auto.ru

Разработка CLI инструментов на Swift

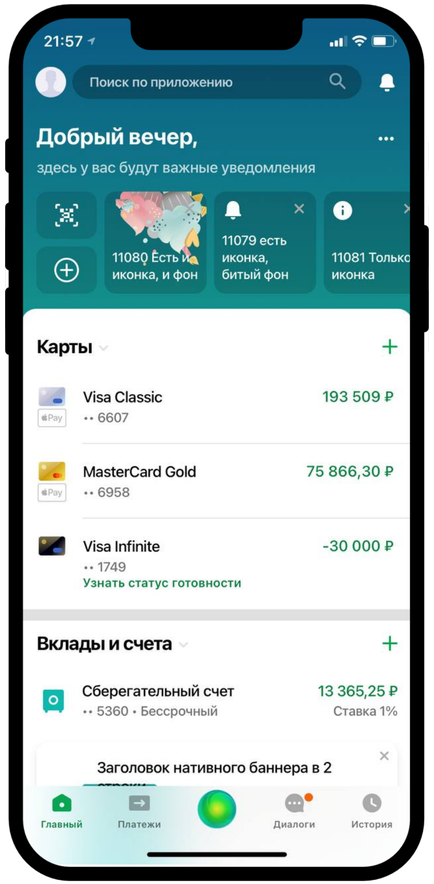
Игорь Шамрин
Яндекс Вертикали — Авто.ру

19 // 10 // 24

Мой опыт

Мой опыт

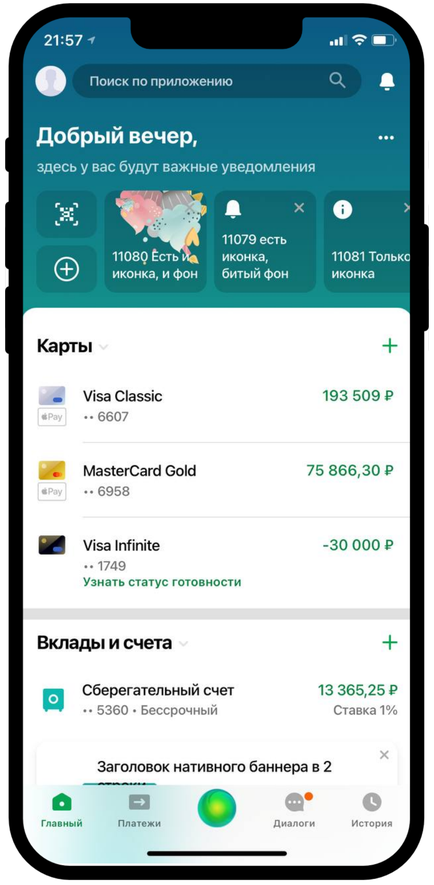
СБЕР



Сбербанк Онлайн

Мой опыт

СБЕР



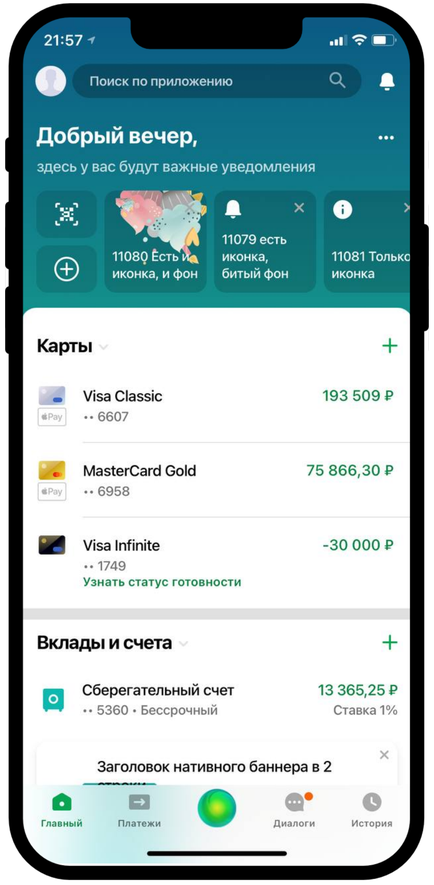
Сбербанк Онлайн

Яндекс

Авто.ру

Мой опыт

СБЕР

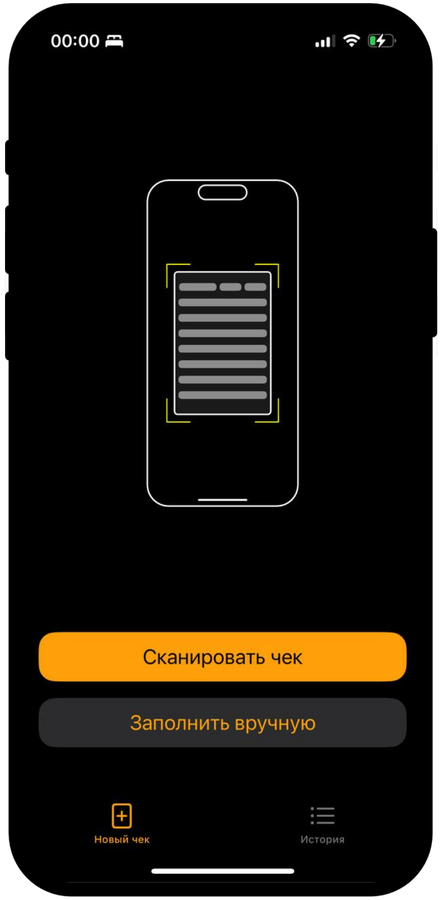


Сбербанк Онлайн

Яндекс

Авто.ру

Пет проекты



Монетки

Содержание доклада

CLI - Command line interface

Пример CLI

```
git commit -m “Hello Mobius!” --verbose
```

Пример CLI


git
commit
-m
“Hello Mobius!”
--verbose
Название программы
Команда
Аргумент
Значение аргумента
Флаг

Зачем CLI мобильному разработчику? 🤔

Зачем CLI мобильному разработчику? 🤔

01

Запуск
кода на CI

Зачем CLI мобильному разработчику? 🤔

01

Запуск
кода на CI

02

Инструменты
для разработки

01

auto.ru

Запуск кода на CI

CI: Build AdHoc

Steps

```
- tools issue-info --auth-token *my token*
- tools pr-info --auth-token *my token*
- tools xcode-archive --configuration Debug --colored-logs --path *путь до проекта*
...
- tools lint --config *путь до конфига*
- tools import-check --xcodebuild-log-path .ci_build/xcodebuild.log --path *путь до проекта*
- tools warnings-check --warnings-logs-file-path .ci_build/warnings.log --path *путь до проекта*
```

CI: Build AdHoc

Steps

```
- tools issue-info --auth-token *my token*
- tools pr-info --auth-token *my token*
- tools xcode-archive --configuration Debug --colored-logs --path *путь до проекта*
...
- tools lint --config *путь до конфига*
- tools import-check --xcodebuild-log-path .ci_build/xcodebuild.log --path *путь до проекта*
- tools warnings-check --warnings-logs-file-path .ci_build/warnings.log --path *путь до проекта*
```

CI: Build AdHoc

Steps

```
- tools init --path ... --auth-token ...
- tools issue-info --auth-token *my token*
- tools pr-info --auth-token *my token*
- tools xcode-archive --configuration Debug --colored-logs --path *путь до проекта*
...
- tools lint --config *путь до конфига*
- tools import-check --xcodebuild-log-path .ci_build/xcodebuild.log --path *путь до проекта*
- tools warnings-check --warnings-logs-file-path .ci_build/warnings.log --path *путь до проекта*
```

CI: Build AdHoc

Steps

```
- tools init --path ... --auth-token ...
- tools issue-info
- tools pr-info
- tools xcode-archive --configuration Debug --colored-logs
...
- tools lint --config *путь до конфига*
- tools import-check --xcodebuild-log-path .ci_build/xcodebuild.log
- tools warnings-check --warnings-logs-file-path .ci_build/warnings.log
```

Какой язык программирования выбрать?

Какой язык программирования выбрать?

Любой язык с готовым парсером аргументов

Какой язык программирования выбрать?

Какой язык программирования выбрать?

Какой язык программирования выбрать?






Что ещё хотим от языка для CI?

Что ещё хотим от языка для CI?

Знакомый всем разработчикам язык

Что ещё хотим от языка для CI?

Знакомый всем разработчикам язык

Что ещё хотим от языка для CI?

Знакомый всем разработчикам язык

API для взаимодействия с проектом

Что ещё хотим от языка для CI?

Знакомый всем разработчикам язык

API для взаимодействия с проектом

Что ещё хотим от языка для CI?

Знакомый всем разработчикам язык

API для взаимодействия с проектом

Библиотеки для сборки и анализа кода

Что ещё хотим от языка для CI?

Знакомый всем разработчикам язык

API для взаимодействия с проектом

Библиотеки для сборки и анализа кода

Что ещё хотим от языка для CI?

Знакомый всем разработчикам язык

API для взаимодействия с проектом

Библиотеки для сборки и анализа кода

Готовые инструменты для логирования

Что ещё хотим от языка для CI?

Знакомый всем разработчикам язык

API для взаимодействия с проектом

Библиотеки для сборки и анализа кода

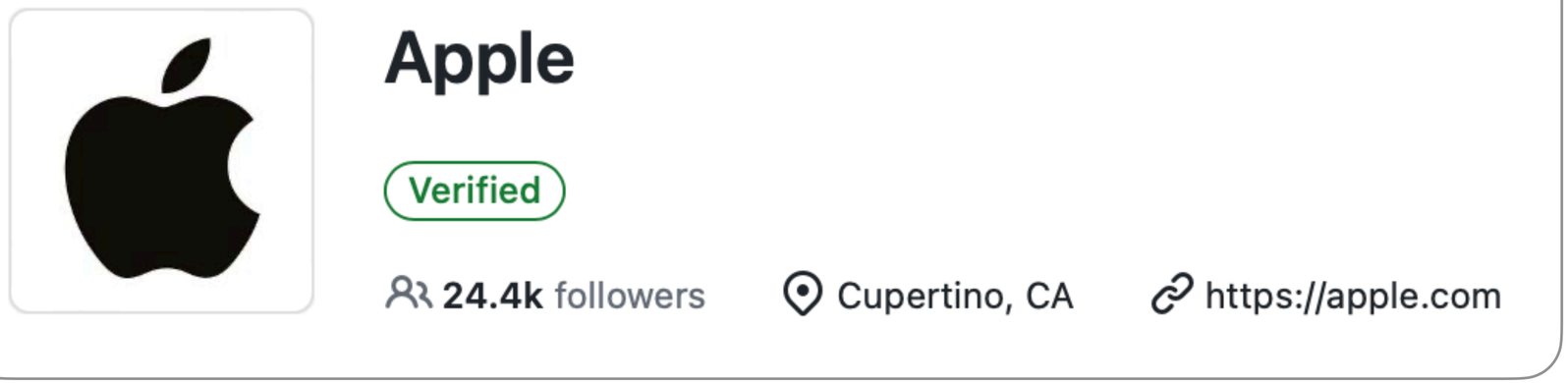
Готовые инструменты для логирования

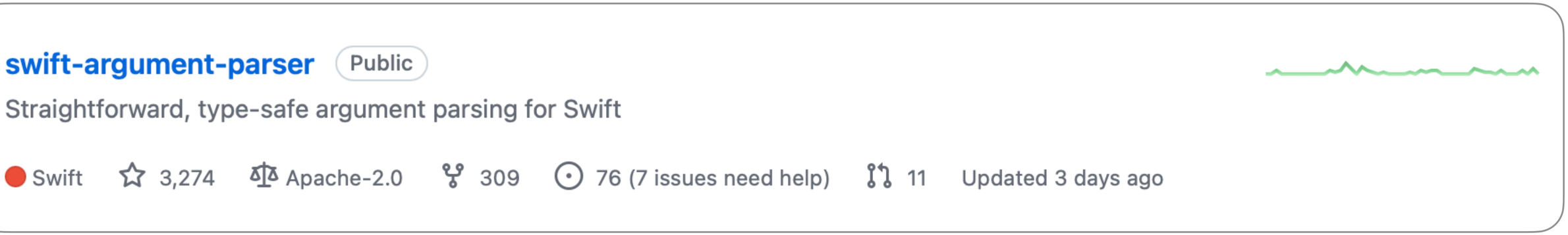
02

auto.ru

Swift Argument Parser

Swift Argument Parser





MyProgam hello --name “Mobius” --newLine

```
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )

    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
    if newLine {
        print("Hello \(name)”.replacingOccurrences(of: “ ”, with: “\n”)
    } else {
        print("Hello \(name)")
    }
}
```

MyProgam hello --name “Mobius” --newLine

```
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )

    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
    if newLine {
        print("Hello \(name)”.replacingOccurrences(of: “ ”, with: “\n”)
    } else {
        print("Hello \(name)")
    }
}
```

MyProgam hello --name “Mobius” --newLine

```
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )

    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
    if newLine {
        print("Hello \(name)”.replacingOccurrences(of: “ ”, with: “\n”)
    } else {
        print("Hello \(name)")
    }
}
```

MyProgam hello --name “Mobius” --newLine

```
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )

    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
    if newLine {
        print("Hello \(name)”.replacingOccurrences(of: “ ”, with: “\n”)
    } else {
        print("Hello \(name)")
    }
}
```

MyProgam hello --name “Mobius” --newLine

```swift
import ArgumentParser
import Foundation

@main
struct MyProgram: ParsableCommand {
    static let configuration = CommandConfiguration(
        subcommands: [HelloCommand.self]
    )
}

struct HelloCommand: ParsableCommand {
    static let configuration = CommandConfiguration(
        commandName: "hello",
        abstract: """
        Напечатать hello
        """
    )

    @Option(help: "Имя для вывода")
    var name: String

    @Flag(help: "Печатать каждое слово с новой строки")
    var newLine: Bool = false

    func run() throws {
    if newLine {
        print("Hello \(name)”.replacingOccurrences(of: “ ”, with: “\n”)
    } else {
        print("Hello \(name)")
    }
}
```

Terminal: ./MyProgram --help

```
USAGE: my-program <subcommand>

OPTIONS:
  -h, --help              Show help information.

SUBCOMMANDS:
  hello                   Напечатать hello
```

Terminal: ./MyProgram hello --help

```
OVERVIEW: Напечатать hello

USAGE: my-program hello --name <name> [--new-line]

OPTIONS:
  --name <name>           Имя для вывода
  --new-line              Печатать каждое слово с новой строки
  -h, --help              Show help information.
```

Пользователи

Swift Package Manager

SwiftLint

SwiftFormat

Tuist

Periphery

03

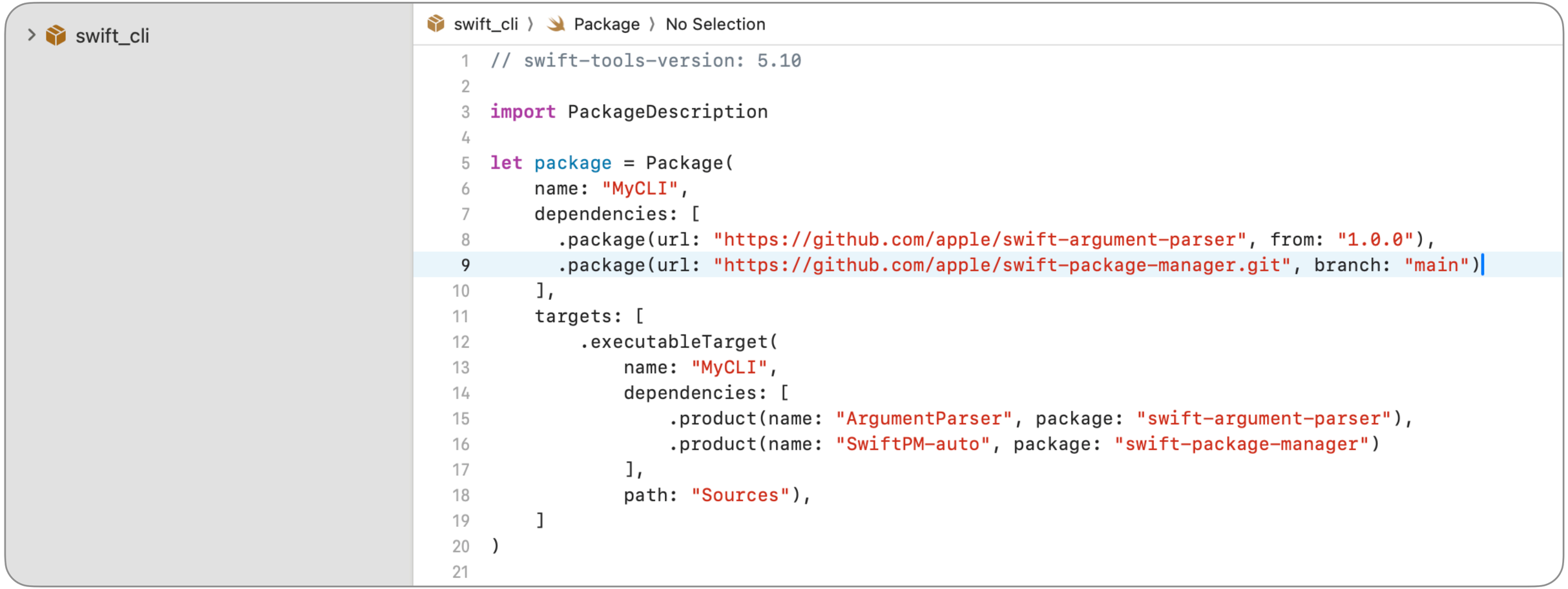
auto.ru

API для взаимодействия со структурой проекта

API для взаимодействия со структурой проекта

Найти все таргеты
тестов в SPM проекте

SwiftPM-auto



Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

Поиск тестов

```
import PackageModel

func findTests(packagePath: AbsolutePath) {
    let workspace = try Workspace(forRootPackage: packagePath)

    let package = try await workspace.loadRootPackage(
        at: rootPackagePath,
        observabilityScope: observability.topScope
    )

    for target in package.targets where target.type == .test {
        result.testTargets.append(target)
    }

    for dependency in package.manifest.dependencies {
        guard case let .fileSystem(fsDependency) = dependency else {
            return
        }

        findTest(packagePath: fsDependency.path)
    }
}
```

04

auto.ru

Анализ кода проекта

Статический анализ кода

Статический анализ кода

Проверить, что в Feature пакетах не создаются экземпляры класса Date

Статический анализ кода

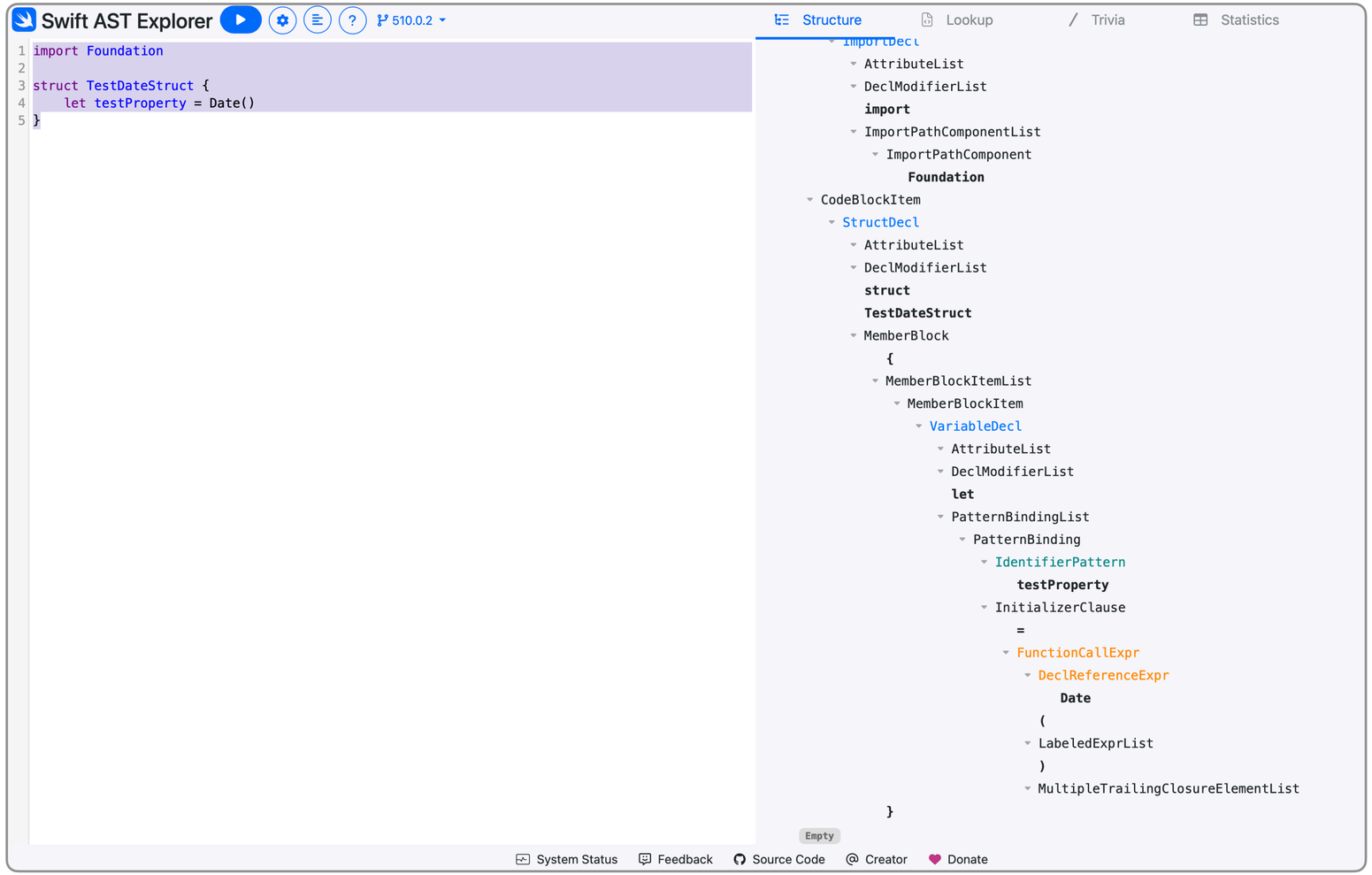
Проверить, что в Feature пакетах не создаются экземпляры класса Date

Исключение:
SwiftUI Preview

SwiftSyntax

Swift-syntax — это набор библиотек, которые работают с древовидным представлением исходного кода Swift с точностью до исходного кода, называемым SwiftSyntax tree

AST Explorer



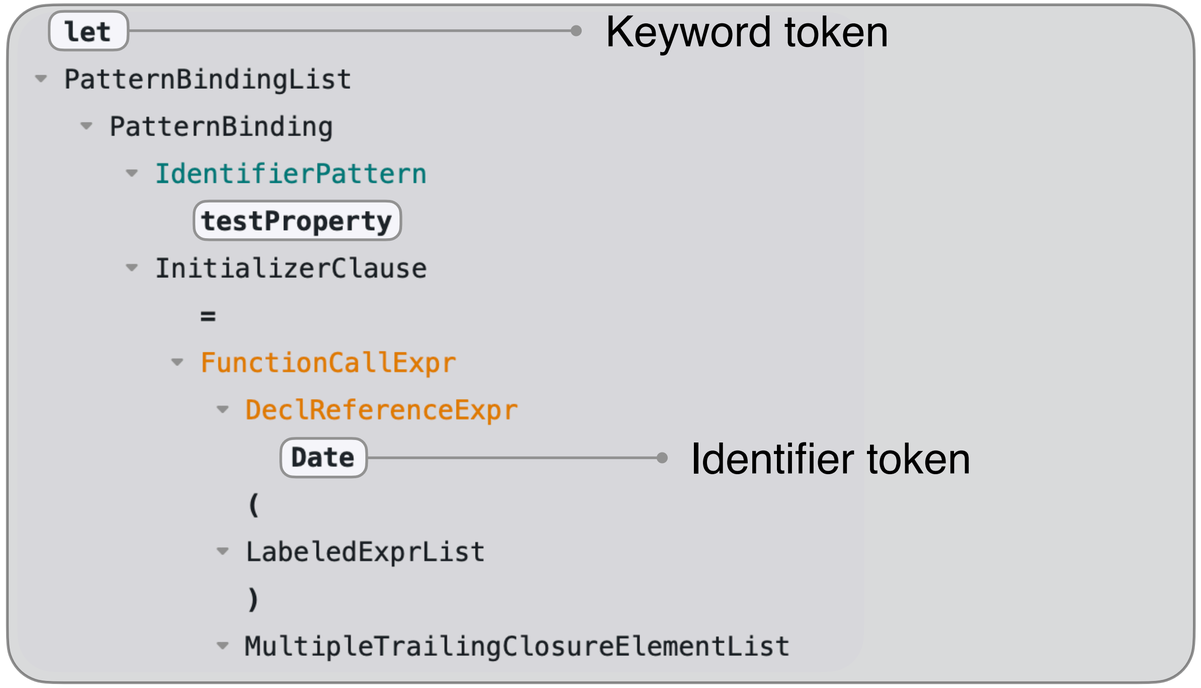
AST Explorer

```
struct Test {
    let testProperty = Date()
}
```

AST Explorer

```
struct Test {
    let testProperty = Date()
}
```

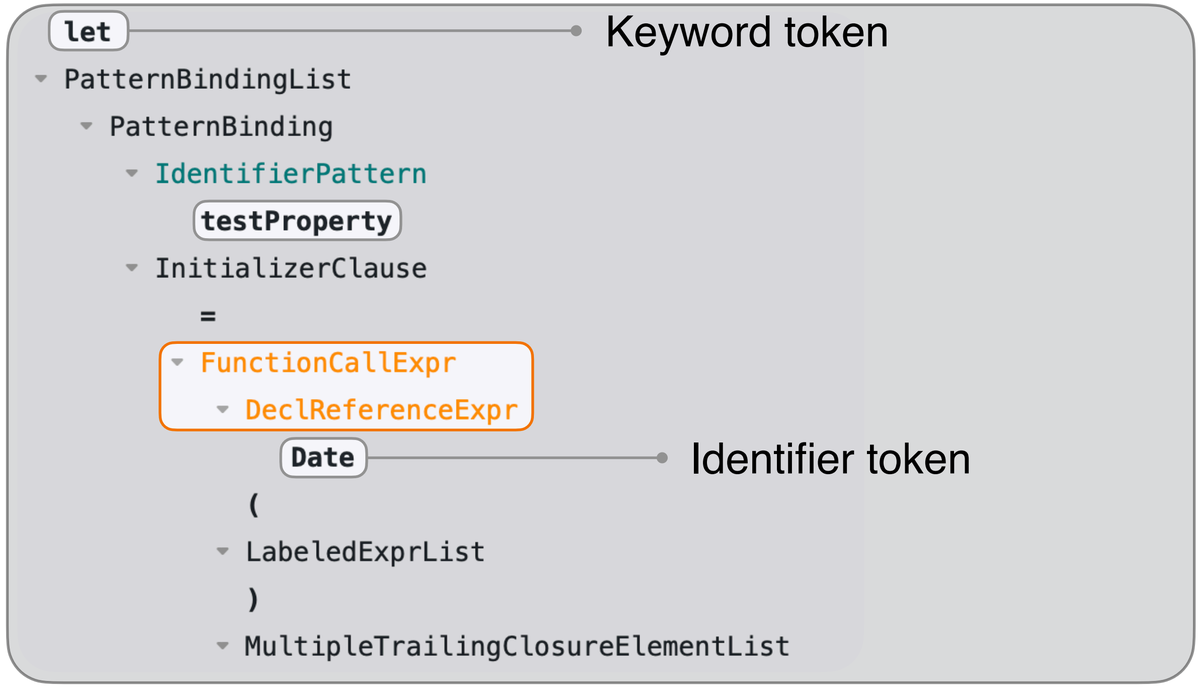
TokenSyntax



AST Explorer

```
struct Test {
    let testProperty = Date()
}
```

TokenSyntax ExprSyntax



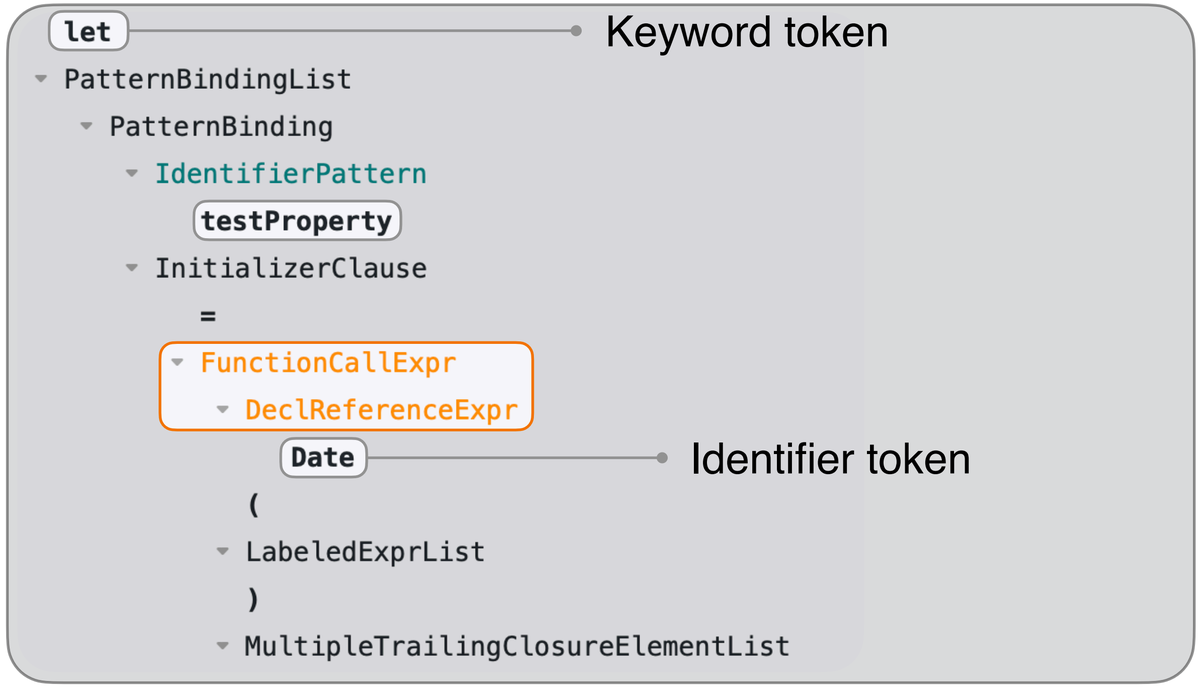
AST Explorer

```swift
import SwiftSyntax

static func isDateInit(token: TokenSyntax) -> Bool {
  if token.tokenKind == .identifier("Date"),
    let declReferenceExpr = token.parent?.as(DeclReferenceExprSyntax.self),
    let functionCallExpr = declReferenceExpr.parent?.as(FunctionCallExprSyntax.self),
    functionCallExpr.arguments.isEmpty { return true }

    return false
}
```

TokenSyntax ExprSyntax



AST Explorer

```
import SwiftSyntax

static func isDateInit(token: TokenSyntax) -> Bool {
  if token.tokenKind == .identifier(“Date"),
    let declReferenceExpr = token.parent?.as(DeclReferenceExprSyntax.self),
    let functionCallExpr = declReferenceExpr.parent?.as(FunctionCallExprSyntax.self),
    functionCallExpr.arguments.isEmpty { return true }

    return false
}
```

Поиск Date init: preview check

```
struct MyView_Previews: PreviewProvider {
    static var previews: some View {
        MyView(date: Date())
    }
}
```

Поиск Date init: preview check

```
struct MyView_Previews: PreviewProvider {
    static var previews: some View {
        MyView(date: Date())
    }
}
```

Поиск Date init: preview check

```
#Preview {
    MyView(date: Date())
}
```

Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
           ...
                // Previews
                if let structDecl = parentToken.as(StructDeclSyntax.self),
                   let inheritedTypeList = structDecl.inheritanceClause?.inheritedTypes.as(InheritedTypeListSyntax.self),
                   inheritedTypeList.contains(where: { inheritedTypeSyntax in
                       let identifierTypeName = inheritedTypeSyntax.type.as(IdentifierTypeSyntax.self)?.name.text
                       return identifierTypeName == "PreviewProvider"
                   })
                {
                    continue sourceTokensLoop
                }
            }

            let converter = SourceLocationConverter(fileName: filePath, tree: tree)
            resultLocations.append(converter.location(for: token.position))
        }
    }

    return resultLocations
}
```

Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
           ...
                // Previews
                if let structDecl = parentToken.as(StructDeclSyntax.self),
                   let inheritedTypeList = structDecl.inheritanceClause?.inheritedTypes.as(InheritedTypeListSyntax.self),
                   inheritedTypeList.contains(where: { inheritedTypeSyntax in
                       let identifierTypeName = inheritedTypeSyntax.type.as(IdentifierTypeSyntax.self)?.name.text
                       return identifierTypeName == "PreviewProvider"
                   })
                {
                    continue sourceTokensLoop
                }
            }

            let converter = SourceLocationConverter(fileName: filePath, tree: tree)
            resultLocations.append(converter.location(for: token.position))
        }
    }

    return resultLocations
}
```

Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
           ...
                // Previews
                if let structDecl = parentToken.as(StructDeclSyntax.self),
                   let inheritedTypeList = structDecl.inheritanceClause?.inheritedTypes.as(InheritedTypeListSyntax.self),
                   inheritedTypeList.contains(where: { inheritedTypeSyntax in
                       let identifierTypeName = inheritedTypeSyntax.type.as(IdentifierTypeSyntax.self)?.name.text
                       return identifierTypeName == "PreviewProvider"
                   })
                {
                    continue sourceTokensLoop
                }
            }

            let converter = SourceLocationConverter(fileName: filePath, tree: tree)
            resultLocations.append(converter.location(for: token.position))
        }
    }

    return resultLocations
}
```

Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
           ...
                // Previews marco
                if let macroDecl = parentToken.as(MacroExpansionExprSyntax.self), macroDecl.macroName.text == “Preview" {
                    continue sourceTokensLoop
                }


              ...
              ...
              ...
              ...
            }

            let converter = SourceLocationConverter(fileName: filePath, tree: tree)
            resultLocations.append(converter.location(for: token.position))
        }
    }

    return resultLocations
}
```

Поиск Date init: preview check

```
static func findDateUsage(filePath: String) throws -> [SourceLocation] {
    let tree = try Parser.parse(source: String(contentsOfFile: filePath))

    var resultLocations = [SourceLocation]()

    sourceTokensLoop: for token in tree.tokens(viewMode: .all) {
        if isDateInit(token: token) {
            for parentToken in sequence(first: token._syntaxNode, next: \.parent).dropFirst() {
           ...
                // Previews marco
                if let macroDecl = parentToken.as(MacroExpansionExprSyntax.self), macroDecl.macroName.text == “Preview" {
                    continue sourceTokensLoop
                }


              ...
              ...
              ...
              ...
            }

            let converter = SourceLocationConverter(fileName: filePath, tree: tree)
            resultLocations.append(converter.location(for: token.position))
        }
    }

    return resultLocations
}
```

Date lint command

```
23:28:39 error: Использование Date вместо UtcClock: Features/ReportDetails/Sources/ReportDetails/ReportDetailsView.swift:46:16
23:28:39 error: Lint failed with errors:
23:28:39 error: The operation couldn’t be completed. (LintCheck.DateLintCheckCommand.DateLintError error 0.)
Error: LintCheckError()
```

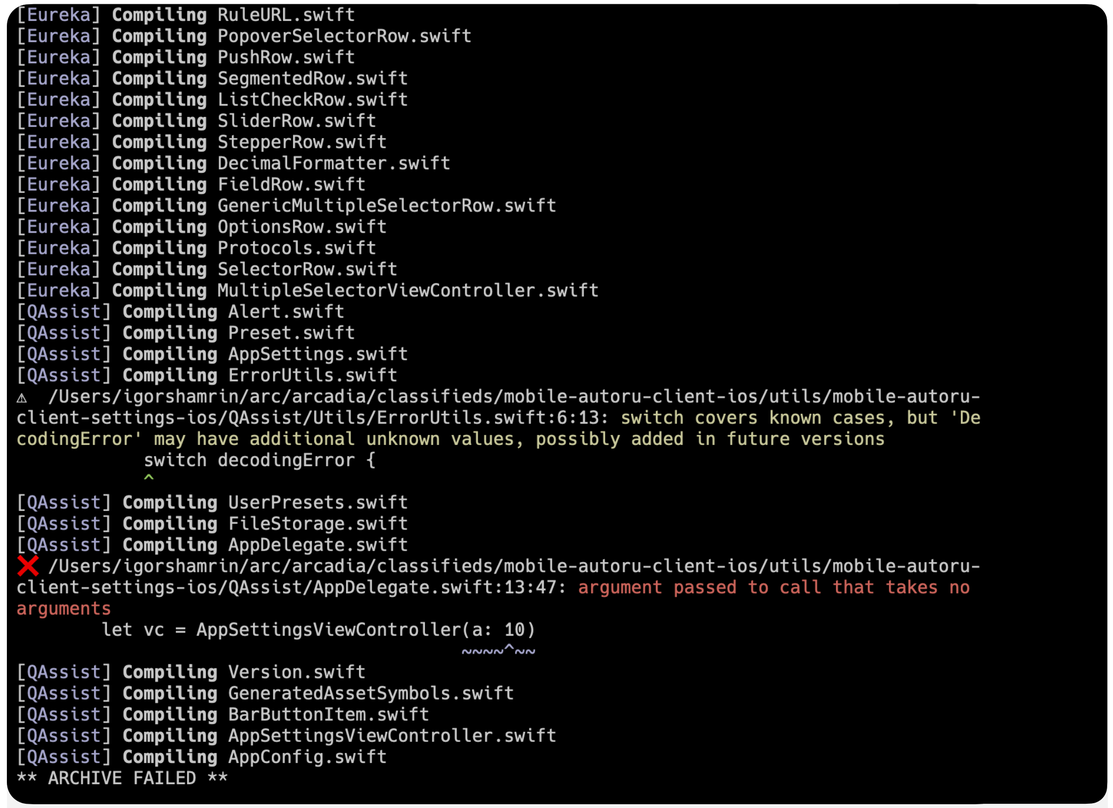
04

auto.ru

Проверка новых warning-ов на CI

Парсим логи с Xcbeatify

Стандартный xcarchvive (~ 335МБ)



Отформатированный xcarchive (< 1МБ)

Парсим логи с Xcbeatify

```
import XcbeautifyLib

func onWriteLine(_ line: String, fromErrorStream: Bool) {
    ...

    if let formattedLine = formatNextLine() {
         outputHandler.write(parser.outputType, formattedLine)

         if parser.outputType == .warning {
             writeWarning(formattedLine) // <-- Записываем warning в warnings.log
         }
     }
}
```

Парсим логи с Xcbeatify

```
import XcbeautifyLib

func onWriteLine(_ line: String, fromErrorStream: Bool) {
    ...

    if let formattedLine = formatNextLine() {
        outputHandler.write(parser.outputType, formattedLine)

        if parser.outputType == .warning {
            writeWarning(formattedLine) // <-- Записываем warning в warnings.log
        }
    }
}
```

warnings.log

```
{
  /Contracts/Sources/ComplaintContracts/ComplaintService.swift:10:14: : [ 1 item
    0 : main actor-isolated property 'service' can not be mutated from a non-isolated context; this is an error in Swift 6
            self.service = service
                 ^
  ]
  /Contracts/Sources/SplitContracts/SplitService.swift:38:9: : [ 1 item
    0 : main actor-isolated property 'objectWillChange' can not be mutated from a non-isolated context; this is an error in Swift 6
            objectWillChange = service.erasedObjectWillChange()
            ^
  ]
  /Services/PhoneConfirmationServices/Sources/PhoneConfirmationFlowImpl.swift:20:14: : [ 1 item ]
  /packages/user_profile/main/UserProfileInteractor.swift:558:13: : [ 1 item ]
  /Shared/Views/Sources/Views/SwipeableView.swift:71:17: : [ 1 item ]
  /Contracts/Sources/PublicProfileContracts/PublicProfileInfoService.swift:12:9: : [ 1 item ]
  /Features/UGC/Sources/UGCUserProfile/UGCUserProfileViewModel.swift:107:14: : [ 1 item ]
  /Features/NotificationBell/Sources/NotificationBell/NotificationBellItemFactory.swift:27:14: : [ 1 item ]
```

Проверка новых warning-ов на CI

Обновляем состояние по таймеру

Проверка новых warning-ов на CI

Обновляем состояние по таймеру

Другие проверки

SwiftFormat

Lint & Format

Periphery, SwiftPM

Unused code check

SourceKitten

Import check

SwiftPM

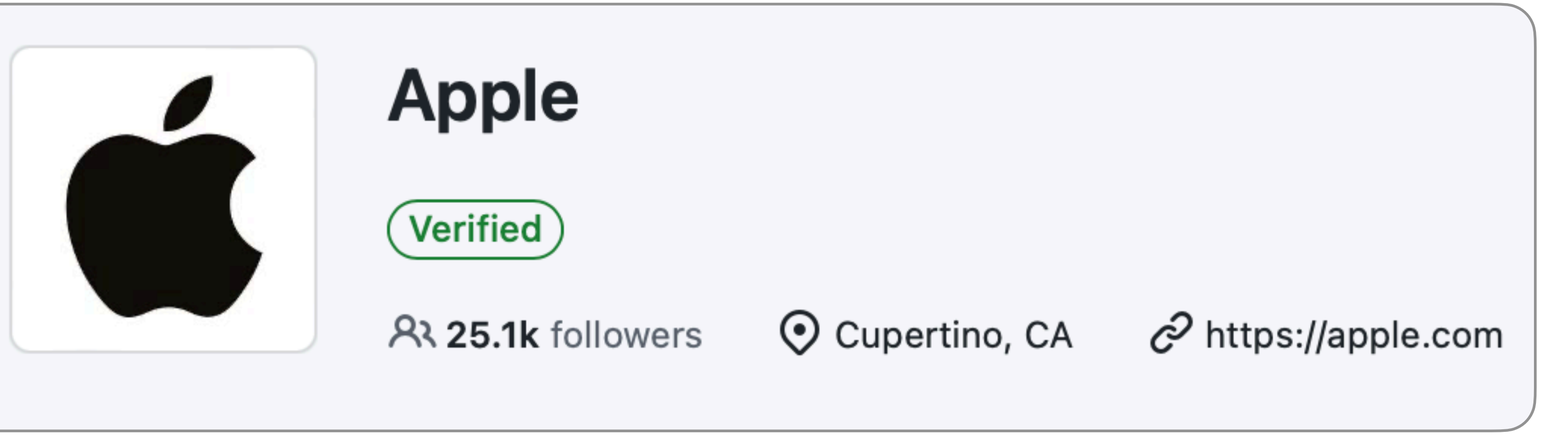
Dependency check

05

auto.ru

Собираем логи красиво

Swift-log


Apple
Verified
25.1k followers
Cupertino, CA
https://apple.com


swift-log
Public
A Logging API for Swift
logging
swift-server
Swift
Apache License 2.0
287
3.5k
24
6
Updated last week

Swift-log

```
let logger = Logger(label: “MyCommand")


logger.info("response: \(string)")

logger.error("xcodebuild run failed, exit code = \(status)")

logger.warning("issueKey is nil")
```

Swift-log

```
extension LoggingSystem {
    static func setUp() {
        bootstrap { label in
            var handlers: [any LogHandler] = [LogHandlerImpl(label: label)]
            do {
                if let logsDirectory = ProcessInfo.processInfo.environment["LOGS_DIRECTORY"] {
                    try handlers.append(FileLogger(directoryPath: logsDirectory, fileName: label))
                }
            } catch { ... }

            return MultiplexLogHandler(handlers)
        }
    }
}
```

Html страничка с логами

☑️ Build and PR checks

📁 Common logs (stdout + stderr) log

▶ **XcodeBuild.log**
▼ **ImportCheckCommand.log**
```
Start checking packages/stock_card/main/StockCardViewController.swift
```
▶ **WarningsCheckCommand.log**

❌ Errors logs (stderr) log

▼ **WarningsCheckCommand_errors.log**
```
Появились новые warning-и
Список warning-ов, отличающихся от trunk:
[!] /packages/stock_card/main/StockCardViewController.swift:170:16: variable 'b' was never used; consider replacing with '_' or removing it
var a, b: Int
^
[!] /packages/stock_card/main/StockCardViewController.swift:170:13: variable 'a' was never used; consider replacing with '_' or removing it
var a, b: Int
^
```

Html страничка с логами

Ansi2Html Public

A Swift library to convert text with ANSI escape codes to HTML

C++ ☆ 1

06

auto.ru

Нюансы сборки и запуска

Запуск

👎

swift run

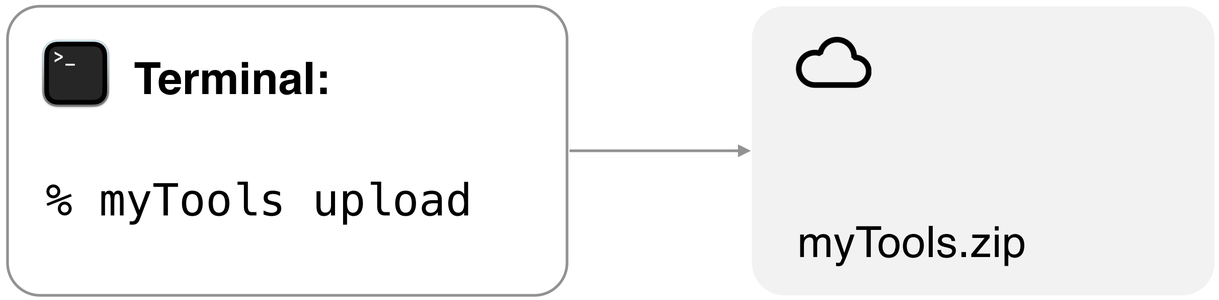
👎

Предсобранный
бинарник
в репозитории

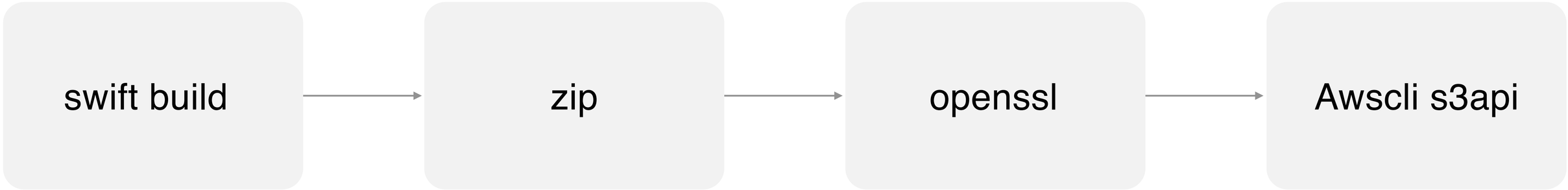
👍

Предсобранный
бинарник в облаке

Upload



Upload



.runnerEnv

```
EXECUTABLE = AutoRuBuildTools
URL[arm64] =
CHECKSUM[arm64] =
URL[x86_64] =
CHECKSUM[x86_64] =
```

Runner

.runnerEnv

EXECUTABLE = AutoRuBuildTools
URL[arm64] =
CHECKSUM[arm64] =
URL[x86_64] =
CHECKSUM[x86_64] =

Runner

.runnerCache

myTools_v1

myTools_v2

myTools_v3

07

auto.ru

Build tools в Авто.ру

CI: Build AdHoc

Steps

```
- tools issue-info --auth-token *my token*
- tools pr-info --auth-token *my token*
- tools xcode-archive --configuration Debug --colored-logs --path *путь до проекта*
...
- tools lint --config *путь до конфига*
- tools import-check --xcodebuild-log-path $PWD/.ci_build/xcodebuild.log --path *путь до проекта*
- tools warnings-check --warnings-logs-file-path .ci_build/warnings.log --path *путь до проекта*
```

CI: Build AdHoc

Steps

```
- tools issue-info --auth-token *my token*
- tools pr-info --auth-token *my token*
- tools xcode-archive --configuration Debug --colored-logs --path *путь до проекта*
...
- tools lint --config *путь до конфига*
- tools import-check --xcodebuild-log-path $PWD/.ci_build/xcodebuild.log --path *путь до проекта*
- tools warnings-check --warnings-logs-file-path .ci_build/warnings.log --path *путь до проекта*
```

CI: Build AdHoc

Steps

```
- tools init --path ... --auth-token ...
- tools issue-info --auth-token *my token*
- tools pr-info --auth-token *my token*
- tools xcode-archive --configuration Debug --colored-logs --path *путь до проекта*
...
- tools lint --config *путь до конфига*
- tools import-check --xcodebuild-log-path $PWD/.ci_build/xcodebuild.log --path *путь до проекта*
- tools warnings-check --warnings-logs-file-path .ci_build/warnings.log --path *путь до проекта*
```

CI: Build AdHoc

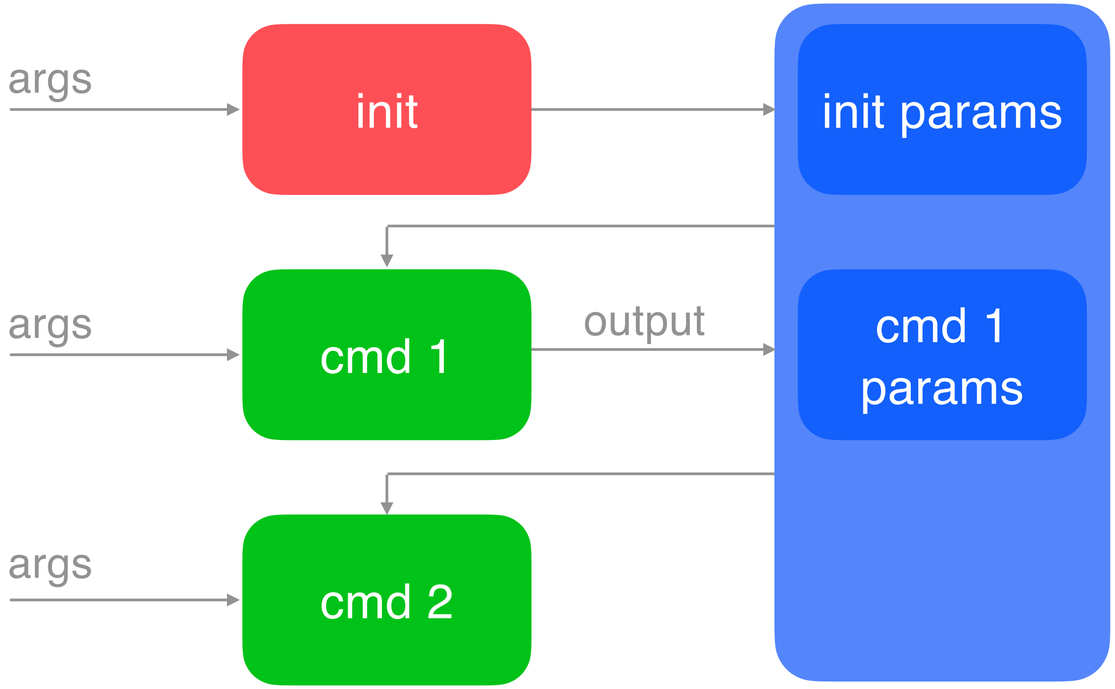
Steps

```
- tools init --path ... --auth-token ...
- tools issue-info
- tools pr-info
- tools xcode-archive --configuration Debug --colored-logs
...
- tools lint --config *путь до конфига*
- tools import-check --xcodebuild-log-path $PWD/.ci_build/xcodebuild.log
- tools warnings-check --warnings-logs-file-path .ci_build/warnings.log
```

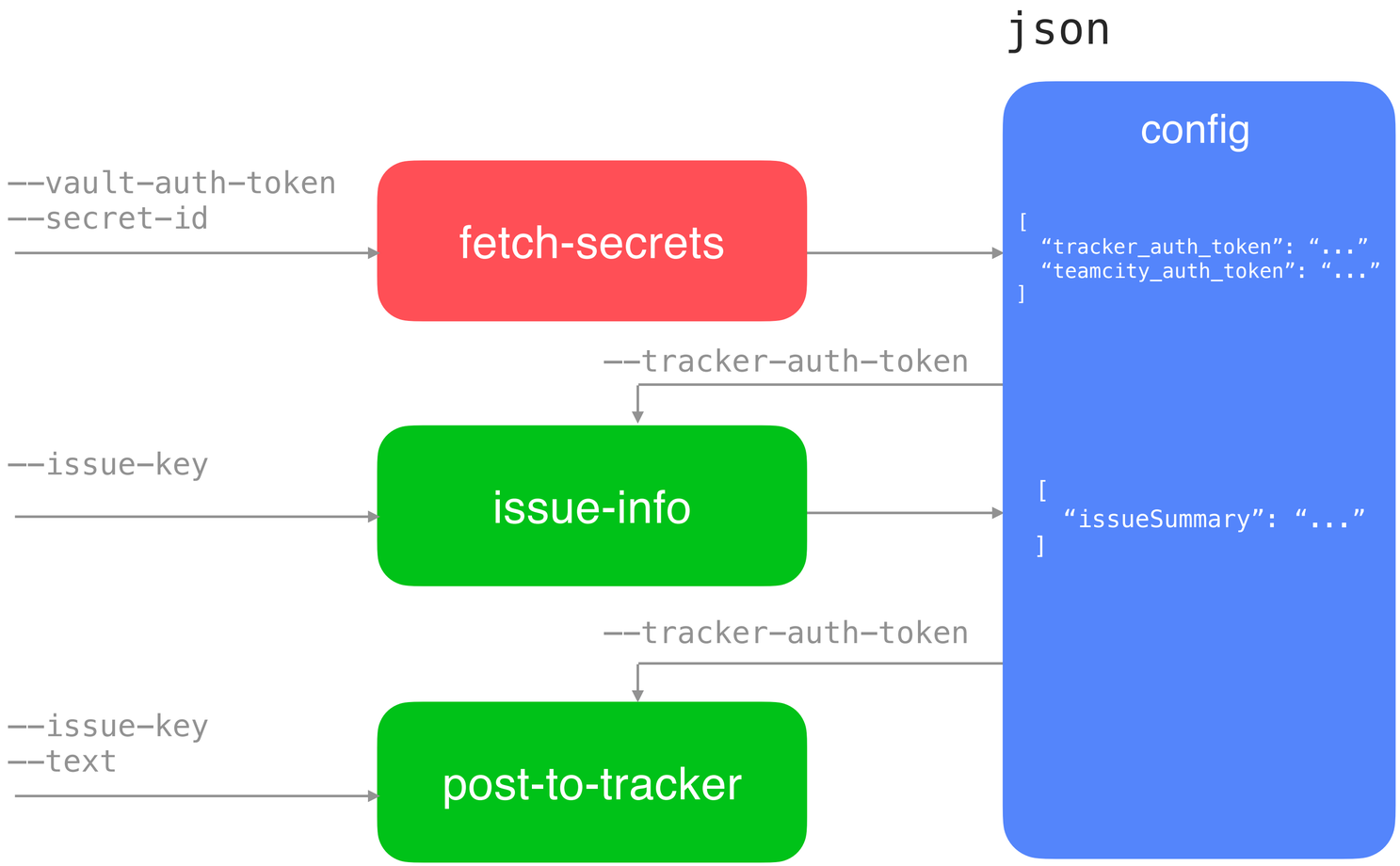
Tools init

a.yaml

```
- tools init
--root-path "$PWD"
--build-number "$BUILD_NUMBER"
--project-path "$PWD/AutoRu.xcodeproj"
--workspace-path "$PWD/AutoRu.xcworkspace"
--scheme AutoRu
--derived-data-path "$PWD/.ci_build/derived_data"
--profiles-path "$PWD/Profiles/adhoc"
--archive-path "$PWD/.ci_build/AutoRu.xcarchive"
--export-path "$PWD/.ci_build/export"
--branch "$BUILD_BRANCH"
```



Command output



Command output

```
public protocol AsyncParsableCommandWithOutput: AsyncParsableCommand {
    associatedtype Output

    mutating func run() async throws -> Output
}
```

CommonBuildTools

Часть общих команд

xcode-archive

Собирает архив проекта

xcode-test

Запускает все или указанные тесты в схеме

upload-build-to-tf

Загружает .ipa в TestFlight

post-to-telegram

Постит сообщение в указанный Telegram чат/канал

upload-symbols-to-appmetrica

Загружает dsym-ы в AppMetrica

upload-file-to-s3

Загружает произвольный файл на s3

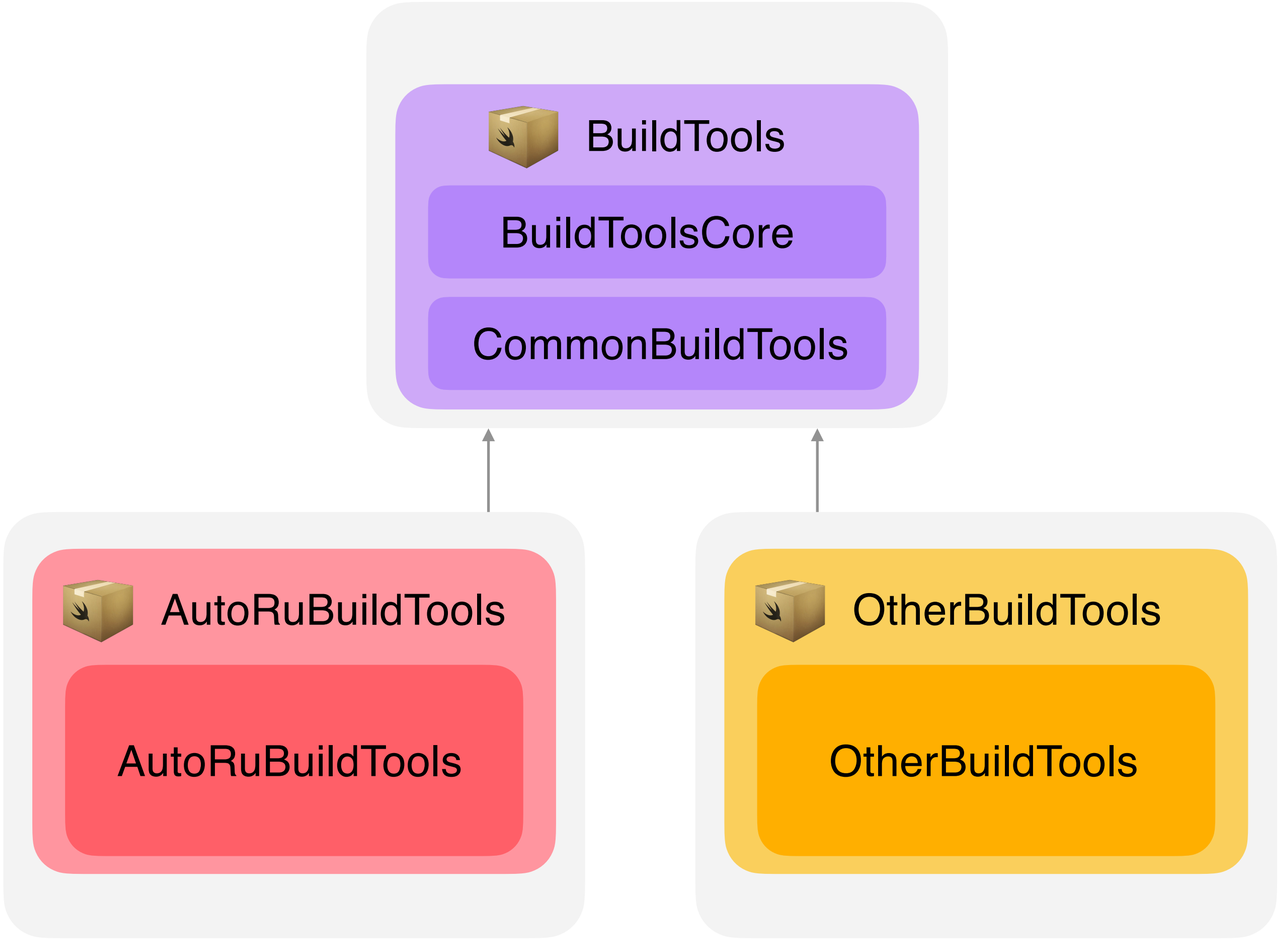
destination

Находит симулятор с указанными параметрами, или создает его, если не найден

update-xcconfig-value

Обновляет значение в xcconfig по ключу

Build tools




```swift
public protocol CommandGroup {
    @CommandsBuilder
    static var commands: Commands { get }
}
```

```swift
import BuildToolsCore

@main
struct RootCommandGroup: CommandGroup {
    static var commands: Commands {
        CommonBuildTools.self,
        MyProjectCommand.self,
    }
}
```

Выводы

01

Открывается много новых возможностей для взаимодействия с кодом проекта и его структурой

02

Удобный API для разработки CLI интерфейса

03

Лёгкий онбординг для iOS разработчиков

auto.ru

Спасибо за внимание!

@ShamrinIgor