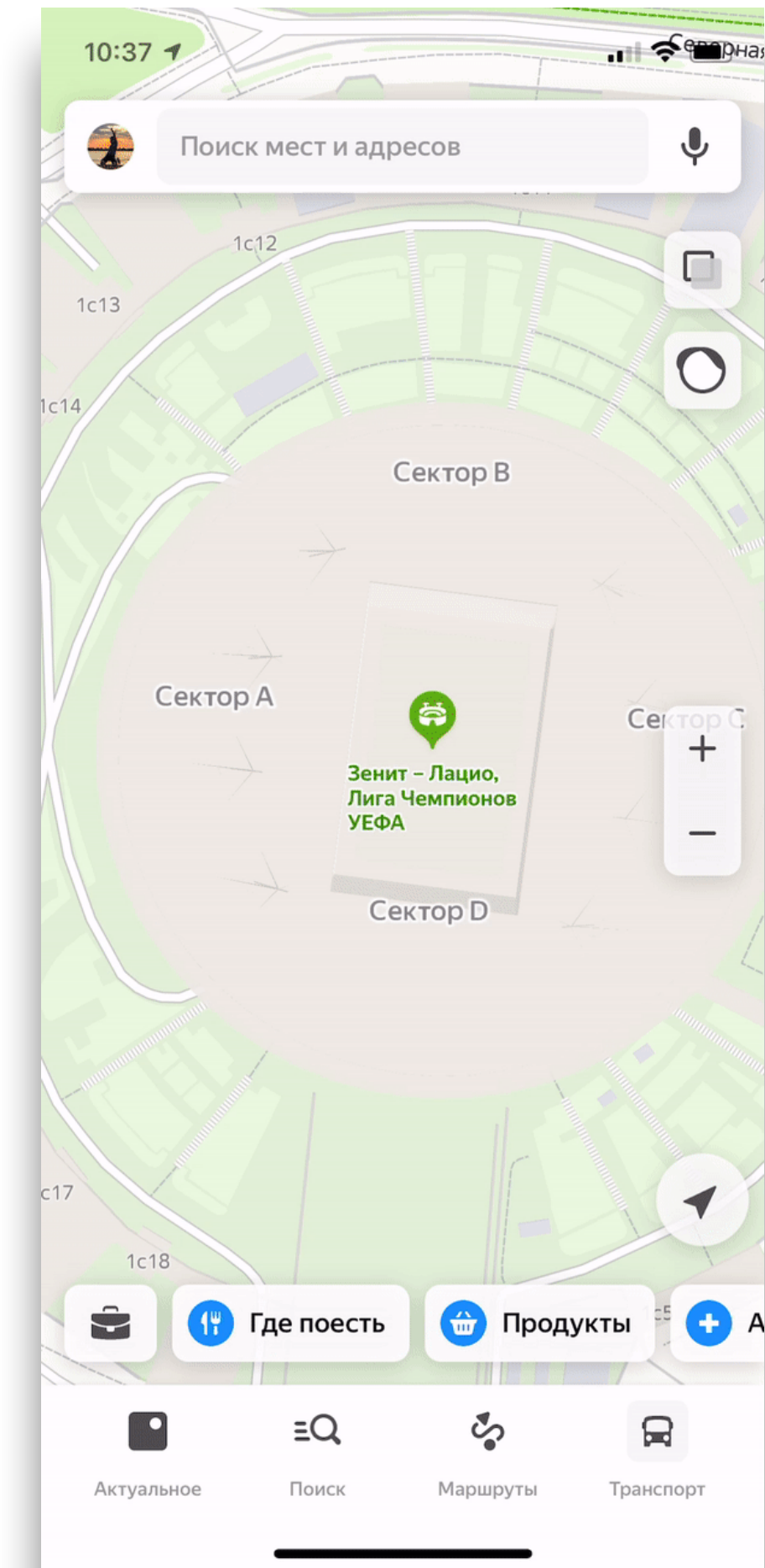
- › Научились быстро настраивать инструменты
- › Не было инфраструктуры для работы с сетью и кэшем



Так же, как и без Kotlin MPP

События на карте

- › Научились быстро настраивать инструменты
- › Не было инфраструктуры для работы с сетью и кэшем

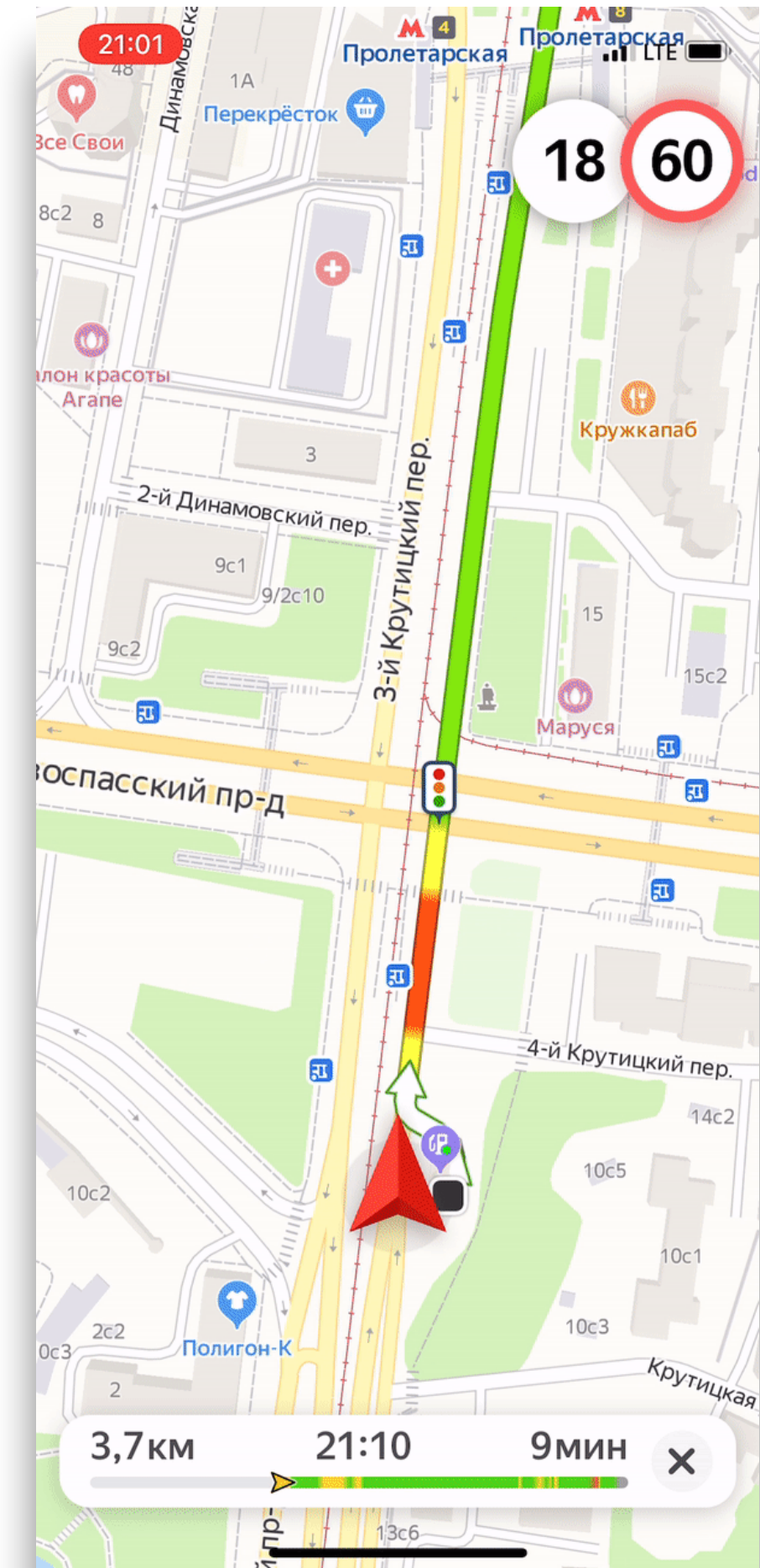


Так же, как и без Kotlin MPP

Автоматическое открытие карточки заправок

- › Мало взаимодействия с платформенными библиотеками
- › Много математики

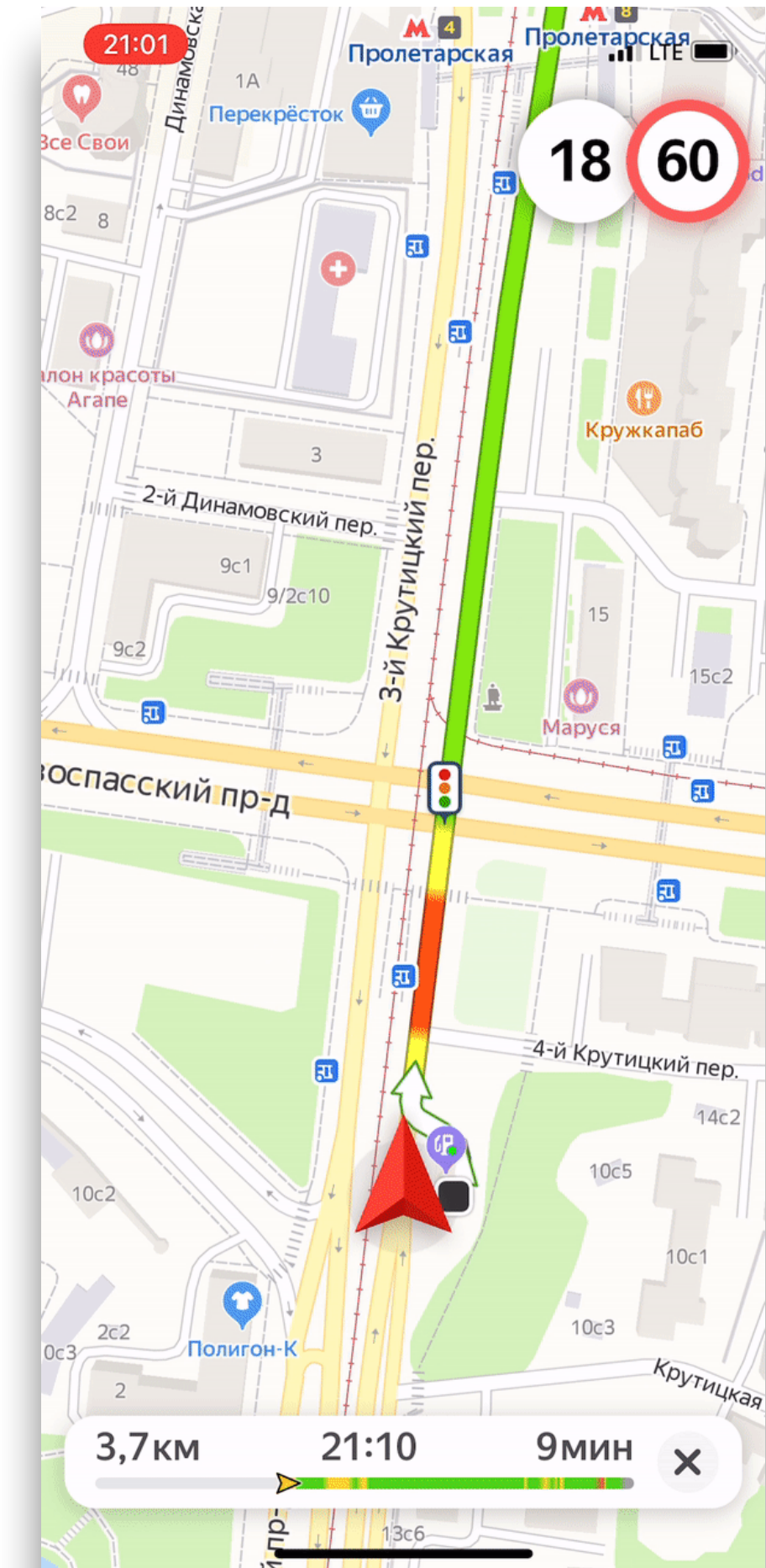
Быстрее на ~20%



Автоматическое открытие карточки заправок

- › Мало взаимодействия с платформенными библиотеками
- › Много математики

Быстрее на ~20%



Большой рефакторинг парсинга диплинков

- › Мало взаимодействия с нативными библиотеками
- › Много бизнес логики
- › Регулярно добавляются новые диплинки

Быстрее на ~25%

Сначала будет медленно

Сначала будет медленно

... очень медленно!

**Сначала будет медленно
... очень медленно!**

**С каждой новой задачей
мы получаем больший прирост в скорости**

**Мы ускорили разработку,
пробуйте и вы**

Яндекс Карты

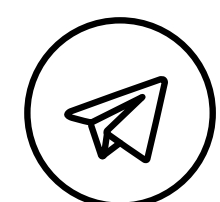
Спасибо

Михаил Куренков

iOS разработчик



<https://medium.com/yandex-maps-mobile>

 t.me/spolispastom