



# Тулкиты кодогенерации Android vs iOS

Анна Жаркова  
Lead Mobile developer

# Обо мне



- В мобильной разработке с 2013
- Ведущий мобильный разработчик в Usetech
- Нативная разработка под iOS и Android (Swift/Objective-C, Kotlin/Java) кросс-платформа (Xamarin, Kotlin multiplatform)
- Ментор, управляю командой направления
- Спикер на конференциях AppsConf, Mobius, TechTrain, DroidCon (2022)
- Преподаватель в Otus (iOS Pro и базовый)
- Автор статей по мобильной разработке (SwiftUI, iOS, KMM)
- Эксперт Skillbox

# Обсудим:

- Для чего нам это все?
- Виды и концепции инструментов
- Решения для новых модулей. Template
- Решения для точечных задач. Plugins, Source Editor Extensions, SPM
- Компиляторный мир
- AI для кодогенерации

# Инструменты кодогенерации. Назначение

- Преобразование существующего кода
- Генерация нового кода
- Решение однотипных задач
- Упрощение и ускорение разработки
- Сокращение написания кода разработчиком



# Инструменты кодогенерации. Назначение

- Преобразование существующего кода
- Генерация нового кода
- Решение однотипных задач
- Упрощение и ускорение разработки
- Сокращение написания кода разработчиком



# Зачем мне знать и уметь это

- Есть же много готовых решений
- AI



# Зачем мне знать и уметь это

- Писать собственные решения под свои конкретные задачи и проекты
- Уметь адаптировать существующее решение
- Правильно выбирать кейс работы
- Повысить свой уровень разработчика



# Типы инструментов

- Изменения здесь и сейчас
- Build-time/ compile-time

# Типы инструментов

- Код пересоздается при генерации в compile-time/build-time
- Созданный код можно изменять

# Типы инструментов

- Байт-код
- Понятные исходники

# Основная задача

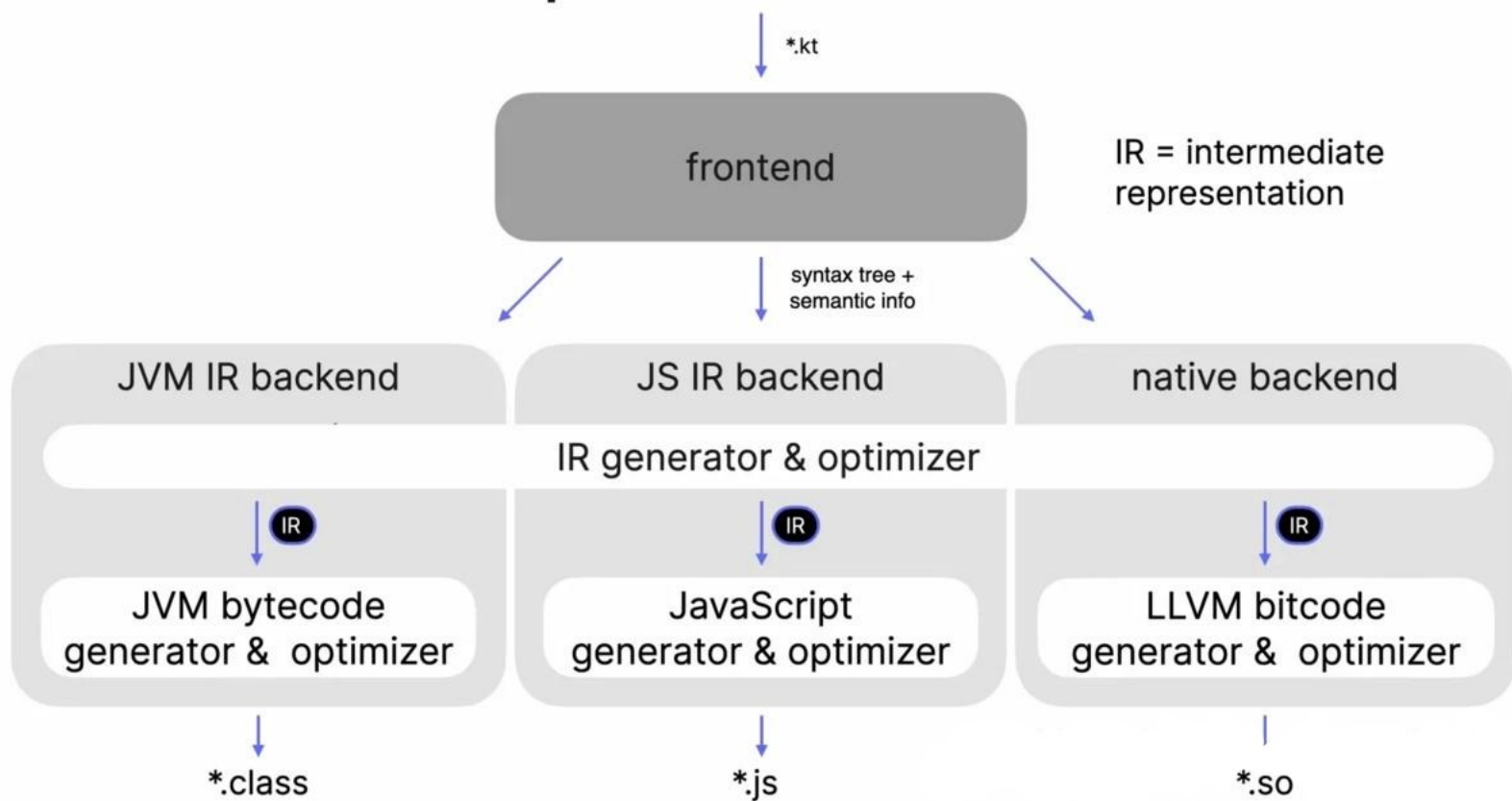
Мы пишем какой-то текст, который будет восприниматься, как код, и компилироваться, как код



**Чтобы знать кодогенерацию, надо  
понимать компиляцию**

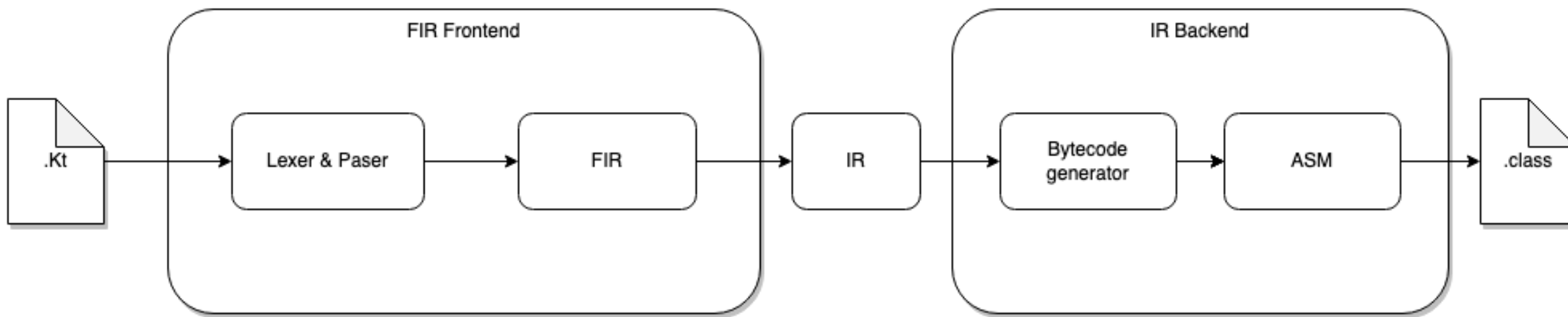


# Под капотом компиляции

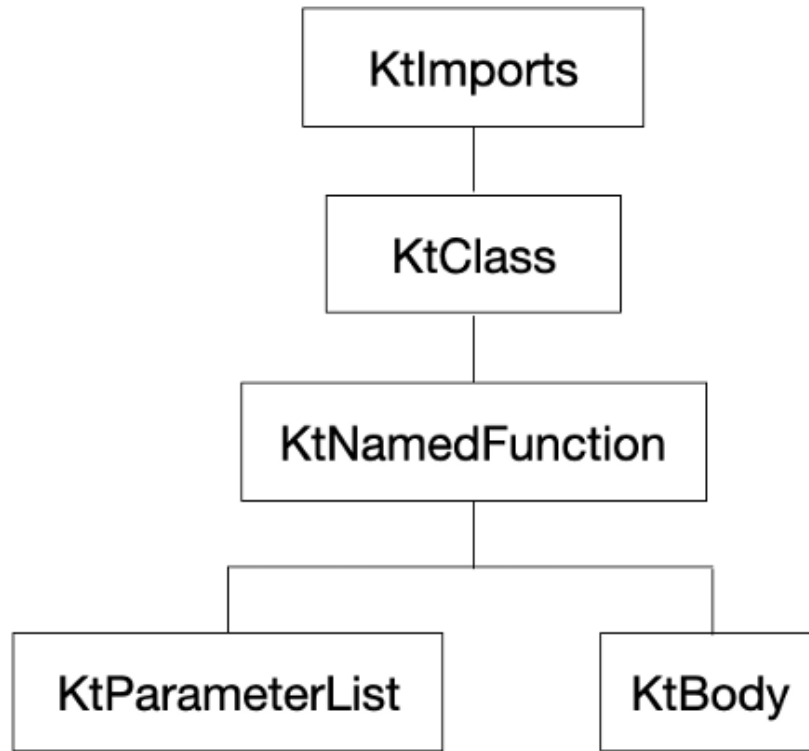


# Компиляция FIR&IR.K2

Kotlin 2.0, K2



# Programming Structure Interface (PSI)



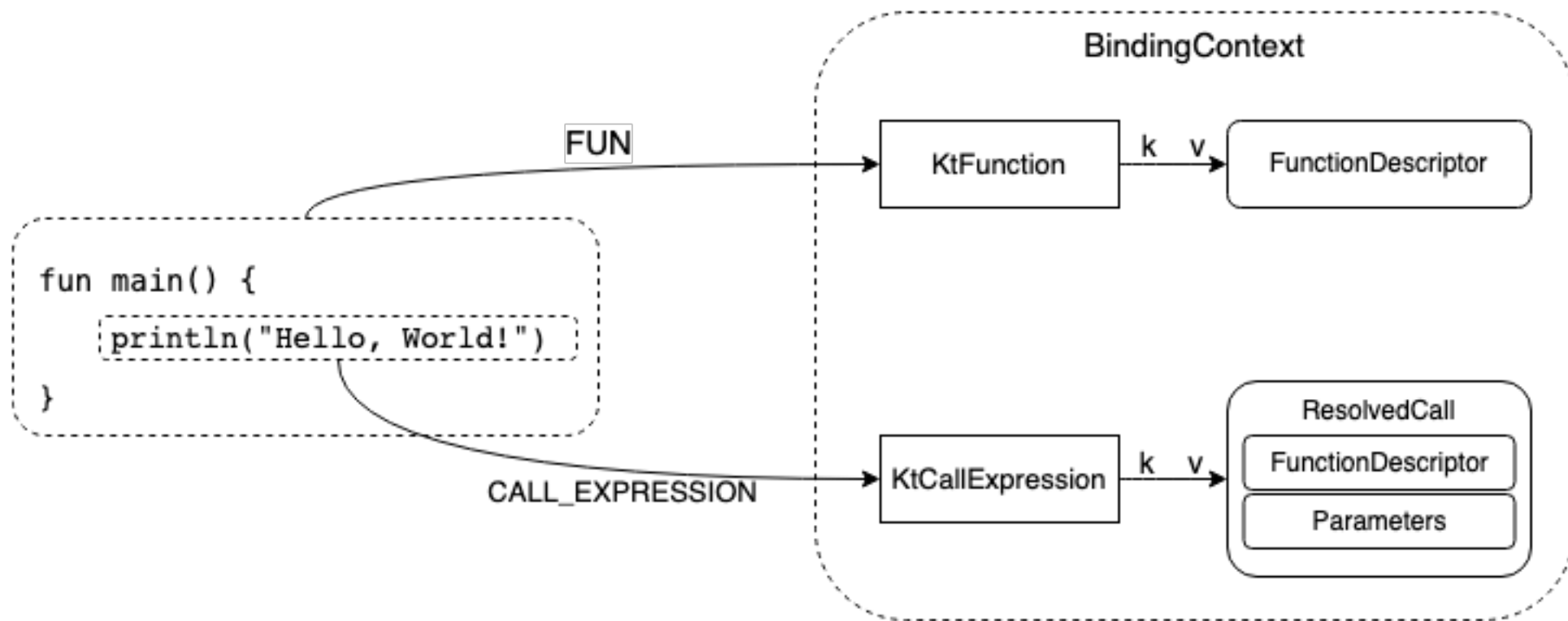
# IR, пример кода

```
//Код
suspend fun test() {
    println(returnsInt())
}

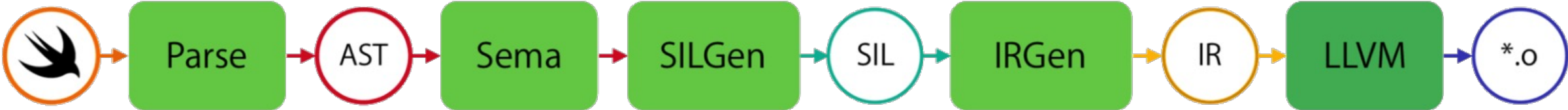
// Kotlin IR
FUN name:test visibility:public modality:FINAL <> () returnType:kotlin.Unit [suspend]
BLOCK_BODY
    CALL 'public final fun println (message: kotlin.Int): kotlin.Unit [inline]
        declared in kotlin.io.ConsoleKt' type=kotlin.Unit origin=null
    message: CALL 'public final fun returnsInt (): kotlin.Int [suspend]
        declared in <root>.TestKt' type=kotlin.Int origin=null
```

# IR обработка AST

Элементы PSI - > дескрипторы

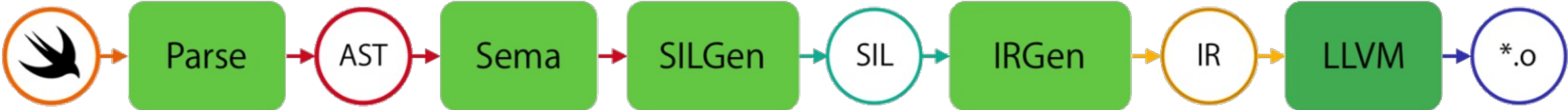


# Swift. Компиляция



# Swift. Компиляция

Swift Intermediate Language










# SwiftSyntax

The image shows a browser window displaying the SwiftSyntax documentation. The browser's address bar shows the URL `swift-syntax > SwiftSyntax > Introduction to SwiftSyntax`. The page title is "Introduction to SwiftSyntax". Below the title, a subtitle reads: "SwiftSyntax provides the foundation for tools that inspect, manipulate, and transform Swift source code." A clock icon indicates a "15min Estimated Time" to read. A blue "Get started" button is prominently displayed. Below this, a card for "Chapter 1 SwiftSyntax By Example" is shown, featuring a Swift logo icon and a brief description: "Explore the SwiftSyntax API by building a tool that sorts imports in a Swift file." A "15min" time indicator is also present for this chapter. On the left side of the browser window, a sidebar navigation menu is visible, listing various sections under "Workspace Documentation" and "Articles". The "Introduction to SwiftSyntax" item is highlighted in blue. The sidebar also includes sections for "Contributing", "Syntax", "SyntaxCollection", "Trivia", and "TriviaPiece". At the bottom of the sidebar, there is a "Filter" input field.



# Swift AST

Swift AST Explorer     509.0.0  Structure  Lookup  Statistics

```
1 import Foundation
2
3 @SwiftRetrofit
4 public protocol NewsService {
5
6     @Get(path: "everything")
7     func loadNews(query: QueryParam<String>) async throws -> NewsList
8 }
9
```

```
▼ SourceFile
  ▼ CodeBlockItemList
    ▼ CodeBlockItem
      ▼ ImportDecl
        ▼ AttributeList
        ▼ DeclModifierList
          import
        ▼ ImportPathComponentList
          ▼ ImportPathComponent
            Foundation
      ▼ CodeBlockItem
        ▼ ProtocolDecl
          ▼ AttributeList
            ▼ Attribute
              @
            ▼ IdentifierType
              SwiftRetrofit
          ▼ DeclModifierList
            ▼ DeclModifier
              public
          protocol
          NewsService
        ▼ MemberBlock
          {
            ▼ MemberBlockItemList
```



# PSI Viewer

The screenshot displays the IntelliJ PSI Viewer interface. On the left, a tree view shows the structure of a Kotlin file. The selected element is a `PsiWhiteSpace` node, which is highlighted in blue. The tree view includes the following elements:

- ψ PsiFile: SecondScreen.kt
  - > ψ PACKAGE\_DIRECTIVE
  - ─ PsiWhiteSpace
  - ψ IMPORT\_LIST
    - ψ IMPORT\_DIRECTIVE
      - ψ PsiElement(import)
        - ─ PsiWhiteSpace
      - > ψ DOT\_QUALIFIED\_EXPRESSION
        - ─ PsiWhiteSpace
      - > ψ IMPORT\_DIRECTIVE
        - ─ PsiWhiteSpace
      - > ψ IMPORT\_DIRECTIVE
        - ─ PsiWhiteSpace
    - > ψ FUN
      - ─ PsiWhiteSpace
    - ψ FUN
      - > ψ MODIFIER\_LIST
        - ─ PsiWhiteSpace
      - ψ PsiElement(fun)
        - ─ PsiWhiteSpace
      - ψ PsiElement(IDENTIFIER)
    - > ψ VALUE\_PARAMETER\_LIST
      - ─ PsiWhiteSpace
    - > ψ BLOCK

Property	Value
acquire	{PsiSymbolReferenceService.OWN_REFERENCES=PsiParameterizedCachedValue{com.intellij.model....
cachedLength	2
chars	
children	[]
class	class com.intellij.psi.impl.source.tree.PsiWhiteSpaceImpl
containingFile	KtFile: SecondScreen.kt
context	KtFile: SecondScreen.kt
elementType	WHITE_SPACE
firstChild	null
firstChildNode	null
language	Language: kotlin
lastChild	null
lastChildNode	null
manager	{}
name	null
navigationElement	PsiWhiteSpace
nextSibling	IMPORT_LIST
node	PsiWhiteSpace
opaque	{PsiSymbolReferenceService.OWN_REFERENCES=PsiParameterizedCachedValue{com.intellij.model....
originalElement	PsiWhiteSpace
parent	KtFile: SecondScreen.kt

# Уровни погружения:

- Работа с IR-представлением
- Работа с PSI-структурой проекта
- Работа с PSI-структурой файла
- Работа с текстом файла

# Посмотрим на практике

# Что оптимизировать можно

Однотипные задачи:

- Генерация View
- Генерация ресурсов
- Навигация
- Бизнес-логика (сеть, хранилища)
- Тесты
- Архитектура, Dependency Injection...

# Что оптимизировать можно

Однотипные задачи:

- Форматирование = изменение существующего кода
- Генерация View из Json = создания нового кода
- Генерация новых архитектурных компонентов = создание нового кода по шаблону

# Как будем сравнивать

Баллы за гибкость, красоту и функциональность



0



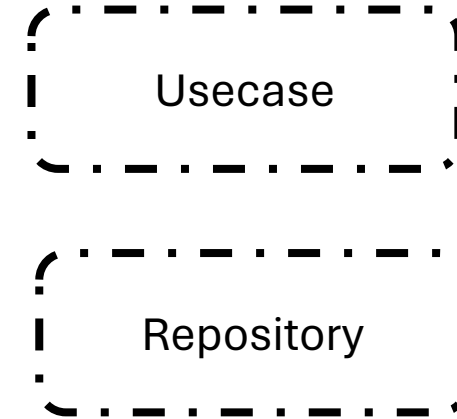
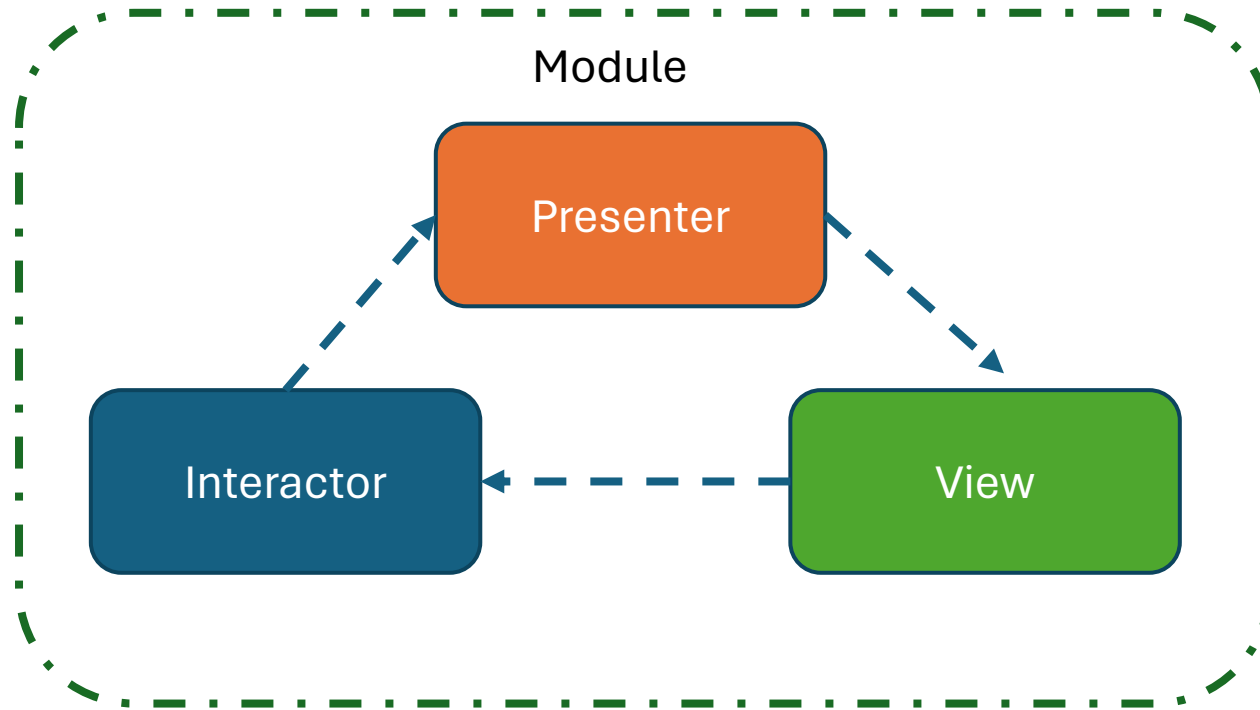
0

**Начнем с простого**

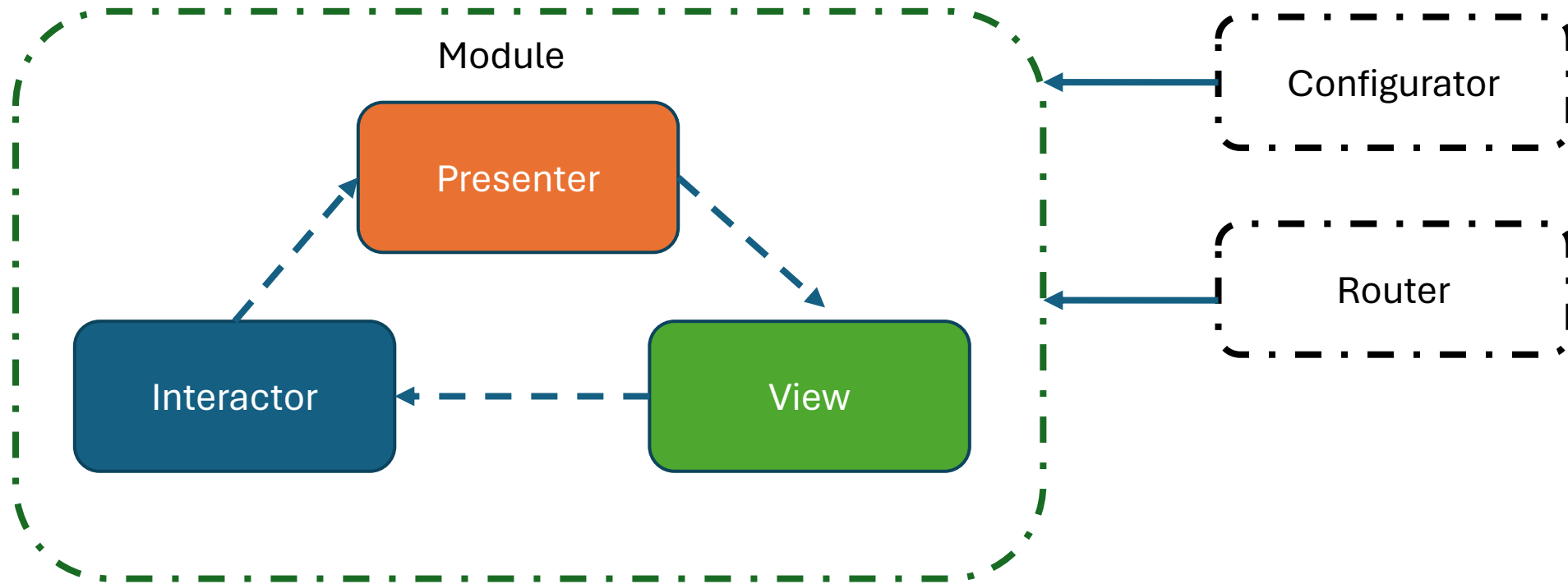
**Новый модуль/экран**



# Новый модуль архитектуры. Android



# Новый модуль архитектуры. iOS

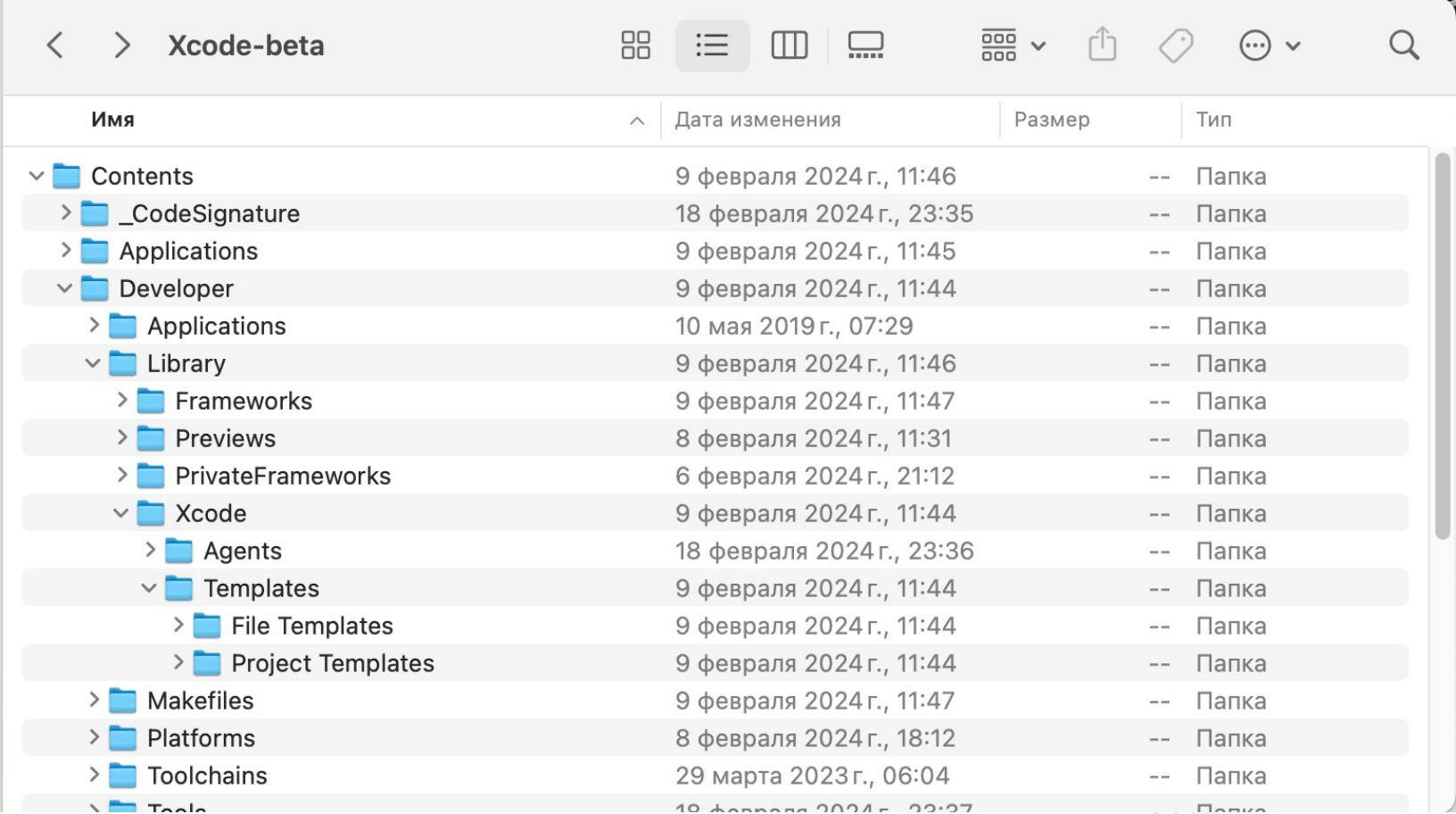


# Способы решения

- Xcode Template
- Android Studio Wizard template

# Xcode Template

Xcode.app -> Templates



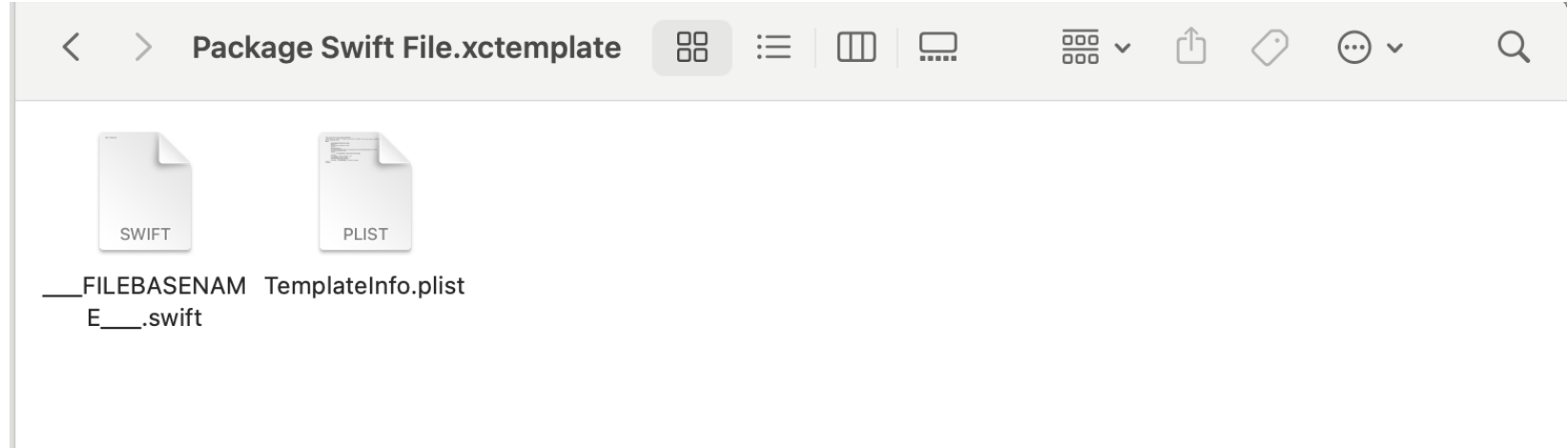
Имя	Дата изменения	Размер	Тип
Contents	9 февраля 2024 г., 11:46	--	Папка
_CodeSignature	18 февраля 2024 г., 23:35	--	Папка
Applications	9 февраля 2024 г., 11:45	--	Папка
Developer	9 февраля 2024 г., 11:44	--	Папка
Applications	10 мая 2019 г., 07:29	--	Папка
Library	9 февраля 2024 г., 11:46	--	Папка
Frameworks	9 февраля 2024 г., 11:47	--	Папка
Previews	8 февраля 2024 г., 11:31	--	Папка
PrivateFrameworks	6 февраля 2024 г., 21:12	--	Папка
Xcode	9 февраля 2024 г., 11:44	--	Папка
Agents	18 февраля 2024 г., 23:36	--	Папка
Templates	9 февраля 2024 г., 11:44	--	Папка
File Templates	9 февраля 2024 г., 11:44	--	Папка
Project Templates	9 февраля 2024 г., 11:44	--	Папка
Makefiles	9 февраля 2024 г., 11:47	--	Папка
Platforms	8 февраля 2024 г., 18:12	--	Папка
Toolchains	29 марта 2023 г., 06:04	--	Папка
Tools	18 февраля 2024 г., 23:37	--	Папка

# Xcode Template

Source -> \*.xtemplate

\_\_FILEBASENAME\_\_.swift

TemplateInfo.plist



# Xcode Template

1. ~/Library/Developer/Xcode/Templates
2. Создаем папку File Templates -> Custom
3. Копируем шаблон
4. Заполняем инфо
- 5 Profit

# Xcode Template

<MODULE>

└ Presenter

└ <MODULE>Presenter.swift

└ VC

└ <MODULE>VC.swift

└ Interactors

└ <MODULE>Interactor.swift

└ Router

└ <MODULE>Router.swift

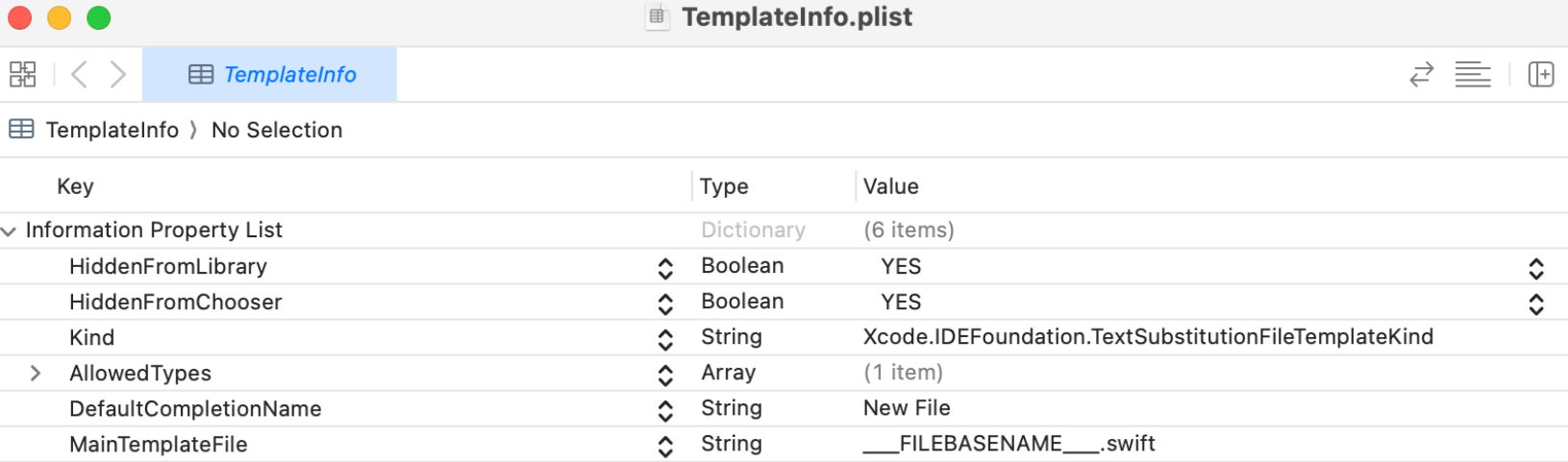
└ Configurator

└ <MODULE>Configurator.swift

# Xcode Template

**HiddenFromLibrary: true**

**HiddenFromChooser: true**



The screenshot shows the Xcode interface for editing a `TemplateInfo.plist` file. The file is displayed as a dictionary with 6 items. The properties are:

Key	Type	Value
Information Property List	Dictionary	(6 items)
HiddenFromLibrary	Boolean	YES
HiddenFromChooser	Boolean	YES
Kind	String	Xcode.IDEFoundation.TextSubstitutionFileTemplateKind
AllowedTypes	Array	(1 item)
DefaultCompletionName	String	New File
MainTemplateFile	String	___FILEBASENAME___.swift



# Xcode Template

Choose a template for your new file:

iOS macOS watchOS tvOS visionOS DriverKit

**Other**

- Exports File (exp)
- Markdown File
- PCH File (pch)
- Shell Script
- StoreKit Configuration File (STOREKIT)
- Test Plan (TESTPLAN)

**Custom**

- MyCleanArchModule
- MyCleanArch (SWIFT)

Cancel

Previous

Next

# Xcode Template

Choose options for your new file:

New Scene Name:

View Controller Name: TestSceneViewController

Interactor Name: TestSceneInteractor

Presenter Name: TestScenePresenter

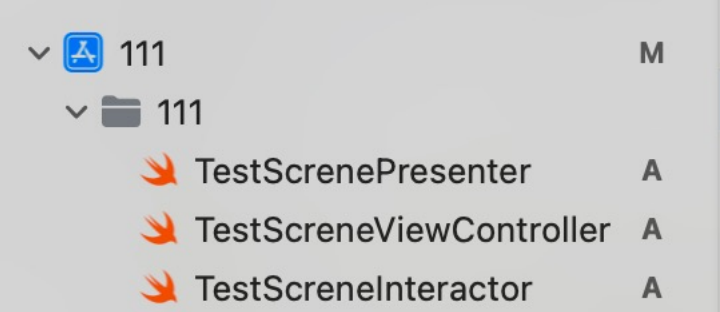
Router Name: TestSceneRouter

Configurator Name: TestSceneConfigurator

Subclass of:

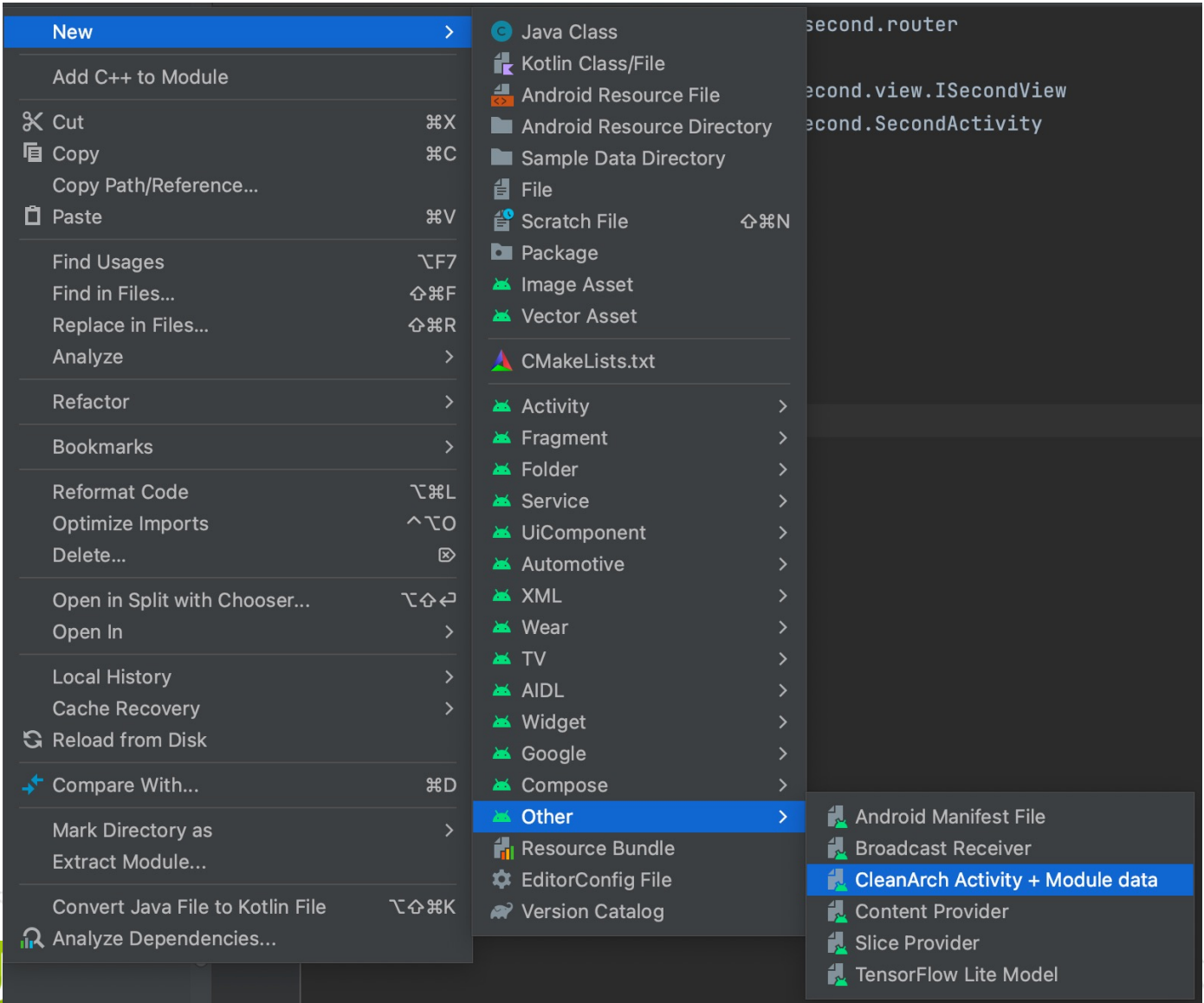
# Xcode Template

Сгенерированные в каталоге приложения файлы

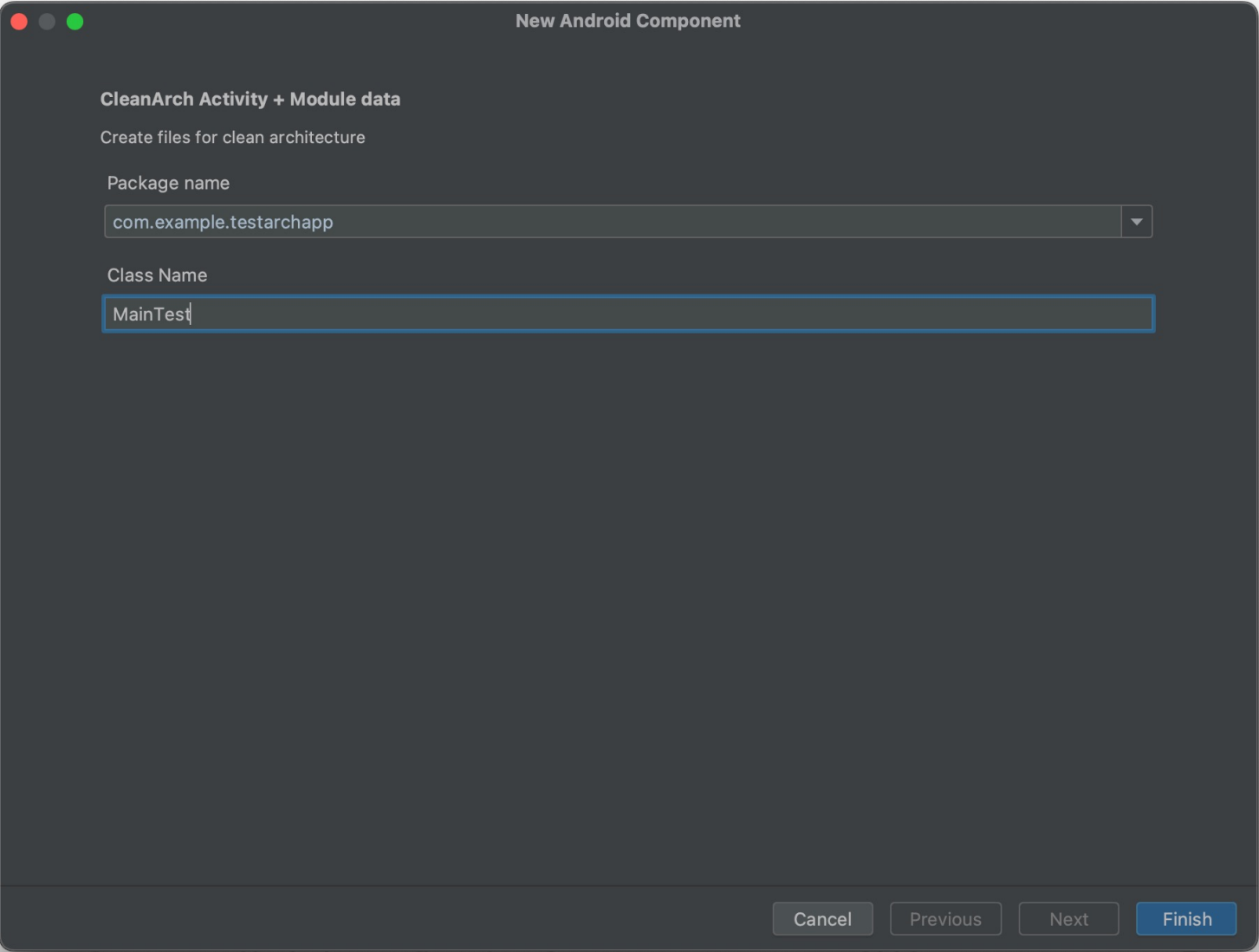


# Чем ответит Android?

# Wizard Template Android Studio



# Wizard Template Android Studio



# А так ли это просто в Android?

- Android Studio Plugin
- IntelliJ Idea Plugin



# Ресурсы по созданию

- Гайды на сайте JetBrains
- ГОТОВЫЕ СЭМПЛЫ И ШАБЛОНЫ

## IntelliJ Platform SDK

Extend the IntelliJ Platform by creating plugins, custom language support, or UI themes.



### Quick Start Guide

Basics of working with the IntelliJ Platform and creating plugins.



### The IntelliJ Platform SDK

Introduction and overview of contents.

## Get started



### Developing a Plugin

Get started building IDE plugins.

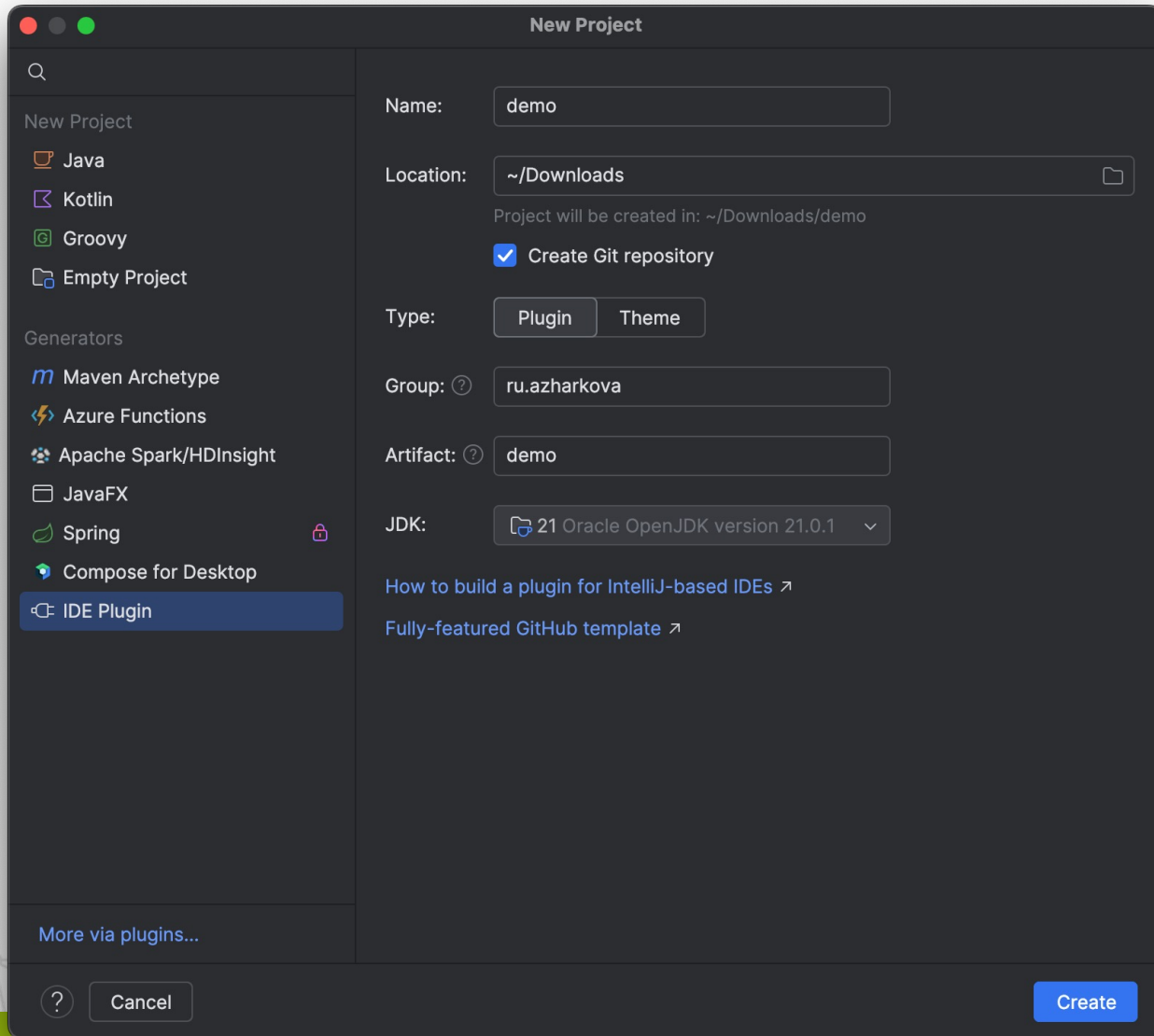


### Developing a Theme

Customize the IDE appearance.

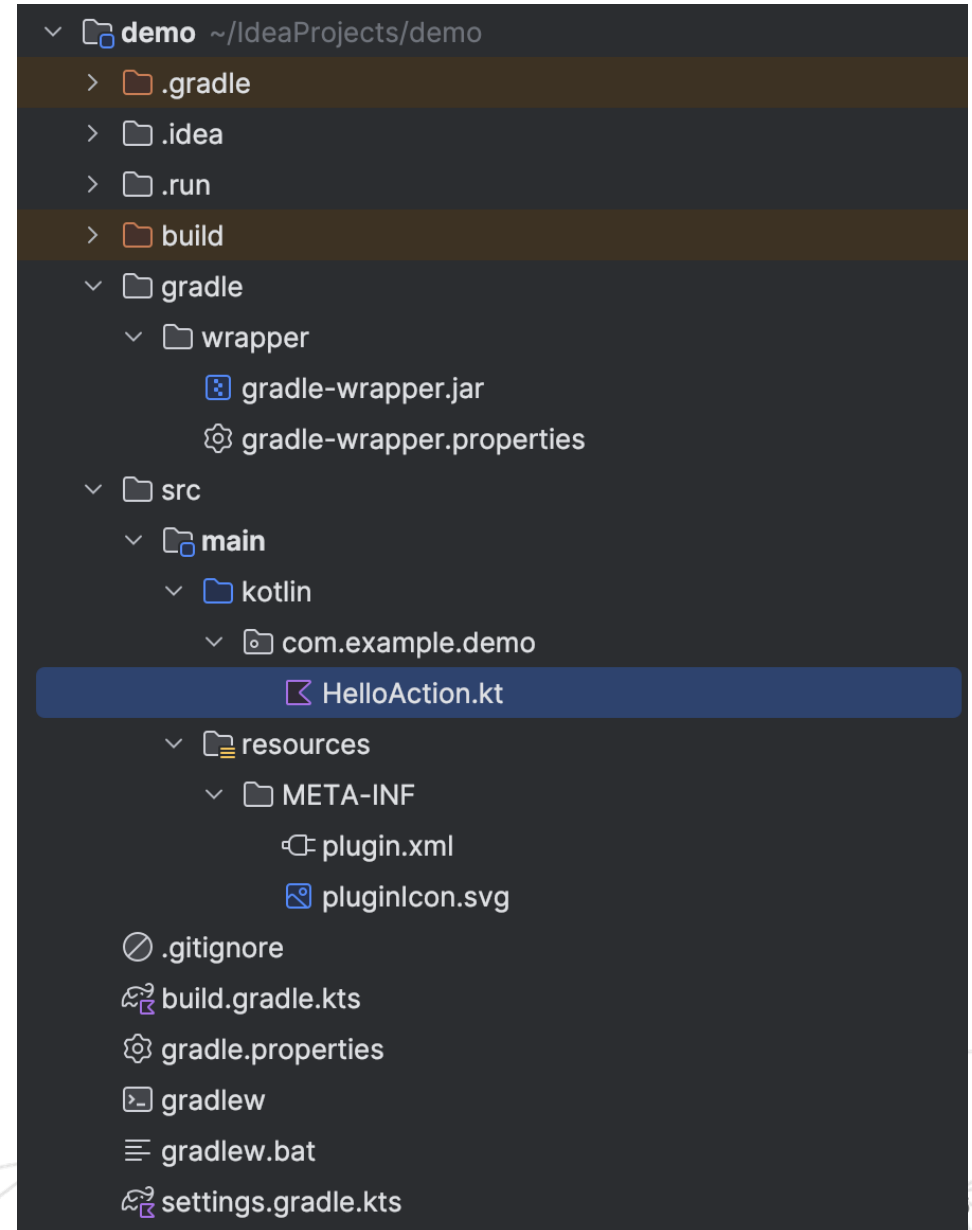


# Плагин IntelliJ Idea



# Структура плагина

Код (действия, шаблоны)  
Resources



# Build.gradle.kts. Настройки плагина

gradle.properties

```
//build.gradle.kts
intellij {
    pluginName.set(properties("pluginName"))
    version.set(properties("platformVersion"))

    type.set(properties("platformType")) // AI

    //Или путь к студии
    //intellij.localPath.set(properties("androidStudioRunPath"))
    //...
}
```

# Plugin.xml. Настройки плагина

```
<idea-plugin>
  <id>com.github.azharkova.cleanarch_plugin</id>
  <name>CleanArch</name>
  <vendor>azharkova</vendor>

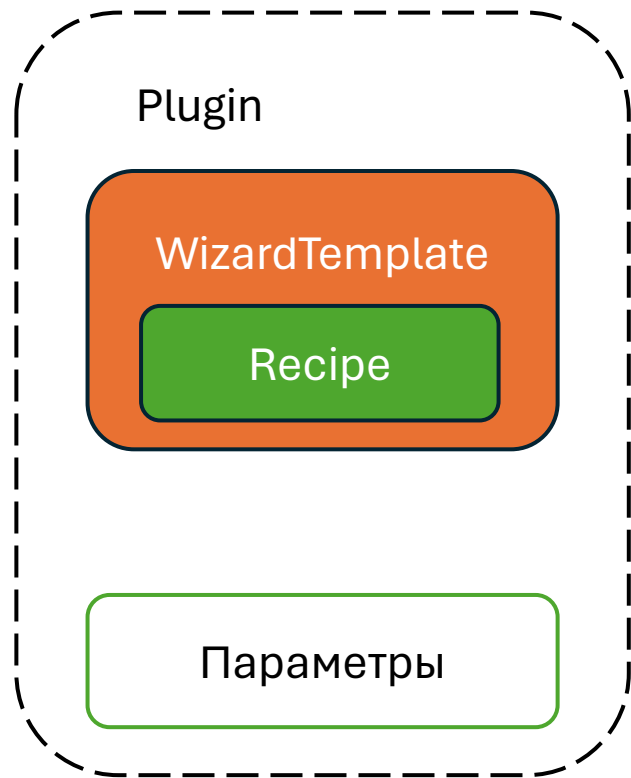
  <depends>org.jetbrains.android</depends>
  <depends>org.jetbrains.kotlin</depends>
  <depends>com.intellij.modules.java</depends>
  <depends>com.intellij.modules.platform</depends>
  <depends>com.intellij.modules.androidstudio</depends>

  <extensions defaultExtensionNs="com.android.tools.idea.wizard.template">
    <wizardTemplateProvider
      implementation="com.github.azharkova.cleanarch_plugin.WizardTemplateProviderImpl" />
  </extensions>

  <applicationListeners>
    <listener class="com.github.azharkova.cleanarch_plugin.listeners.MyProjectManagerListener"
      topic="com.intellij.openapi.project.ProjectManagerListener"/>
  </applicationListeners>
</idea-plugin>
```

**Внимание! Дальше действительно  
несложно**

# Реализуем с помощью плагина IntelliJ Idea



# Wizard Template

Template

Recipe

```
val cleanArchActivityTemplate
    get() = template {
        name = "CleanArch Activity + Module data"
        description = "Create files for clean architecture"
        minApi = 21
        category = Category.Other
        formFactor = FormFactor.Mobile
        screens = listOf(WizardUiContext.MenuEntry)

        /**
         * Работа с параметрами магическая
         */

        recipe = { data: TemplateData ->
            cleanArchActivityTemplate(
                data as ModuleTemplateData,
                packageNameParam.value,
                classNameParam.value
            )
        }
    }
}
```

# Wizard Template

## Params

```
val packageNameParam = defaultPackageNameParameter
val classNameParam = stringParameter {
    name = "Class Name"
    default = "Main"
    help = "Use the class name for prefix"
    constraints = listOf(Constraint.NONEMPTY)
}

widgets(
    PackageNameWidget(packageNameParam),
    TextFieldWidget(classNameParam)
)
```



# Wizard Template

WizardTemplateProvider подключаем в plugin.xml

```
class WizardTemplateProviderImpl : WizardTemplateProvider() {  
    override fun getTemplates(): List<Template> = listOf(cleanArchActivityTemplate)  
}
```

# Wizard Template. Компоненты

Подставляем параметр  
в шаблон

Profit!

```
fun createInteractor(  
    domainPackageName: String = PackageManager.domainPackageName,  
    className: String  
) = """"  
    package $domainPackageName  
  
    import ${PackageManager.presentationPackageName} T${className}Presenter  
    import ${PackageManager.routerPackageName}.I${className}Router  
  
    class ${className}Interactor(  
        private val presenter: I${className}Presenter,  
        private val router: I${className}Router  
    ): I${className}Interactor {  
  
    }  
"""".trimIndent()
```

# Wizard Template. Recipe

Собираем все в Recipe шаблона

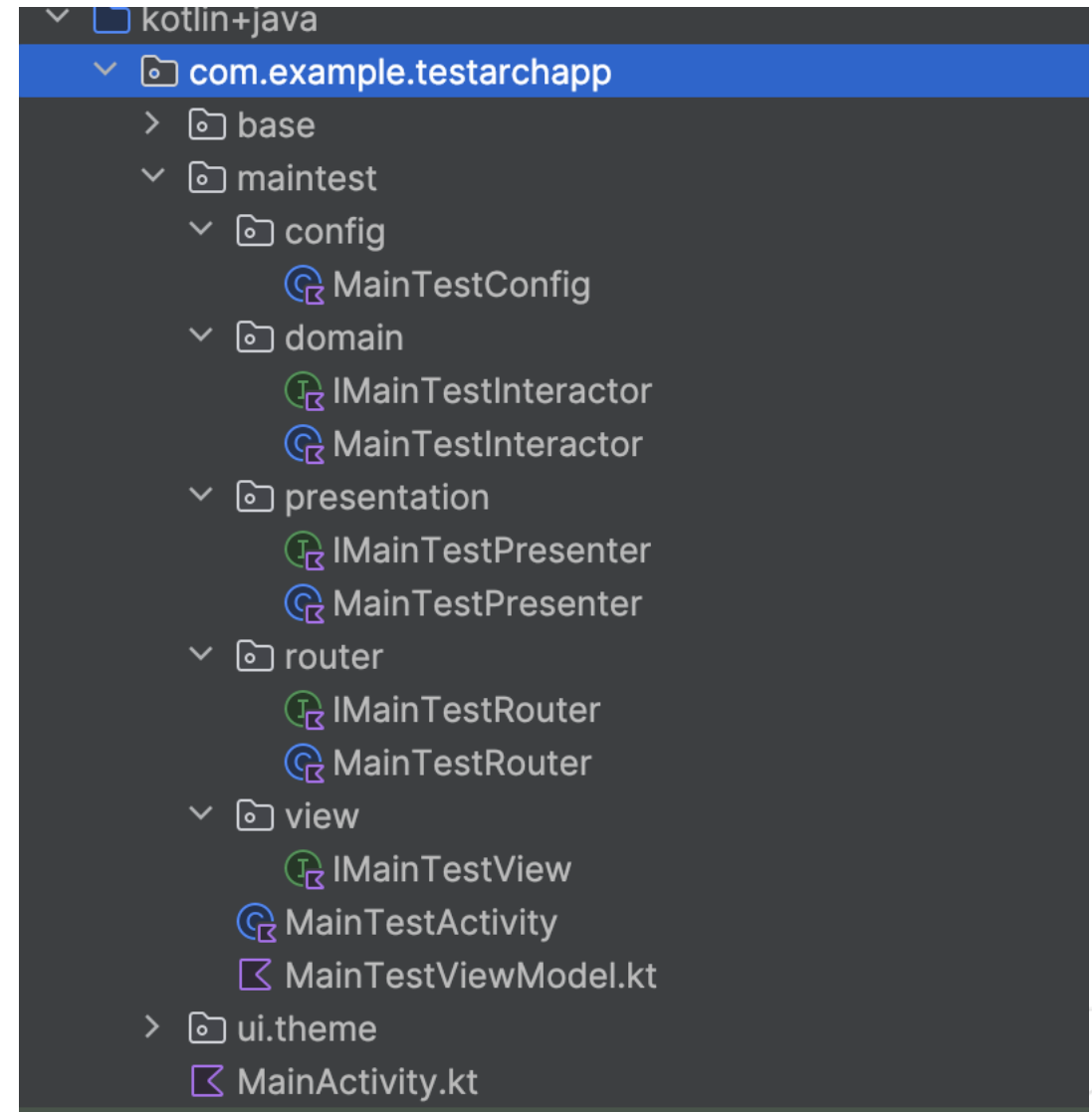
```
fun RecipeExecutor.cleanArchActivityTemplate(  
    moduleData: ModuleTemplateData,  
    packageName: String,  
    className: String  
) {  
  
    val (projectData, _, _, manifestOut) = moduleData  
    val project = projectInstance ?: return  
  
    //Зависимости ...  
  
    val pfm = ProjectFileManager(project)  
  
    createActivity(className = className, manifestOut = manifestOut, moduleData = moduleData)  
        .save(pfm.dirOf(APP),  
            "$packageName.${className.lowercase()}",  
            "${className}Activity".asKt()  
        )  
    //...  
    createInteractor(className = className)  
        .save(pfm.dirOf(APP),  
            PackageManager.domainPackageName,  
            "${className}Interactor".asKt()  
        )  
}
```

# Ресар

1. Создаем WizardTemplate
2. Регистрируем в plugin.xml
3. Маппим параметры из полей визарда
4. Вставляем в шаблон
5. Сохраняем

# Wizard Template

Проверяем результат работы



# Summary

- IntelliJ Idea плагины позволяет создавать свои шаблоны для wizard
- Xcode Template – поддерживает создание и добавление своих плагинов вручную

# Что по баллам

- Android +100
- iOS +50 (не гибко)



**100**



**50**

# Попробуем другой кейс



# Поработаем с существующим кодом

# Задачи оптимизации

- Форматирование, применение стиля
- Генерация моков
- Оптимизация кода
- Конвертация

# Задачи оптимизации

- Форматирование, применение стиля
- Генерация моков
- Оптимизация кода
- Конвертация

**Android Studio Plugin**

# @ToComposable

View -> Composable

```
@ToComposable
class TestText @JvmOverloads constructor(
    context: Context,
    attrs: AttributeSet? = null,
    defStyle: Int = 0
) : AppCompatActivity(context, attrs, defStyle) {

    //To Modifier
    fun setText(text: String) {
        //....
    }
}
```

# План-капкан

1. Создаем Action
2. Регистрируем в plugin.xml
3. Маппим параметры из PSI
4. Вставляем в шаблон
5. Сохраняем

# План-капкан

1. Создаем Action
2. Регистрируем в plugin.xml
3. Маппим параметры из PSI
4. Вставляем в шаблон
5. Сохраняем в тот же файл

# Android Plugin. Action

AnAction – действие плагина

Name – название в меню

```
class ViewToComposableGen : AnAction("View to Composable") {
    private var packageName: String = ""

    override fun beforeActionPerformedUpdate(action: AnActionEvent) {
        //1. Получаем доступ к структуре данных из действия
        val psiClass = getPsiClassFromContext(action) ?: return
        val editor = action.getData(PlatformDataKeys.EDITOR)

        //2. Проверяем условия (по аннотации)
        val isToComposable = psiClass.annotations.any {
            it.text.contains("ToComposable")
        }
        if (!isToComposable || editor == null) {
            return
        }

        //3. Собираем нужные параметры (маппим)
        val (params, applyData) = prepareParams(psiClass)

        //4. Пишем в шаблон
        Generator(
            psiClass.project,
            editor,
            psiClass.containingFile,
            psiClass.name.orEmpty(),
            params,
            applyData,
            element = psiClass.originalElement,
        ).writePreview("")
    }
}
```

# Android Plugin. Action

Actions -> часть контекстного меню Generate

```
<actions>
  <action id="viewToComposable" text="View to Composable"
    class="ru.azharkova.view_to_composable.actions.ViewToComposableGen">
    <add-to-group group-id="GenerateGroup" anchor="last"/>
  </action>
</actions>
```



# Работа с PSI-структурой

The image shows a screenshot of the 'PSI Viewer: TestComposable.kt' application. At the top, there are controls for 'Show PSI structure for:' (set to 'Kotlin file'), 'Show PsiWhiteSpace' (checked), 'Show tree nodes' (checked), and 'Extension:' (set to '.kt').

The main text area displays Kotlin code for a class constructor and methods:

```
17 ) : TextView(context, attrs, defStyleAttr) {
18
19     @PropertyModifier
20     var testProperty: String = ""
21
22     @ClickModifier
23     fun setClick() {
24         this.setOnClickListener { }
25     }
26
27     fun setText(text: String) {}
```

Below the code, the 'PSI Tree' panel shows a hierarchical view of the code structure. The 'CLASS\_BODY' node is expanded, showing its children: 'PsiElement(LBRACE)', 'PsiWhiteSpace', and 'PROPERTY'. The 'PROPERTY' node is further expanded to show 'PsiWhiteSpace'.

To the right of the PSI Tree, there are tabs for 'References', 'Block Structure', and 'Stub Structure'. The 'Stub Structure' tab is active, showing a tree of stubs for the selected code block, including 'PsiJetFileStubImpl', 'KotlinStub\$PACKAGE\_DIRECTIVE', 'KotlinStub\$IMPORT\_LIST', 'KotlinStub\$CLASS', 'KotlinStub\$MODIFIER\_LIST', 'KotlinStub\$PRIMARY\_CONSTRUCTOR', 'KotlinStub\$SUPER\_TYPE\_LIST', and 'KotlinStub\$CLASS\_BODY'.

At the bottom of the viewer, there are three buttons: 'Copy PSI', 'Close', and 'Build PSI Tree'.

# Сохраняем

## WriteCommandAction

```
WriteCommandAction.writeCommandAction(project, file).run<Throwable> {  
    val factory = KtPsiFactory(project)  
    val newFunction = factory.createFunction(functionBody)  
    editor.document.insertString(  
        element.startOffset,  
        newFunction.text + "\n\n"  
    )  
    editor.document.saveToDisk()  
    editor.scrollingModel.scrollToCaret(ScrollType.CENTER)  
}
```

# ГОТОВИМ ИСХОДНИК

Наполняем шаблон замаппленными данными из PSI

```
val functionBody = """
    @Composable
    fun $newFunctionName(
        $params
        modifier: Modifier = Modifier,
        viewConfig: ()->Unit
    ) {
        AndroidView (
            factory = { context ->
                $sourceFunctionName(context = context)
            },
            modifier = modifier,
            update = { view ->
                view.apply {
                    $applyData
                }
                viewConfig()
            }
        )
    }
    """.trimMargin() + "\n\n"
```

# View ToComposable Plugin

```
package com.example.testcomposeplugins

import android.content.Context
import android.util.AttributeSet
import android.widget.TextView
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.viewinterop.AndroidView

@ToComposable
open class TestText constructor(
    context: Context,
    attrs: AttributeSet? = null,
    defStyle: Int = 0
) : TextView(context, attrs, defStyle) {

    @PropertyModifier
    var testProperty: String = ""

    @ClickModifier
    fun setClick() {
        this.setOnClickListener { }
    }

    fun setText(text: String) {}
}
```

# Summary. Android Studio Plugins

- Гибкое решение
- Можно брать параметры из формы, можно из PSI
- Точечное решение
- Можно плагины распространять через маркет JetBrains

# Что по баллам

- Android +100



**200**



**50**

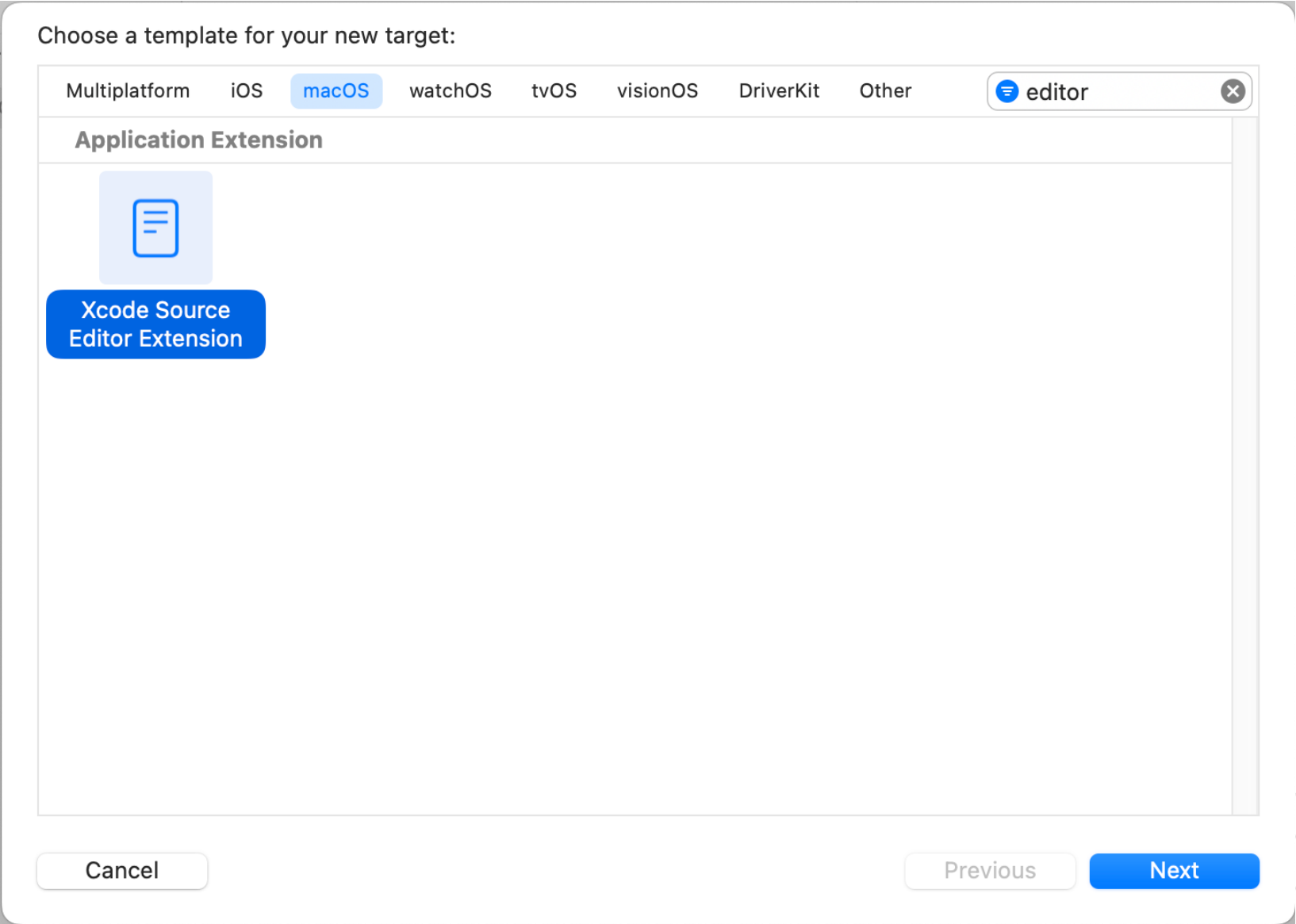
# Чем ответит iOS?

# Xcode Source Editor Extension

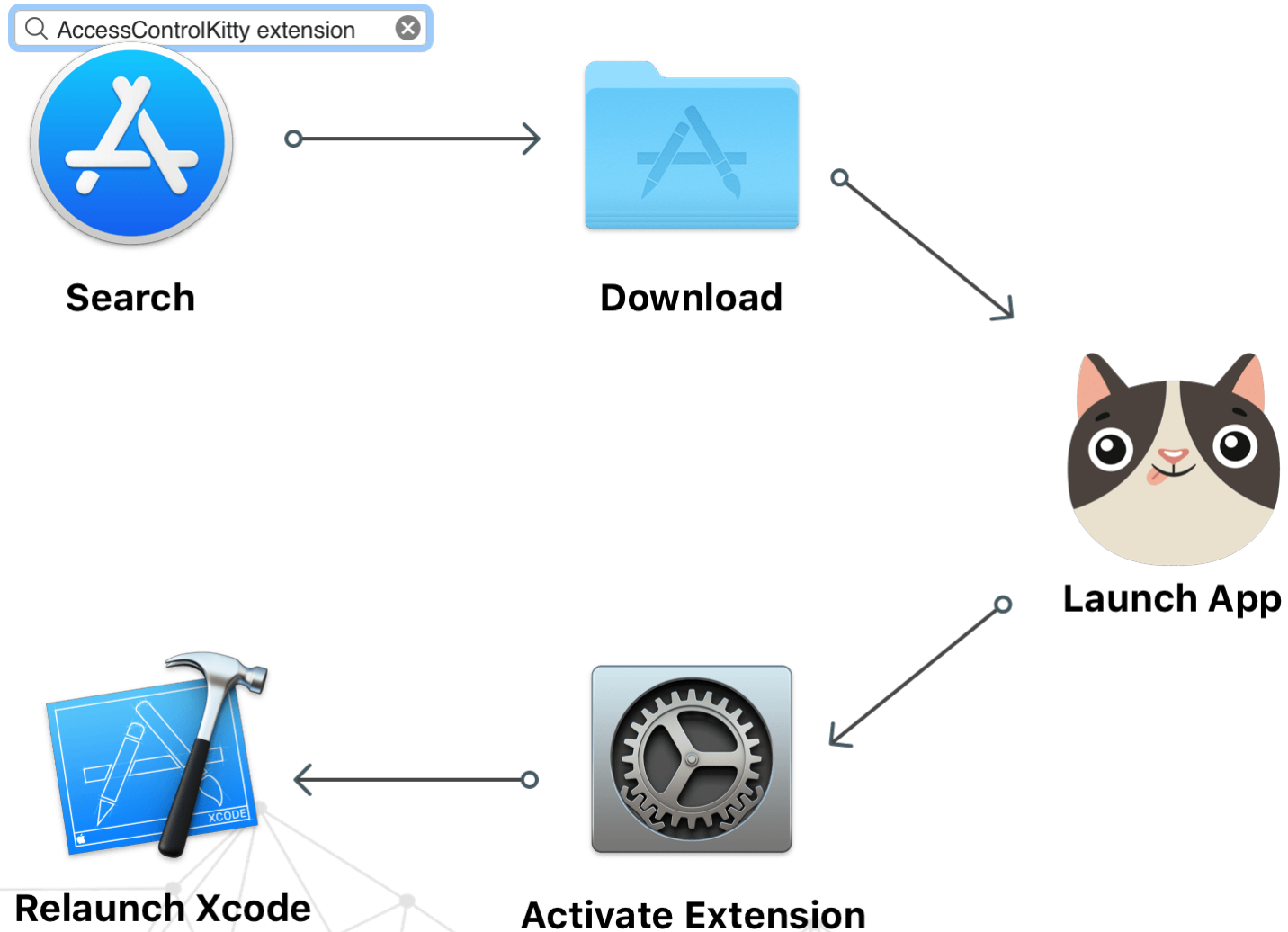


# Добавляем расширение

Таргет для Mac app



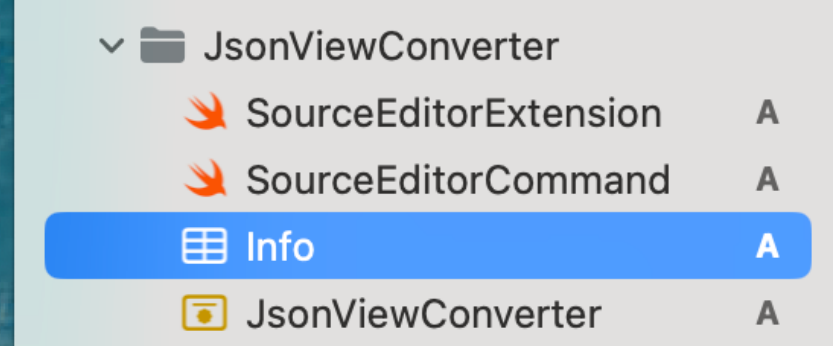
# Как подключать готовое расширение



# Xcode Source Editor Extension. Анатомия

SourceEditorExtension

SourceEditorCommand - Команда



# Xcode Source Editor Extension. Настройка

## Декларация команд

Custom macOS Bundle Target Properties

Key	Type	Value
Bundle version string (short)	String	\$(MARKETING_VERSION)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
NSExtension	Dictionary	(2 items)
NSExtensionAttributes	Dictionary	(2 items)
XCSourceEditorCommandDefinitions	Array	(1 item)
Item 0	Dictionary	(3 items)
XCSourceEditorCommandIdentifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER).SourceEditorCom
XCSourceEditorCommandName	String	Source Editor Command
XCSourceEditorCommandClassName	String	\$(PRODUCT_MODULE_NAME).SourceEditorComman
XCSourceEditorExtensionPrincipalClass	String	\$(PRODUCT_MODULE_NAME).SourceEditorExtensior
NSExtensionPointIdentifier	String	com.apple.dt.Xcode.extension.source-editor
Bundle name	String	\$(PRODUCT_NAME)
Bundle display name	String	JsonViewConverter

# XcodeKit

- Xcode Editor Extensions

Framework

## XcodeKit




Create extensions to add commands to the Xcode source editor.

macOS 10.12+

# Xcode Source Editor Extension. Настройка

XcodeKit -> Embed & Sign

Frameworks and Libraries

Name	Embed
 Cocoa.framework	Do Not Embed ⚡
 Kingfisher	
 XcodeKit.framework	Embed & Sign ⚡
+ -	

# Sourcery

<https://github.com/krzysztofzablocki/Sourcery>

Можно использовать для  
генерации данных под капотом,  
как зависимость

## krzysztofzablocki/ Sourcery

Meta-programming for Swift, stop writing  
boilerplate code.



👤 155

Contributors

📦 19

Used by

★ 7k

Stars

🍴 611

Forks



# Xcode Source Editor Extension

## SourceEditorCommand

```
func perform(with invocation: XCSourceEditorCommandInvocation,
             completionHandler: @escaping (Error?) -> Void ) -> Void
{
    let input = invocation.buffer.lines

    let createOutput = JsonViewConverter.shared.process(input)

    invocation.buffer.lines.insert(input, at: endLineIndex)

    completionHandler(nil)
}
```



# Xcode Source Editor Extension

SourceEditorCommand

```
func perform(with invocation: XCSourceEditorCommandInvocation,
             completionHandler: @escaping (Error?) -> Void ) -> Void
{
    let input = invocation.buffer.lines

    let createOutput = JsonViewConverter.shared.process(input)

    invocation.buffer.lines.insert(input, at: endLineIndex)

    completionHandler(nil)
}
```



# Проблемы и ограничения Xcode Source Editor extensions

- Анализирует один файл за раз
- Не работают в фоне
- Нет доступа к AST, работаем со строками текста
- Нет доступа к структуре проекта или файла



# Xcode Source Editor extensions. Парсинг

Используем регулярки для маппинга в свои модели

```
func initFuncSignatures() throws {
    var funcSignatures: [String] = []
    for line in lines {
        guard let _ = "func .*?\(\".firstMatch(in: line) else { continue }
        let funcSignature = line.replacingOccurrences(of: "{", with: "")
            .replacingOccurrences(of: "\n", with: "")
        funcSignatures.append(funcSignature)
    }
    self.funcSignatures = try funcSignatures.map { try FuncSignature(string: $0) }
}
```

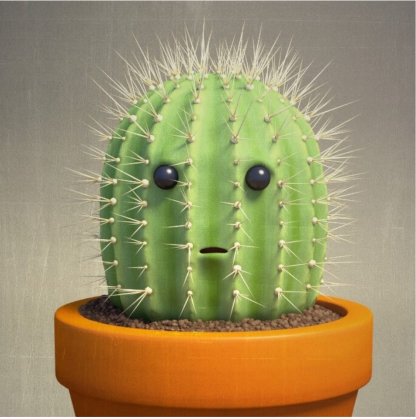
# Json to SwiftUI

```
{
  "type": "VStack",
  "subviews": [
    {
      "type": "Text",
      "properties": {
        "font": "title"
      },
      "values": {
        "text": "LARGE TITLE TEXT"
      }
    },
    {
      "type": "Image",
      "values": { /*...*/ },
      "properties": { /*...*/ },
    },
    {
      "type": "Text",
      "properties": {
        "padding": 16,
        "font": "title",
        "fontWeight": "semibold"
      },
      "values": {
        "text": "Semibold Title"
      }
    },
    {
      "type": "List",
      "properties": {
        "horizontalAlignment": "leading",
        "spacing": 16,
        "padding": 8
      },
      "subviews": [ /*...*/ ]
    }
  ]
}
```



11:19 [status bar icons]

LARGE TITLE TEXT



Semibold Title



Item.1 Title  
Here is multiline  
description text in  
VStack which is inside  
HStack



Item.2 Title  
Here is second multiline  
description text in  
VStack which is inside  
HStack

# JsonSwiftUICommand

Данные берутся из буфера  
(Json текст)

Генерация ответа в виде  
текста (SwiftUIView)

```
class JsonSwiftUICommand : NSObject, XCSourceEditorCommand {
    func perform(with invocation: XCSourceEditorCommandInvocation,
                 completionHandler: @escaping ((any Error?) -> Void) {
        guard let json = NSPasteboard.general.string(forType: .string) else {
            return
        }
        let lines = invocation.buffer.lines
        if let data = json.data(using: .utf8) {
            let newLines = JsonSwiftUIGenerator.decodedViewString(json: data)
            lines.add(newLines)
        }
        completionHandler(nil)
    }
}
```

# Editor -> Extension

Xcode -> Editor

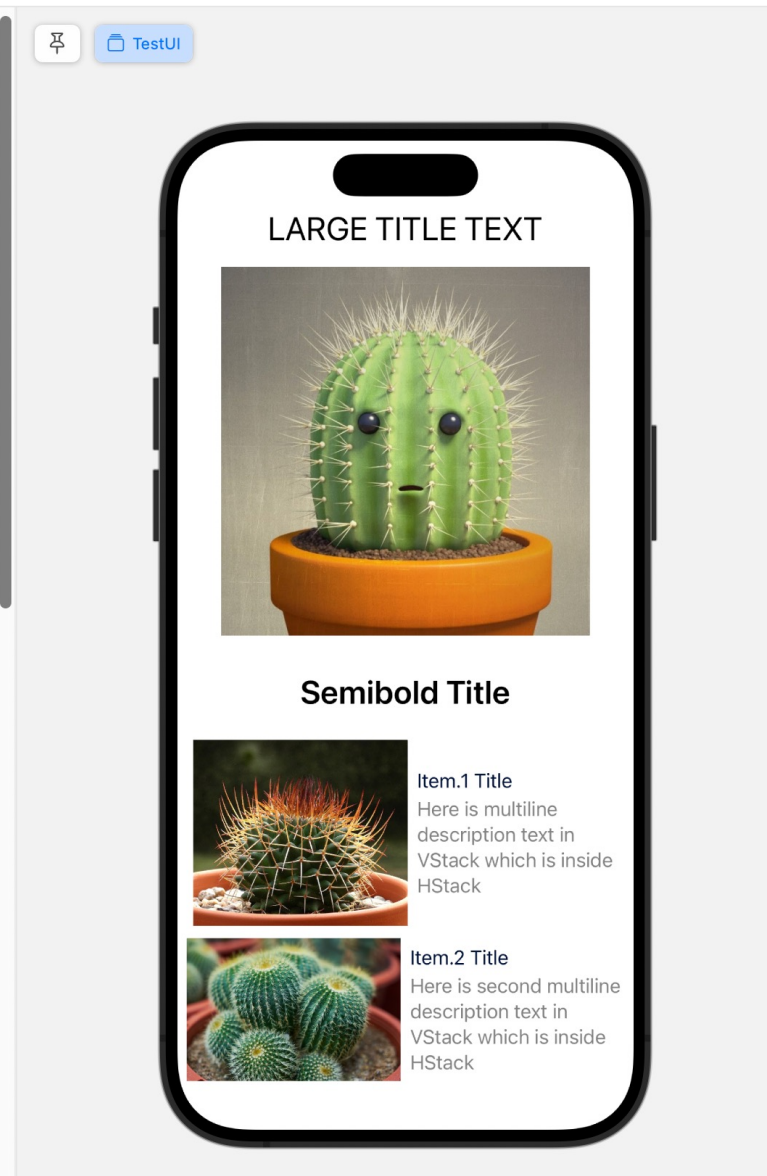
(данные берутся из буфера обмена)





# Смотрим результат. Cmd+C, команда

```
1 //
2 // TestUI.swift
3 // 111
4 //
5 // Created by Anna Zharkova on 21.05.2024.
6 //
7
8 import SwiftUI
9 import Kingfisher
10
11 struct SwiftUIView : View {
12     var body: some View {
13         VStack(alignment: .center, spacing: 0.0) {
14             Text("LARGE TITLE TEXT")
15                 .font(.title)
16                 .fontWeight(.regular)
17             KFIImage(URL(string:
18                 "https://bronk
19                 .club/uploads/posts/2023-03/1677709089_bro
20                 nk-club-p-otkritka-kaktus-vkontakte-26
21                 .jpg")!)
22                 .resizable()
23                 .scaledToFit()
24                 .padding(16.0)
25             Text("Semibold Title")
26                 .font(.title)
27                 .fontWeight(.semibold)
28                 .padding(16.0)
29             VStack {
30                 HStack(alignment: .center, spacing: 8.0) {
31                     KFIImage(URL(string:
32                         "https://www.flowerbunker
33                         .ru/media/TSvety/komnatnyetsvety/k
34                         aktusy/2.jpg")!)
35                         .resizable()
36                         .scaledToFit()
37                     Text("Item.1 Title")
38                         .font(.title)
39                         .fontWeight(.semibold)
40                         .padding(16.0)
41                     Text("Here is multiline
42                         description text in
43                         VStack which is inside
44                         HStack")
45                         .font(.body)
46                         .padding(16.0)
47                 }
48                 HStack(alignment: .center, spacing: 8.0) {
49                     KFIImage(URL(string:
50                         "https://www.flowerbunker
51                         .ru/media/TSvety/komnatnyetsvety/k
52                         aktusy/2.jpg")!)
53                         .resizable()
54                         .scaledToFit()
55                     Text("Item.2 Title")
56                         .font(.title)
57                         .fontWeight(.semibold)
58                         .padding(16.0)
59                     Text("Here is second multiline
60                         description text in
61                         VStack which is inside
62                         HStack")
63                         .font(.body)
64                         .padding(16.0)
65                 }
66             }
67         }
68     }
69 }
```



# Summary. Xcode Source Editor extensions . Pro

- Легковесное решение
- Подходит для точечных задач
- Результат здесь и сейчас
- Результат можно редактировать
- Можно брать исходники из буфера обмена



# Summary. Xcode Source Editor extensions . Cons

- Анализирует один файл за раз
- Не работают в фоне
- Нет доступа к AST, работаем со строками текста
- Нет доступа к структуре проекта или файла

# Что по баллам

- iOS +50 (работает, ну окэй)



**200**



**100**

# Идем вглубь, к компиляторам

# Xcode SPM плагины

# Xcode SPM плагины

- Command Plugin
- BuildTool Plugin

# Command Plugin

- Подключаются как SPM
- Генерируют код внутри другого SPM (песочница)
- Генерируют код для проекта

# Build-Tool Plugin

- Подключаются как SPM
- Прописываются как Run Script в Build Settings
- Запускаются при сборке

# Swift solutions

Вспомогательные решения (библиотеки):

- Xcode arg parser
- Swift-format



# Sourcery

<https://github.com/krzysztofzablocki/Sourcery>

Можно использовать для  
генерации данных под капотом,  
как зависимость

## krzysztofzablocki/ Sourcery

Meta-programming for Swift, stop writing  
boilerplate code.



👤 155

Contributors

📦 19

Used by

★ 7k

Stars

🍴 611

Forks




# SPM Command plugin

## SPM Package


Choose a template for your new package:


Multiplatform macOS Filter


**Library**


  
Library

**Other**

 Empty

 Build Tool Plug-in







 **Command Plug-in**

 Swift Macro

Cancel Previous **Next**

# SPM Command plugin

SPM Package

- ▼  JsonViewConverter M
-  Package M
- ▼  Plugins
  -  JsonViewConverter
- ▼  Sources
  -  JsonViewParser A

# SPM Command plugin

## SPM Package

```
import PackageDescription

let package = Package(
    name: "JsonViewConverter",
    products: [
        .plugin(
            name: "JsonViewConverter",
            targets: ["JsonViewConverter"]),
    ],
    targets: [
        .plugin(
            name: "JsonViewConverter",
            capability: .command(intent: .custom(
                verb: "json-view-convert",
                description: "Converts Json to SwiftUI views"
            )),
            permissions: [
                .writeToPackageDirectory(reason: "Need an access to save data")
            ]
        )
    ]
)
```

# SPM Command plugin

## CommandPlugin

```
import PackagePlugin

@main // Plugin's entry point
struct JsonViewConverter: CommandPlugin {

    func performCommand(context: PluginContext, arguments: [String]) async throws {
        // Магия по сбору информации и подготовке

        let preparedViewData = JsonViewParser.shared.convert(inputDataString)

        try preparedViewData.write(
            to: URL(fileURLWithPath: output.string),
            atomically: true,
            encoding: .utf8)
    }
}
```

# SPM Command plugin

XcodeCommandPlugin

Для поддержки  
применения к проекту

```
import PackagePlugin
import XcodeProjectPlugin

@main // Plugin's entry point
struct JsonViewConverter: CommandPlugin, XcodeCommandPlugin {

    func performCommand(context: XcodeProjectPlugin.XcodePluginContext,
        arguments: [String]) throws {

        //Такая же магия, но немного по-другому с путями
    }
}
```

# Xcode SPM плагины. Минусы

- Нет возможности менять результаты генерации
- Если не предусмотреть, данные запишутся куда-то в build-песочницу
- Нет доступа к сети
- Нет возможности шарить результаты между плагинами

# Когда стоит использовать плагины SPM

- Нужно обработать код в фоне или build-time
- Нужно обработать более 1 файла за раз
- Автоматизация задач (включая скрипты)



# Больше информации про плагины. Mobius 2023

## Swift Plugins: ускоряем сборку проекта

Максим расскажет о том, как с помощью Swift Plugins можно оптимизировать сборку проекта. Подробно разберем, что такое Swift Plugins и какие они бывают. Посмотрим, как при запуске билда с помощниками (SwiftFormat, SwiftGen, Sourcery, Needle, Prefire) можно уменьшить время прогона с 1м 30с до 5с. И все это с помощью инструмента, представленного в 2022 году – Swift Package plugins.

P. S.: Плагины можно использовать как для Swift Package Manager, так и для Xcode 14.

### Спикеры



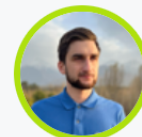
**Максим Гришутин**

Ozon

▾ Биография

Страница спикера →

### Эксперты



**Дмитрий Чернов**

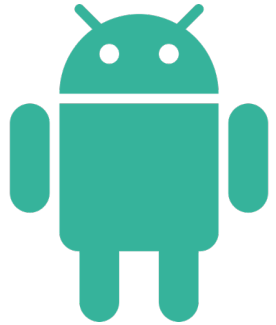
OZON

▾ Биография

Страница эксперта →

# Что по баллам

- iOS +50 (работает, ну окэй)



**200**

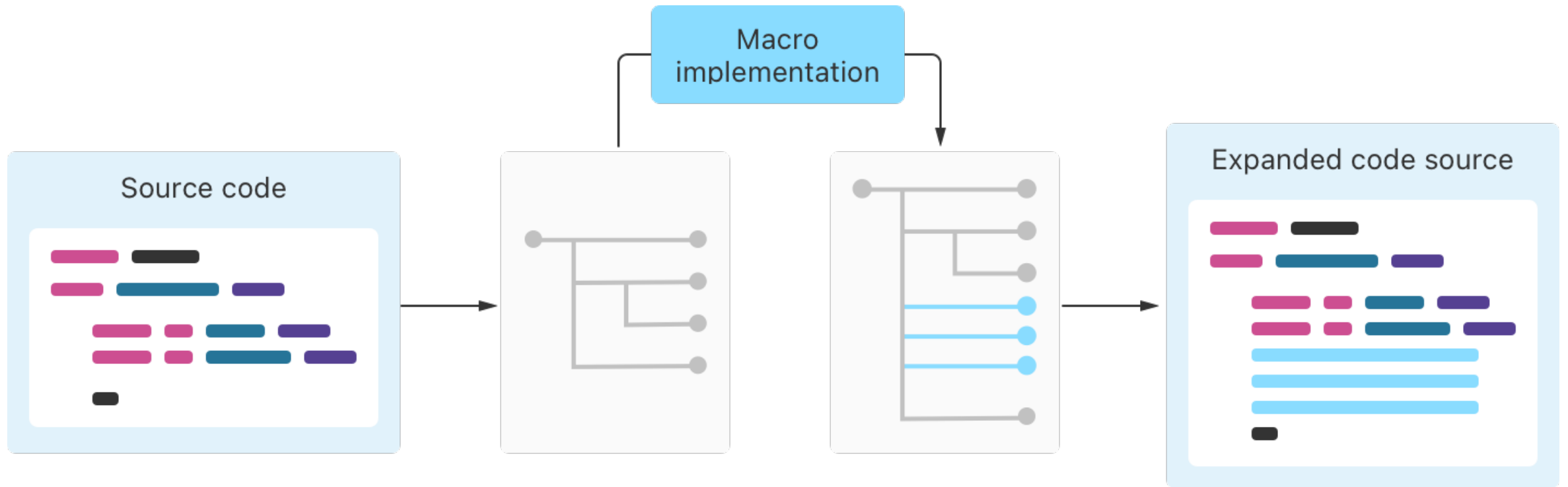


**200**

И бойлерплейт за собой убери



# Макросы Swift 5.9. Расширение

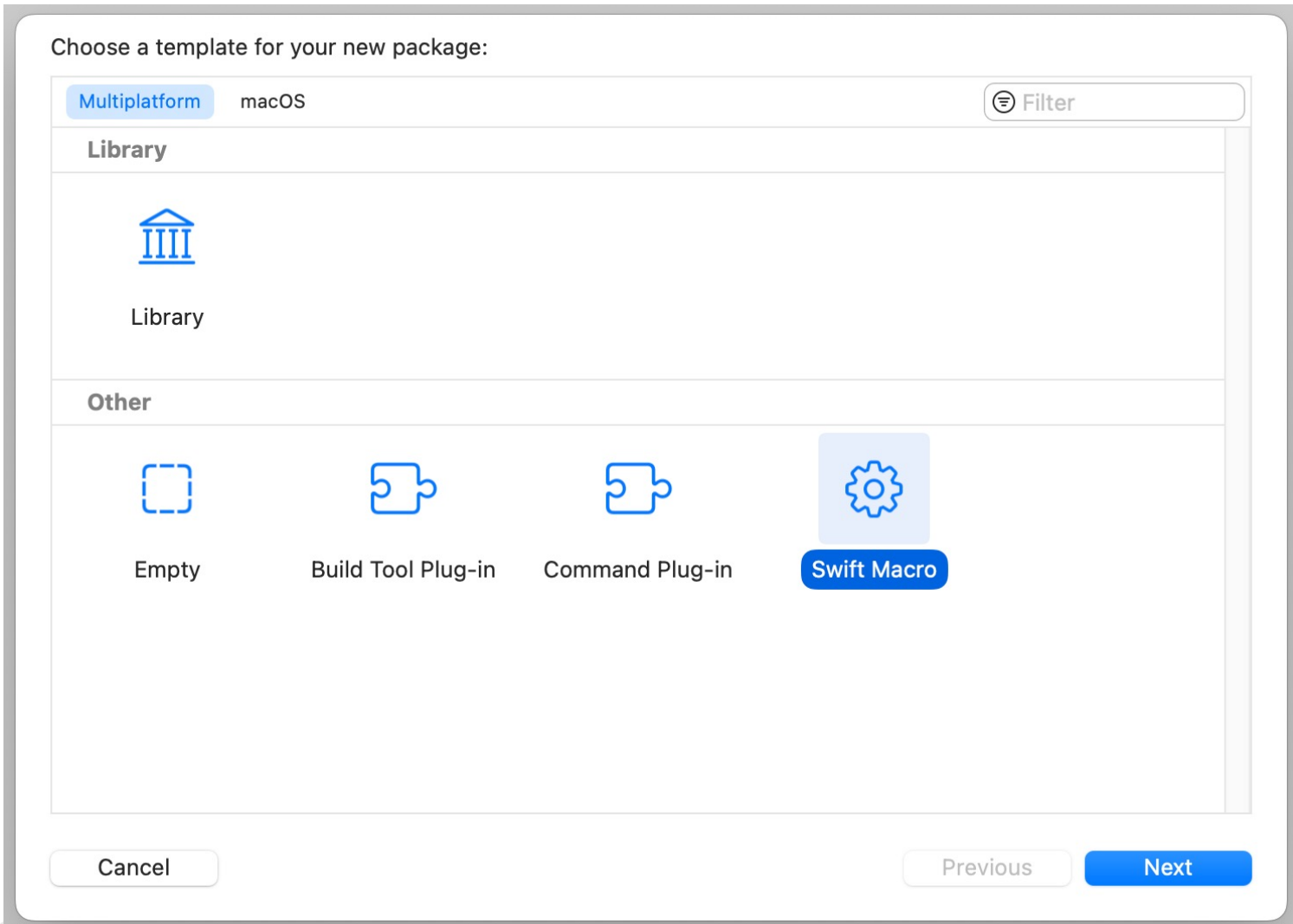


# Создание макроса. Расширение

Моментальный результат

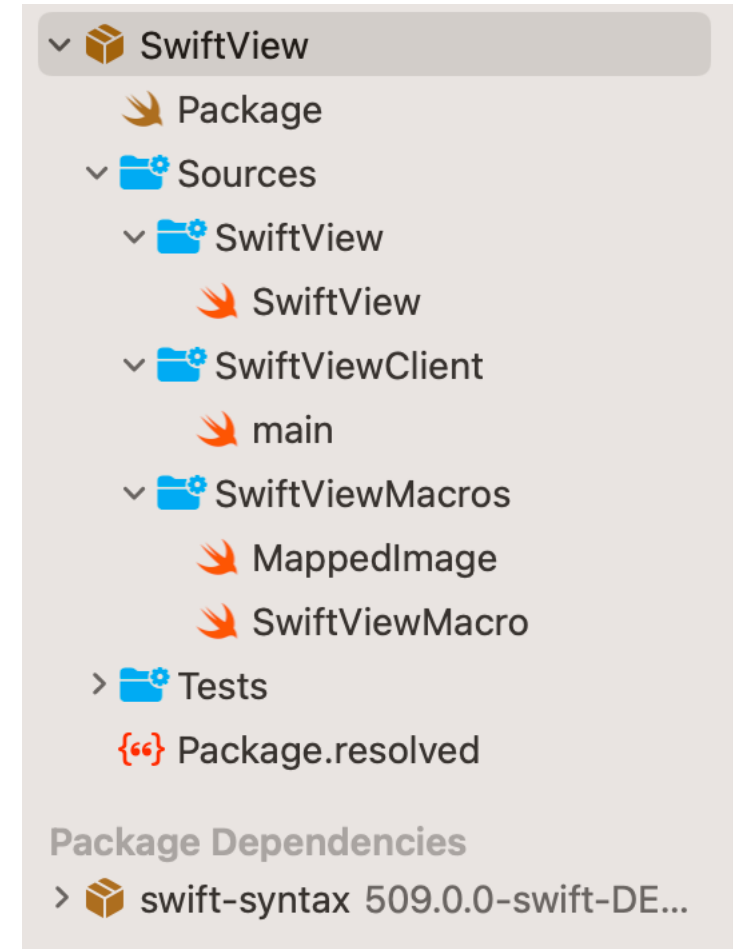
```
18 @AddAsync
19     func fetch(completion: (Data) -> Void) {
20         completion(Data())
21     }
22
23     func fetch() async -> Data { @AddAsync
24         await withCheckedContinuation { continuation in
25             fetch { value in
26                 continuation.resume(with:
27                     .success(value))
28             }
29         }
30     }
31 }
```

# Создание макроса



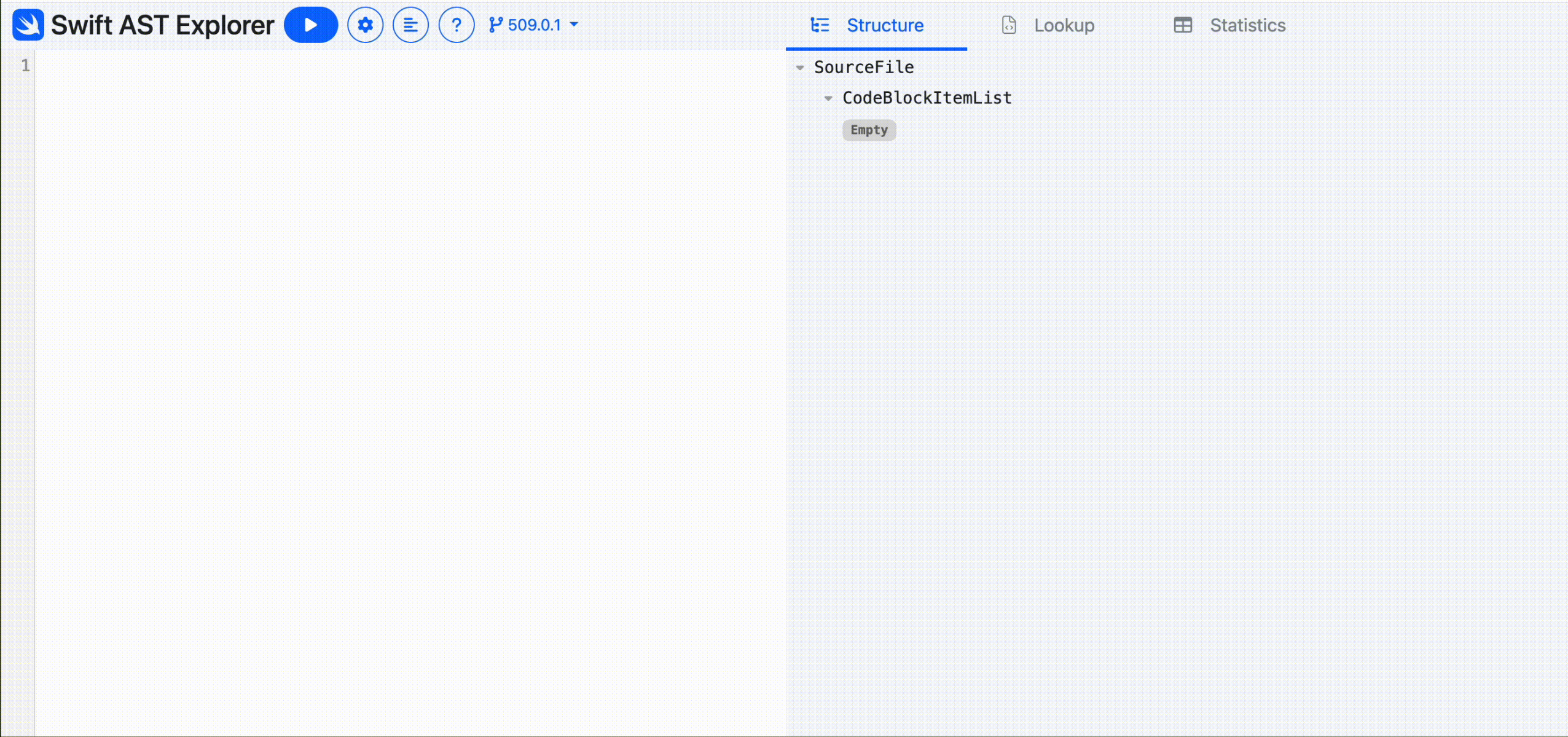
# Создание макроса

- Макрос
- Плагин компилятора Swift
- Тестовое окружение





# AST Explorer в помощь





# Макрос Swift

- SPM
- Имеет роли и уровни доступа к элементам AST
- Работает с AST
- Compile-time

# Макрос Swift. Нюансы и сложности

- Каждый макрос работает с кодом независимо
- Сложность комбинации ролей для решения задач
- API меняется

# Swift макросы. Mobius 2023

ДОКЛАД Под капотом Swift 10.11.2023 / 20:00 – 20:45 UTC+7 (Asia/Barnaul)

ЗАЛ 2

## Макросы Swift: проще, чище, быстрее

### Спикеры



**Анна Жаркова**

Usetech

▼ Биография

Страница спикера →

### Эксперты



**Юрий Дубовой**

Делимобиль

▼ Биография

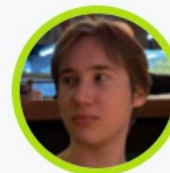
Страница эксперта →

# Swift макросы. Mobius 2024

ДОКЛАД **Под капотом** 01.06 / 20:00 – 20:45 UTC+7 (Asia/Barnaul)

## Макросы спустя полгода: не все так гладко

### Спикеры



**Дмитрий Головин**

Ozon

▾ Биография

Страница спикера →

### Эксперты



**Анна Жаркова**

Usetech

▾ Биография

Страница эксперта →

# Что по баллам

- iOS +200



**200**



**400**

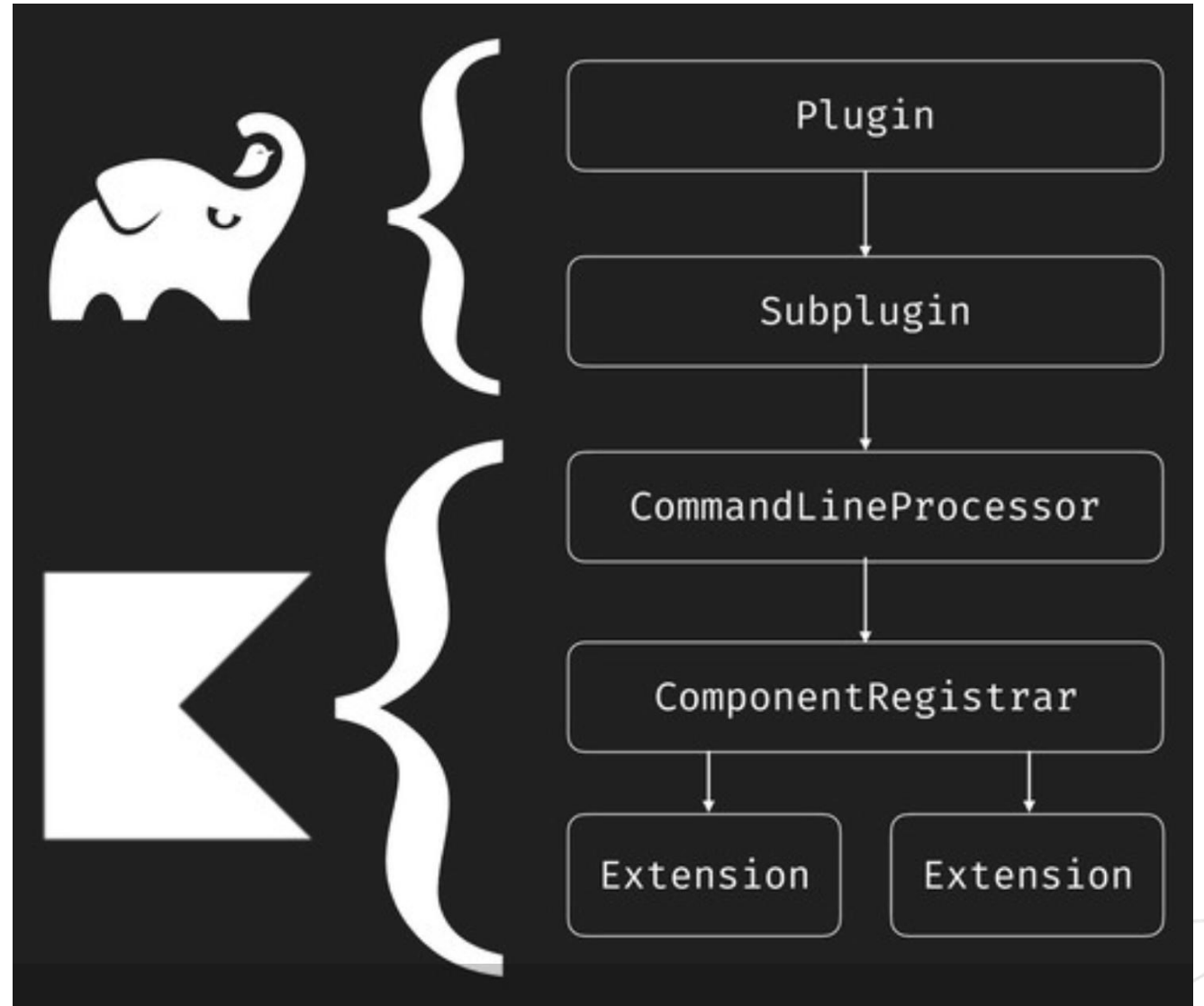
# Чем ответит нам Android?

# Android-генераторы build-time

- Kotlin Symbol Processing
- Kotlin Compiler Plugin

# KCP – Kotlin Plugin Compiler

Мощный инструмент для  
изменения текущего кода во время  
КОМПИЛЯЦИИ





# Что могут плагины компилятора

- Доступ к абстрактному синтаксическому дереву
- Изменяют существующий код
- Мультиплатформенное решение

# Генерация

```
@GenUseCase(repo = NewsApi::class, request = "loadNews")  
class NewsLoadCase
```

```
//После генерации  
class NewsLoadCase {  
    companion object {  
        fun usecase() = $usecase  
    }  
  
    object $usecase : SuspendUseCase {  
        val repo by lazy {  
            NewsApiImpl()  
        }  
  
        override suspend fun execute(param: Any?): Any {  
            return repo.loadNews()  
        }  
    }  
}
```

# Для чего плагины компилятора

- Модифицировать internal/private код
- Мета-программирование (анализ код с помощью аннотаций)

# Когда не подойдут плагины компилятора

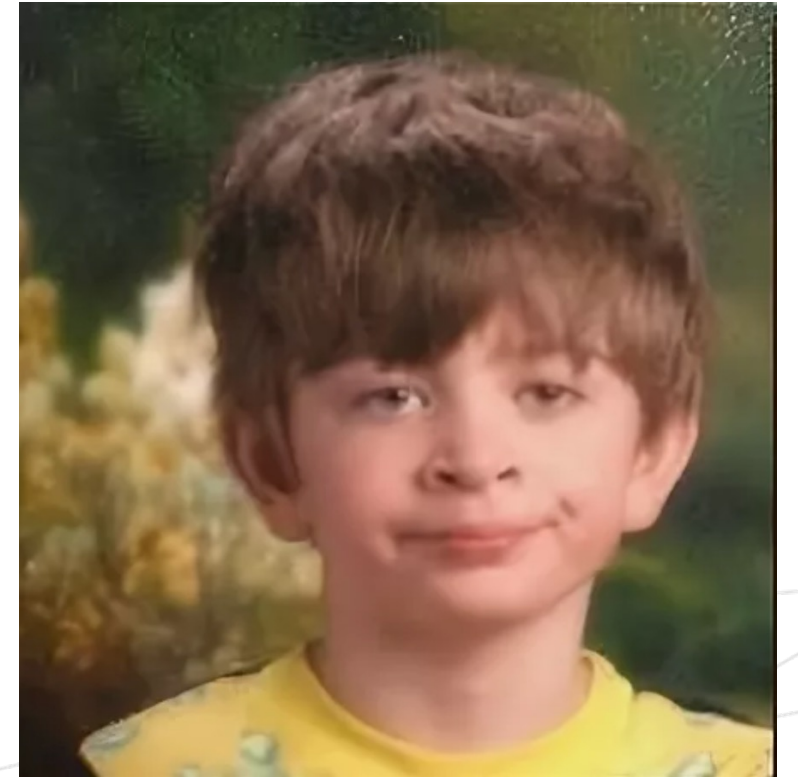
Когда их можно заменить процессингом аннотаций:

- Анализ и генерация только нового кода
- Совмещение с другими плагинами компиляции (KSP идет до своих плагинов компилятора)

Или слишком много работы

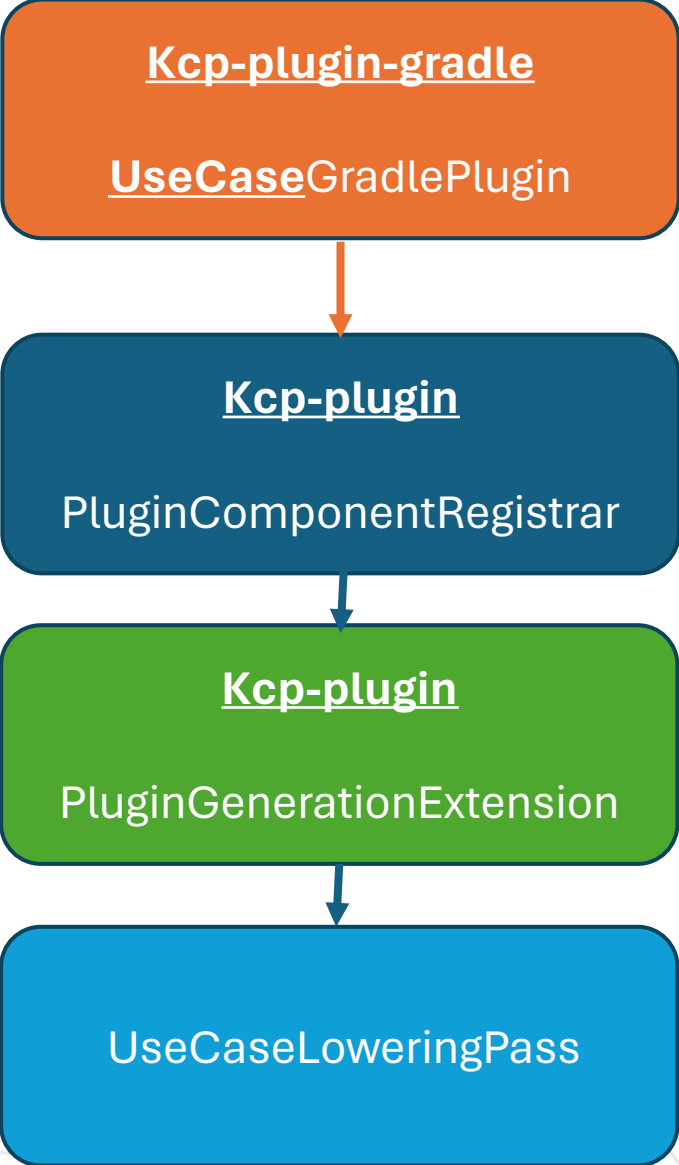
# КСР. Ограничения

- Типы указываем явно – указание ссылки на родительский/абстрактный тип/интерфейс дает доступ к родителю
- Kotlin/Java примитивы, KClass<\*>, Array, String
- Нужно точно знать внутреннее API для вызова



# Архитектура решения

@GenUsecase



# Регистрация расширения

IrGenerationExtension

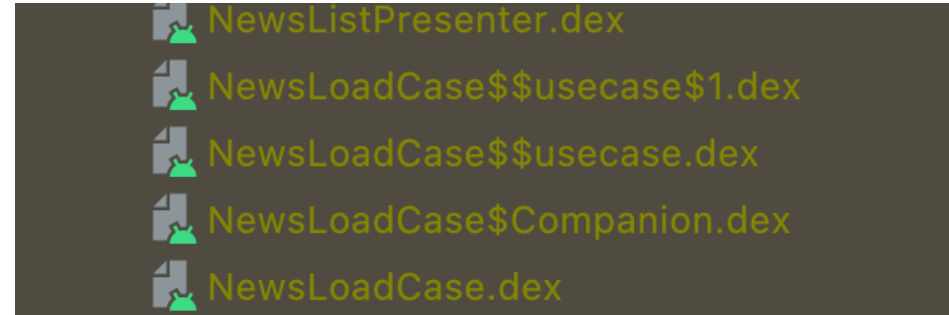
```
class PluginGenerationExtension (private val messageCollector: MessageCollector)
    : IrGenerationExtension {

    override fun generate(moduleFragment: TrModuleFragment, pluginContext: TrPluginContext) {
        UseCaseLoweringPass(pluginContext, messageCollector).lower(moduleFragment)
    }
}
```

# Все скрыто в dex

Новый код -> dex

Dex будет доступен после компиляции





# @GenUseCase. Dex

NewsLoadCase\$\$\$usecase\$1.dex

Files under the "build" folder are generated and should not be edited.

Load Proguard mappings... x file defines 1 classes with 4 methods, and references 7 metho

Class	Defined Methods	Referenced Met...	Size
com	4	6	273 B
azharkova	4	6	273 B
kmmkspcases	4	6	273 B
NewsLoadCase\$\$\$usecase\$1	4	4	257 B
<clinit>()	1	1	46 B
<init>()	1	1	40 B
com.azharkova.kmmkspcases.NewsApilmpl invoke()	1	1	40 B
java.lang.Object invoke()	1	1	39 B
com.azharkova.kmmkspcases.NewsLoadCase\$\$\$usecas			10 B
NewsApilmpl		1	8 B
<init>()		1	8 B
NewsLoadCase\$\$\$usecase		1	8 B
<init>()		1	8 B
kotlin		1	8 B
jvm		1	8 B
internal		1	8 B

# @GenUseCase. Байткод

В байткоде можно посмотреть все сгенерированные методы, свойства и вызовы

```
DEX Byte Code for NewsLoadCase$$usecase$1
.class final Lcom/azharkova/kmmkspcases/NewsLoadCase$$usecase$1;
.super Lkotlin/jvm/internal/Lambda;
.source "NewsLoadCase.kt"

# interfaces
.implements Lkotlin/jvm/functions/Function0;

# annotations
.annotation system Ldalvik/annotation/EnclosingMethod;
    value = Lcom/azharkova/kmmkspcases/NewsLoadCase$$usecase$1;
.end annotation

.annotation system Ldalvik/annotation/InnerClass;
    accessFlags = 0x18
    name = null
.end annotation

.annotation system Ldalvik/annotation/Signature;
    value = {
        "Lkotlin/jvm/internal/Lambda;",
        "Lkotlin/jvm/functions/Function0<",
        "Lcom/azharkova/kmmkspcases/NewsApiImpl;",
        ">";
    }
}
```

# Kotlin compiler plugins

ГОТОВЫЙ СЭМПЛ

<https://github.com/anioutkazharkova/ksp-kmm-cases>

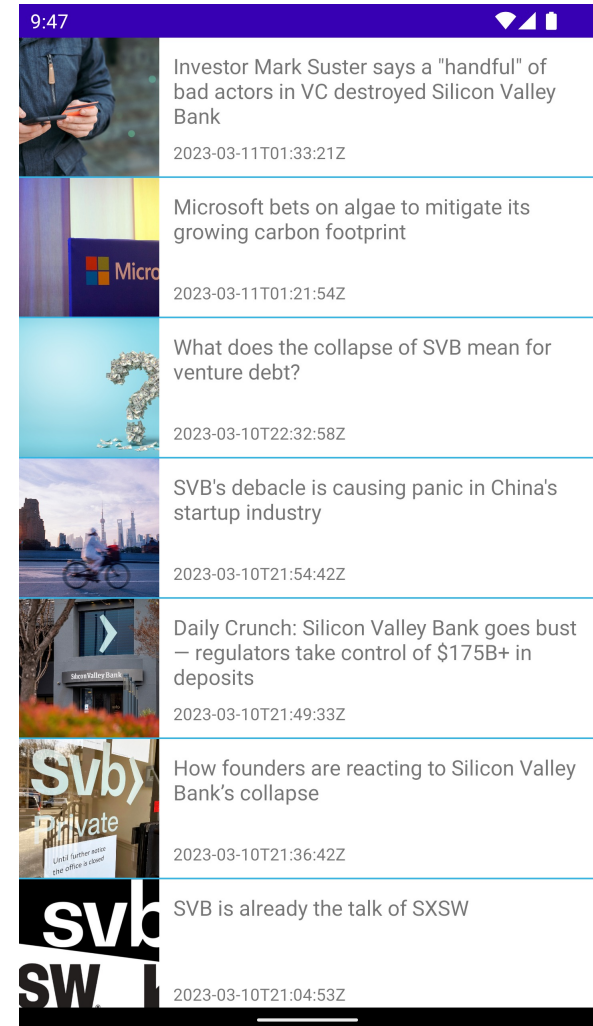
## Спикеры



**Анна Жаркова**  
Usetech

ДОКЛАД Под капотом 12.05 / 20:15 – 21:00 (UTC+7)

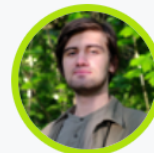
## Оно само: используем плагины компиляции



# Kotlin compiler plugins. 2023

Компиляторные плагины: модификация и анализ Compose

## Спикеры



**Асхар Айдаров**

VK / ВКонтакте

▾ Биография

Страница спикера →

## Эксперты



**Никита Куликов**

Flipper Devices

▾ Биография

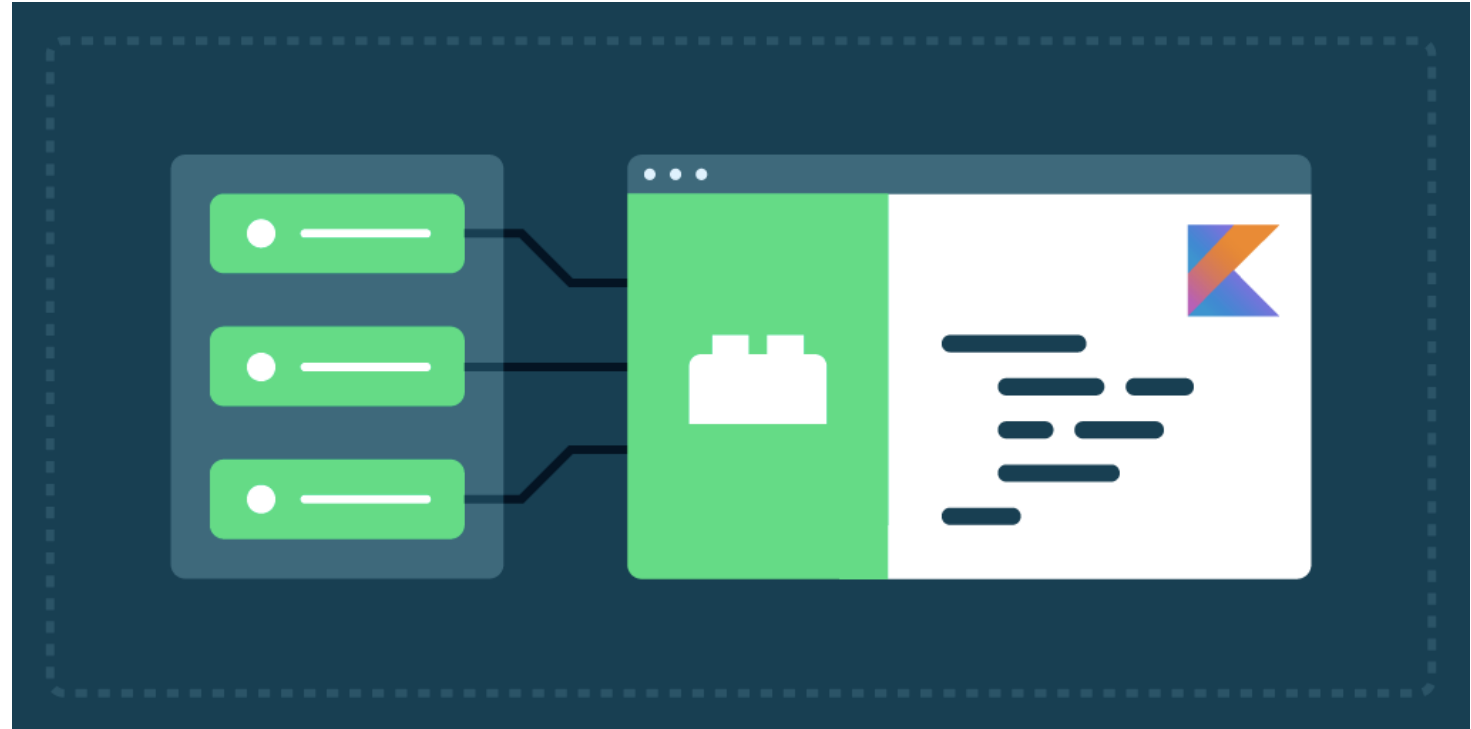
Страница эксперта →

# Kotlin Symbol Processing

[2.0.0-1.0.21](#)

<https://github.com/google/ksp>

KSP – рекомендуемый инструмент  
от Google (2023)



# Когда подойдет процессинг аннотаций

- Генерация нового кода
- Использование шаблонирования

# Когда не подойдет процессинг аннотаций

- Изменение существующего кода

Kotlin Compiler Plugin

# Когда не подойдет процессинг аннотаций

- Изменение существующего кода

Kotlin Compiler Plugin

KCP +KSP



# Готовые решения

- Room
- Glide
- Mochi
- Koin

# Kotlin symbol processing. Анатомия

Annotation Processor ->  
Symbol processor

```
/**
 * Интерфейс провайдера
 */
fun interface SymbolProcessorProvider {
    fun create(environment: SymbolProcessorEnvironment): SymbolProcessor
}

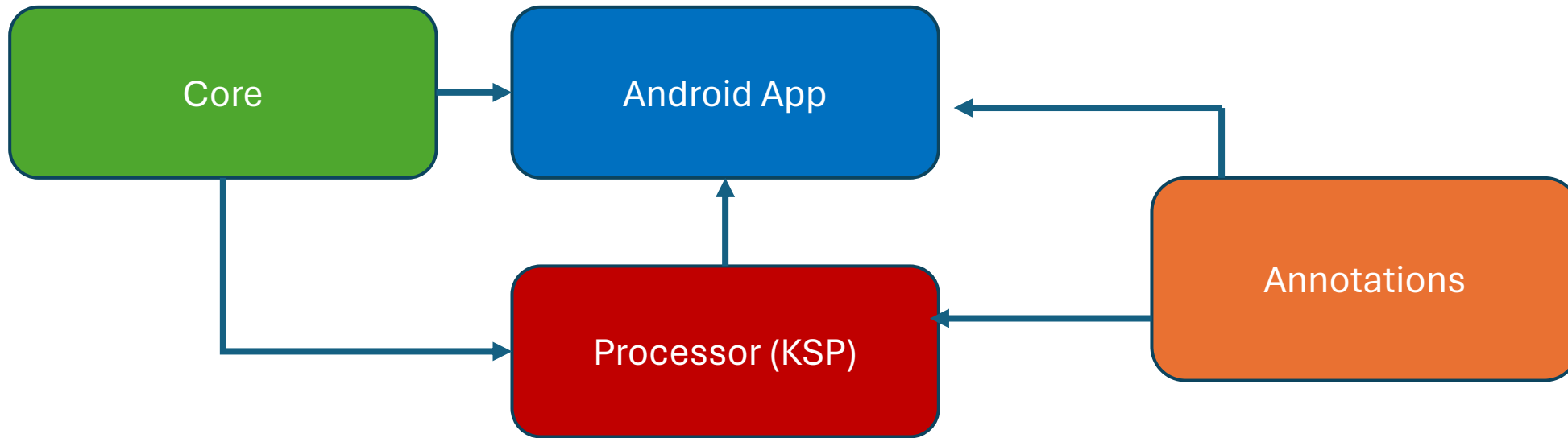
//Интерфейс процессора
interface SymbolProcessor {

    /**
     * Смотрим сюда
     */
    fun process(resolver: Resolver): List<KSAnnotated>

    fun finish() {}

    fun onError() {}
}
```

# Подключение к проекту



# @ToComposable. AndroidView

View < - > Composable

```
@Target(AnnotationTarget.CLASS, AnnotationTarget.FUNCTION)
@Retention(AnnotationRetention.SOURCE)
annotation class ToView
```

```
@Target(AnnotationTarget.CLASS)
@Retention(AnnotationRetention.SOURCE)
annotation class ToComposable
```

# Собственный Processor

## SymbolProcessor

```
class ComposeProcessor constructor(private val env: SymbolProcessorEnvironment)
: SymbolProcessor {

    override fun process(resolver: Resolver): List<KSAnnotated> {
        val viewVisitor = ViewVisitor(resolver, codeGenerator, logger)

        val views = toComposables(resolver)

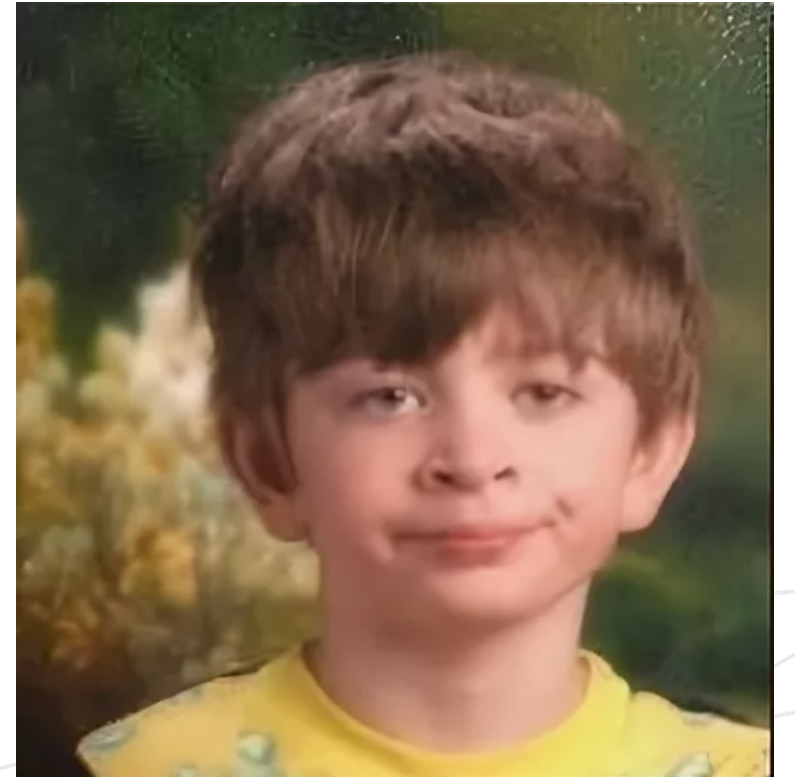
        views.forEach {
            it.accept(viewVisitor, Unit)
        }
        it.accept(functionVisitor, Unit)
    }
    return emptyList()
}
```

# Стандартный план-капкан по работе с KSP

1. Ищем KSDeclaration с определенными аннотациями
2. Маппим модели, зависимости и параметры
3. Генерируем классы и методы
4. Profit!

# Kotlin Symbol Processing. Ограничения

- Нельзя изменить существующие данные и файлы
- Только генерация новых конструкций в новых файлах
- Типы указываем явно – указание ссылки на родительский/абстрактный тип/интерфейс дает доступ к родителю
- Kotlin/Java примитивы, `KClass<*>`, `Array`, `String`



# Результат

Готовые данные доступны для  
использования и компиляции

Build -> generated

```
kotlin/.../TestText.kt ×
Generated source files should not be edited. The changes will be lo

1  package com.azharkova.kspgenandroid
2
3  import ...
4
5
6
7
8
9  @Composable
10 public fun TestText(
11     click: (() -> Unit),
12     text: String,
13     modifier: Modifier = Modifier,
14     viewConfig: TestText.() -> Unit,
15 ): Unit {
16     AndroidView(
17         factory = { context -> TestText(context) },
18         modifier = modifier,
19         update = { view ->
20             view.apply { this: TestText
21                 setText(text)
22                 setClick().apply { this: Unit
23                     click.invoke()
24                 }
25                 viewConfig()
26             }
27         }
28     )
29 }
```



# KSP. Composable экран с логикой

ГОТОВЫЙ СЭМПЛ

<https://github.com/anioutkazharkova/kgen-android>

## Спикеры



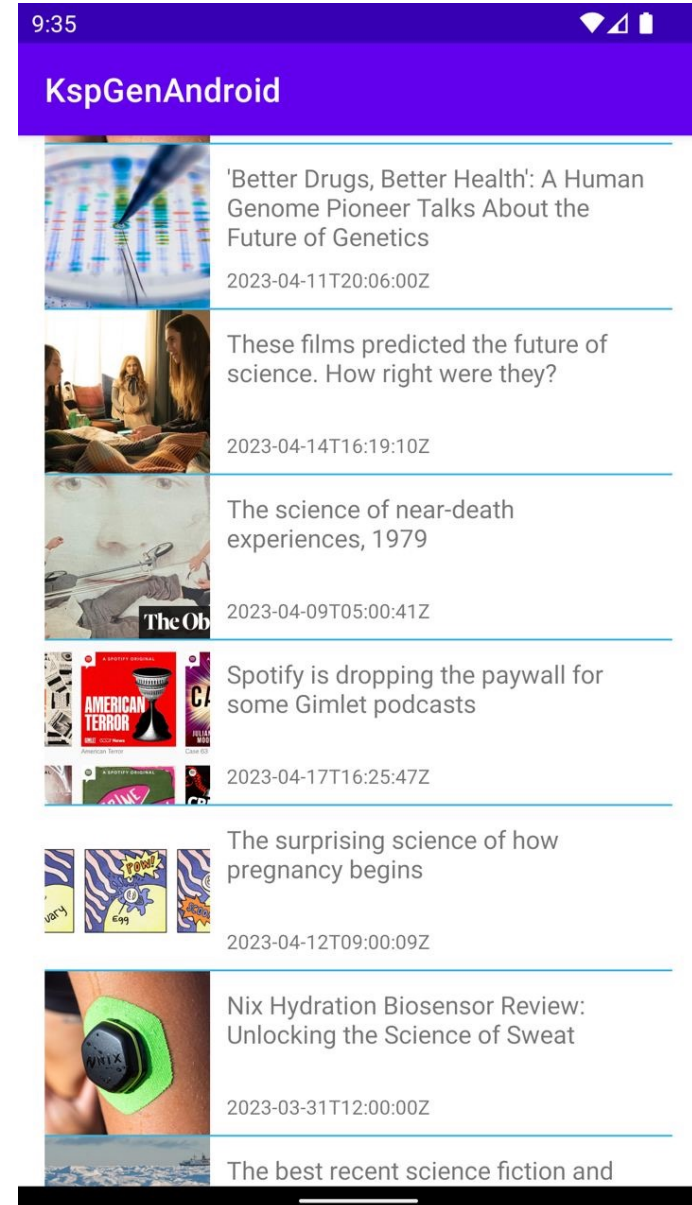
**Анна Жаркова**  
Usetech

ДОКЛАД

UI/UX в мобильной разработке

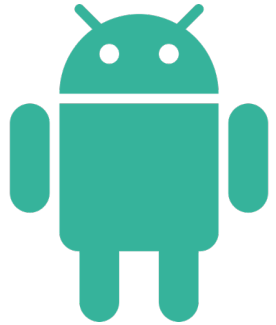
19.05 / 18:45 – 19:30 (UTC+7)

## Упрощаем и укрощаем UI для Android с помощью аннотаций



# Что по баллам

- Android +150



**350**



**250**

**А если я хочу, чтобы  
вообще все само?**



# Генерация с помощью ИИ

- GPT
- Copilot
- Gemini

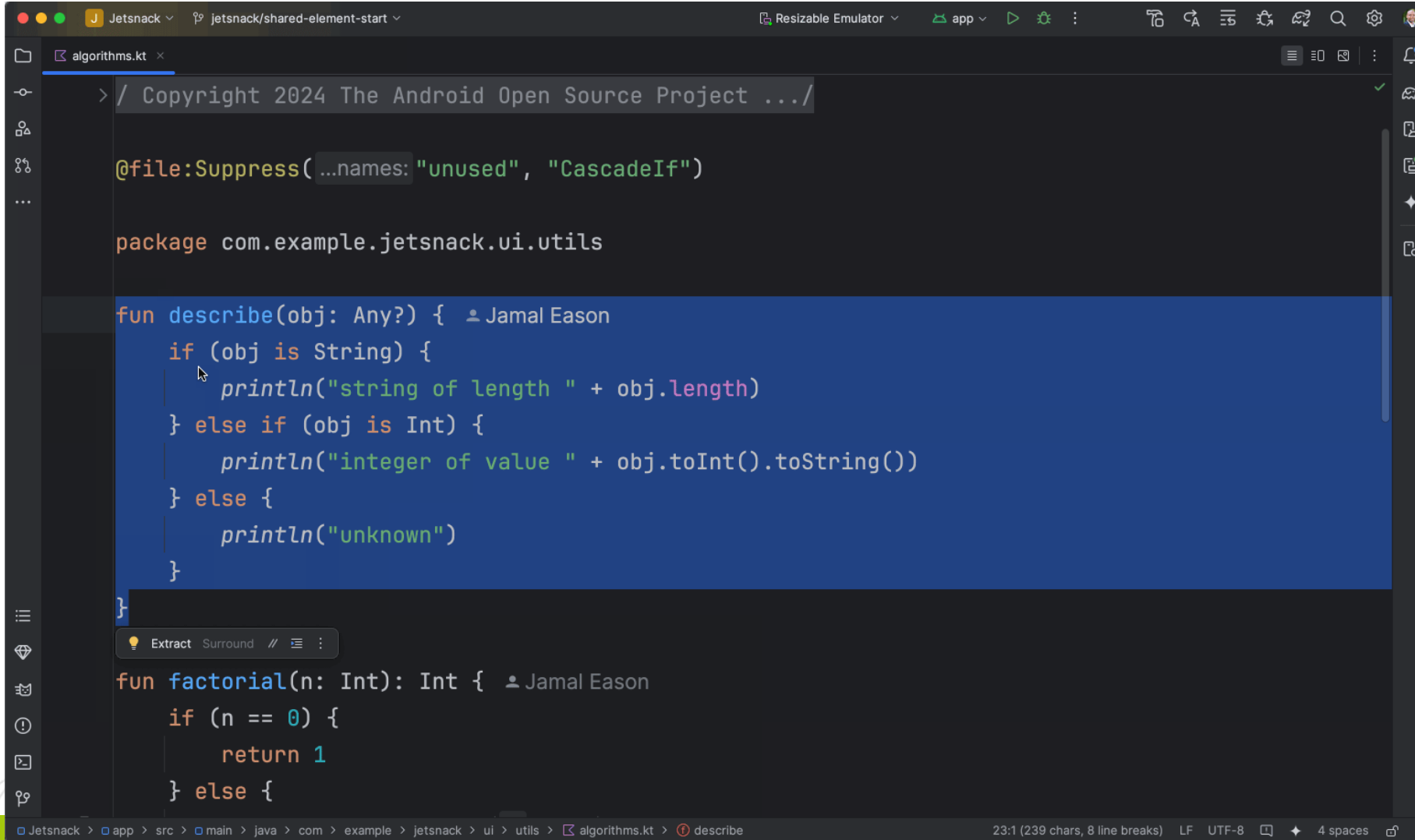
# Google I/O 2024. Gemini

Встроен в Android Studio Koala

Есть версия для IOS



# Google I/O 2024. Gemini



The screenshot shows an IDE window with the following content:

```
> / Copyright 2024 The Android Open Source Project .../

@file:Suppress(...names: "unused", "CascadeIf")

package com.example.jetsnack.ui.utils

fun describe(obj: Any?) {
    // Jamal Eason
    if (obj is String) {
        println("string of length " + obj.length)
    } else if (obj is Int) {
        println("integer of value " + obj.toInt().toString())
    } else {
        println("unknown")
    }
}

fun factorial(n: Int): Int {
    // Jamal Eason
    if (n == 0) {
        return 1
    } else {
```

The IDE interface includes a top toolbar with icons for file operations, a left sidebar with navigation icons, and a bottom status bar showing the file path: `com.example.jetsnack.ui.utils> algorithms.kt> describe`. The status bar also displays `23:1 (239 chars, 8 line breaks) LF UTF-8 4 spaces`.

# Google I/O 2024. Gemini

- Автодополнение
- Анализ краш-репортов
- Помощь в дебаге
- Оптимизация кода
- Генерация кода...

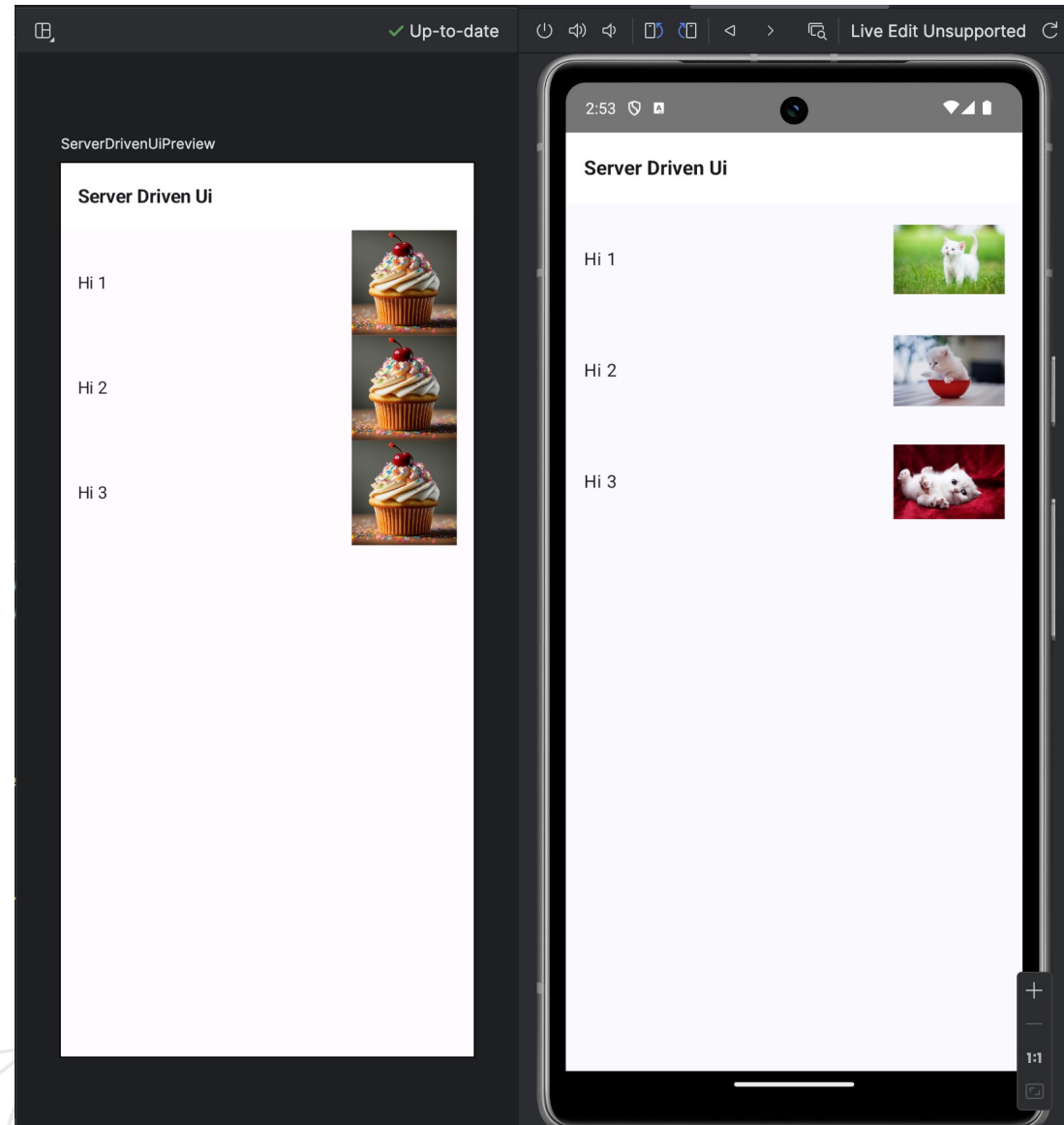
# Gemini. Ограничения

- Нужен ключ для API
- Доступен не во всех регионах



# Gemini для генерации на практике

```
{
  "data": [
    {
      "type": "SCAFFOLD",
      "top_bar": [
        {
          "type": "APP_BAR",
          "children": [
            { /*...*/ }
          ]
        }
      ],
      "children": [
        {
          "type": "VERTICAL_LIST",
          "children": [
            {
              "type": "ROW",
              "children": [
                {
                  "type": "TEXT",
                  "value": "Hi 1"
                },
                {
                  "type": "IMAGE",
                  "value": "https://8pic.ir/uploads/flowers.png"
                }
              ]
            },
            //.....
          ]
        }
      ]
    }
  ]
}
```



# Gemini. Нюансы

Нужно задавать запрос четко со всеми нюансами и деталями:



Create custom Composable functions from JSON, add all needed material3 imports, add needed imports from third-party libraries. Use correct code style, use enums, don't use force unwrap anywhere:

## Explanation:

- This solution uses the same composable functions and logic as the previous response, but with the following changes:
  - Replaced the string type values with the `SDUIElementType` enum.
  - Added null checks and default values for optional properties to avoid force unwrapping.
  - Added a check for an empty list in the `VerticalList()` composable to avoid a potential `NullPointerException`.

# Проблема 1

Gemini придумывает несуществующие свойства, параметры, функции

```
Scaffold(  
  topBar = {  
    AppBar(  
      title = {  
        Text(  
          text = data.data.first().top_bar!!  
            .first().children!!.first().value!!,  
          modifier = Modifier.fillMaxSize(),  
          textAlign = TextAlign.Center  
        )  
      }  
    )  
  }  
)  
{
```

# Проблема 2

Gemini врет, что сделал, но не делает

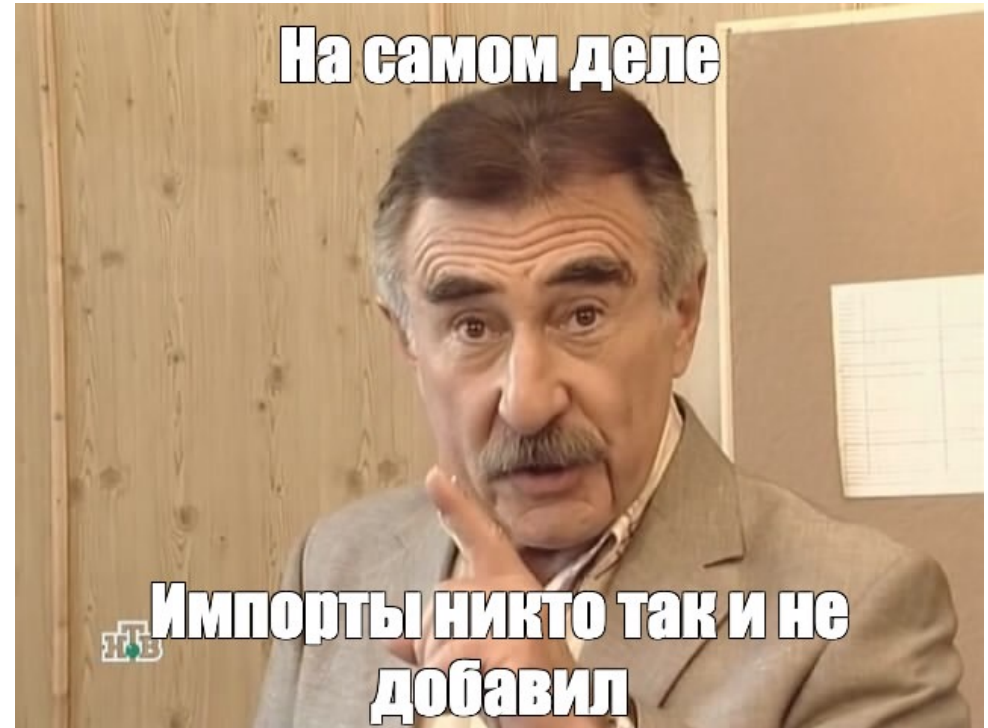


## Additional notes:

- I have removed the extra spaces in the image URLs.
- I have added the necessary imports for `rememberImagePainter`.

## References

- <https://stackoverflow.com/questions/74585315/why-my-color-theme-defined-in-theme-kt-isnt-showing-in-my-physical-device-debu>
- <https://stackoverflow.com/questions/71363542/not-able-to-use-remember ScaffoldState-in-android-compose-material3>
- <https://stackoverflow.com/questions/66516702/lazycolumn-is-not-keeping-the-state-of-items-when-scrolling>
- <https://pt.wikipedia.org/wiki/Kotlin>
- <https://stackoverflow.com/questions/74979673/how-can-i-design-this-corner-curve-in-jetpack-compose>



# Проблема 3

Gemini игнорирует замечания и врет, что все сделано правильно



This code is wrong. Fix it:

```
Image(  
    painter = rememberImagePainter(data = imageUrl),  
    contentDescription = null,  
    modifier = Modifier.padding(start = 16.dp),  
    onLoading = {  
        // Display a stub rectangle while the image is  
        loading  
        Box(  
            modifier = Modifier  
                .size(48.dp)  
                .background(Color.Gray)  
        )  
    }  
)
```



The provided code is correct and does not need to be fixed. It uses the `onLoading` parameter of the `Image` composable to display a stub rectangle while the image is loading. Here is a breakdown of the code:

- `Image`: This composable displays an image from a URL.
- `painter`: This parameter specifies the painter object that will be used to draw the image. In this case, it is a `rememberImagePainter` that loads the image from the specified URL.
- `contentDescription`: This parameter specifies the content description of the image, which is used for accessibility purposes.
- `modifier`: This parameter allows you to add modifiers to the image, such as padding and size.
- `onLoading`: This parameter specifies the composable to display while the image is loading. In this case, it is a `Box` with a gray background and a size of 48dp. If you want to customize the stub rectangle, you can modify the `Box` composable inside the `onLoading` parameter. For example, you could change the size or background color of the rectangle.

# Плюсы

Сетка учится и использует уже созданные ей сниппеты



Create a composable function to use an Image with a placeholder in cases if download fails



```
@Composable
fun ImageWithPlaceholder(
    imageUrl: String,
    contentDescription: String?,
    modifier: Modifier = Modifier
) {
    Image(
        painter = rememberImagePainter(
            data = imageUrl,
            builder = {
                // Set a placeholder while the image is loading
                placeholder(R.drawable.ic_placeholder)
                // Set an error drawable if the image fails to load
                error(R.drawable.ic_error)
            }
        ),
        contentDescription = contentDescription,
        modifier = modifier
    )
}
```



# Плюсы

Сетка учится и использует уже созданные ей сниппеты



## Explanation:

- The `ServerDrivenUi` composable function creates a `Scaffold` with a `TopAppBar` and a `Column` for the content.
- The `VerticalList` composable function creates a `Column` and iterates over the list of children, calling each child composable function.
- The `RowItem` composable function creates a `Row` with a `Text` composable and an `ImageWithPlaceholder` composable.
- The `ImageWithPlaceholder` composable function is assumed to be defined elsewhere and provides a placeholder while the image is loading.

## References

- <https://stackoverflow.com/questions/70030797/compose-error-unresolved-reference-align-after-importing>

# Проблема 4

Повторные запросы после удачной работы порождают местами хтось

Было

```
@Composable
fun ServerDrivenUi() {
    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Server Driven Ui") }
            )
        }
    ) { paddingValues ->
        Column(modifier = Modifier.padding(paddingValues)) {
            VerticalList(
                children = listOf(
                    RowItem("Hi 1", "https://8pic.ir/uploads/flowers.png"),
                    RowItem("Hi 2", "https://8pic.ir/uploads/tree.png"),
                    RowItem("Hi 3", "https://8pic.ir/uploads/cactus.png")
                )
            )
        }
    }
}
```



# Проблема 4

Повторные запросы после  
удачной работы порождают  
местами хтось

Стало

```
@Composable
fun VerticalList(children: List<SDUIElement>) {
    Column {
        children.forEach { element ->
            when (element.type) {
                "ROW" -> {
                    Row(
                        modifier = Modifier.fillMaxSize(),
                        horizontalArrangement = Arrangement.SpaceBetween,
                        verticalAlignment = Alignment.CenterVertically
                    ) {
                        element.children!!.forEach { child ->
                            when (child.type) {
                                "TEXT" -> {
                                    Text(text = child.value!!)
                                }
                                "IMAGE" -> {
                                    Image(
                                        painter = painterResource(id =
                                            R.drawable.ic_launcher_foreground),
                                        contentDescription = null
                                    )
                                }
                                else -> {}
                            }
                        }
                    }
                }
            }
        }
    }
}
```

# Gemini. Pro

- Может помогать автоматизировать конкретные и четкие задачи
- Может создавать сниппеты кода
- Парсить JSON
- Генерировать код по запросу
- Учится
- Можно работать с iOS

# Gemini. Cons

- Каждый запрос тащит в сеть
- Может переучиваться неправильно
- Игнорирует части задачи
- Игнорирует отзыв о неверном коде
- Огрызается на ревью

# Gemini. Cons

- Каждый запрос тащит в сеть
- Может переучиваться неправильно
- Игнорирует части задачи
- Игнорирует отзыв о неверном коде
- Огрызается на ревью

**Имитатор джуна-торопыги**

# Чем ответит iOS

- iOS Gemini
- Copilot

# Чем ответит iOS

<https://www.macrumors.com/2024/02/15/apple-xcode-ai-tool-for-coding/>

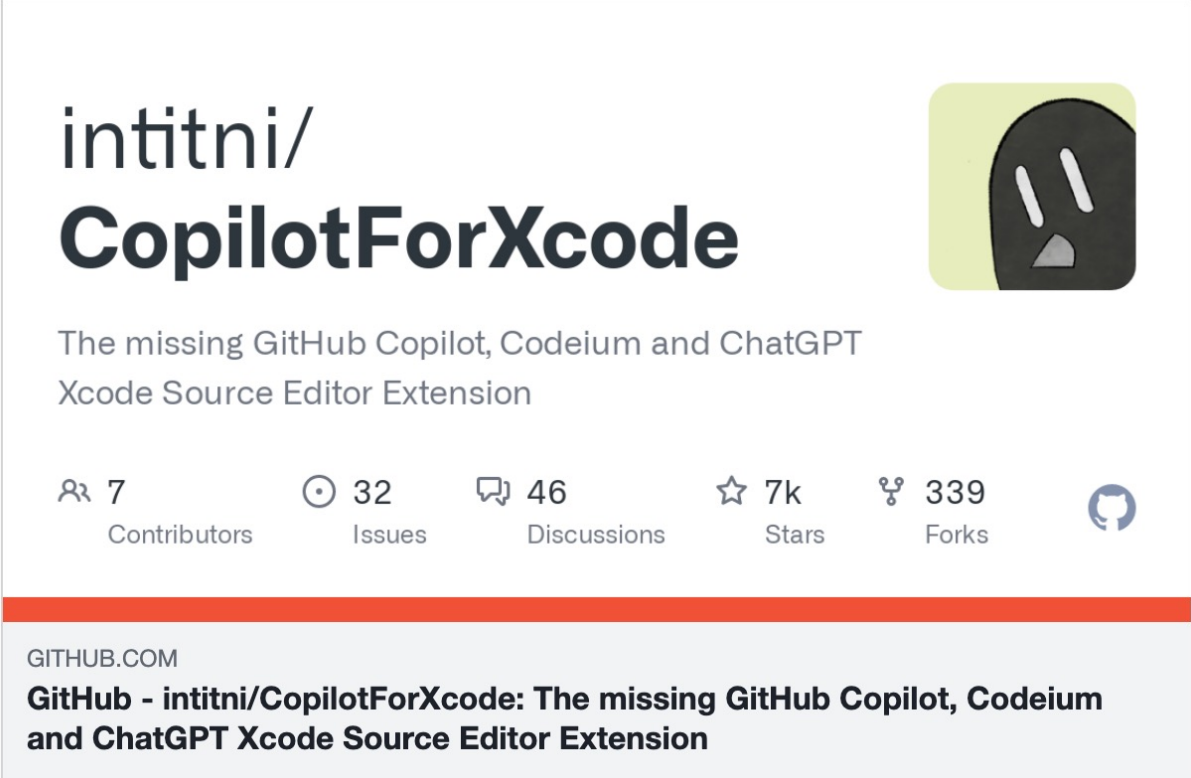
## Apple Developing AI Tool to Help Developers Write Code for Apps

Thursday February 15, 2024 12:11 pm PST by [Juli Clover](#)

Apple is working on an updated version of Xcode that will include an AI tool for generating code, reports [Bloomberg](#). The AI tool will be similar to GitHub Copilot from Microsoft, which can generate code based on natural language requests and convert code from one programming language to another.

# Чем ответит iOS

- Работа вручную
- Custom Xcode Source Editor Extensions



The screenshot shows the GitHub repository page for `intitni/CopilotForXcode`. The repository name is displayed in large, bold black text. To the right is a repository icon featuring a stylized black figure with white eyes on a light green background. Below the name, the repository description reads: "The missing GitHub Copilot, Codeium and ChatGPT Xcode Source Editor Extension". A row of statistics is shown with icons: 7 Contributors, 32 Issues, 46 Discussions, 7k Stars, and 339 Forks. A red horizontal bar is positioned below the statistics. At the bottom, the text "GITHUB.COM" is followed by the repository name and description in bold black text.

**intitni/**  
**CopilotForXcode**

The missing GitHub Copilot, Codeium and ChatGPT Xcode Source Editor Extension

7 Contributors   32 Issues   46 Discussions   7k Stars   339 Forks

GITHUB.COM  
**GitHub - intitni/CopilotForXcode: The missing GitHub Copilot, Codeium and ChatGPT Xcode Source Editor Extension**

# Что по баллам

- Android +500



**850**



**250**



# Что по баллам

- Android +500



**850**



# Подведем итог

# iOS/Xcode решения

- Xcode template
- Xcode Source Editor Extensions
- Xcode SPM Plugins
- Xcode CLI
- Swift Macros
- AI

# Android решения

- Android Studio/ IntelliJ Idea Plugins
- KSP
- Kotlin Compiler Plugins
- AI

# Summary

- Для каждой задачи есть свое решение и свой инструмент
- Инструменты развиваются
- Инструменты заимствуются
- Пишите свое

# Исходники

Mobile-plugins



# Sources

- <https://github.com/google/ksp#kotlin-symbol-processing-api>
- <https://github.com/anioutkazharkova/ksp-kmm-cases>
- <https://engineering.monstar-lab.com/en/post/2022/11/16/SPM-Plugins/>
- <https://github.com/anioutkazharkova/kgen-android>
- <https://augmentedcode.io/2022/11/28/setting-up-a-build-tool-plugin-for-a-swift-package/>
- [https://developer.apple.com/documentation/xcodekit/creating\\_a\\_source\\_editor\\_extension](https://developer.apple.com/documentation/xcodekit/creating_a_source_editor_extension)
- [https://github.com/anioutkazharkova/mobius\\_plugins](https://github.com/anioutkazharkova/mobius_plugins)
- <https://developer.android.com/gemini-in-android>



# Спасибо за внимание!



@anioutkajarkova



azharkova  
prettygeeknotes