Продукт — это не только приложение

Опыт разработки iOS framework как самостоятельного продукта

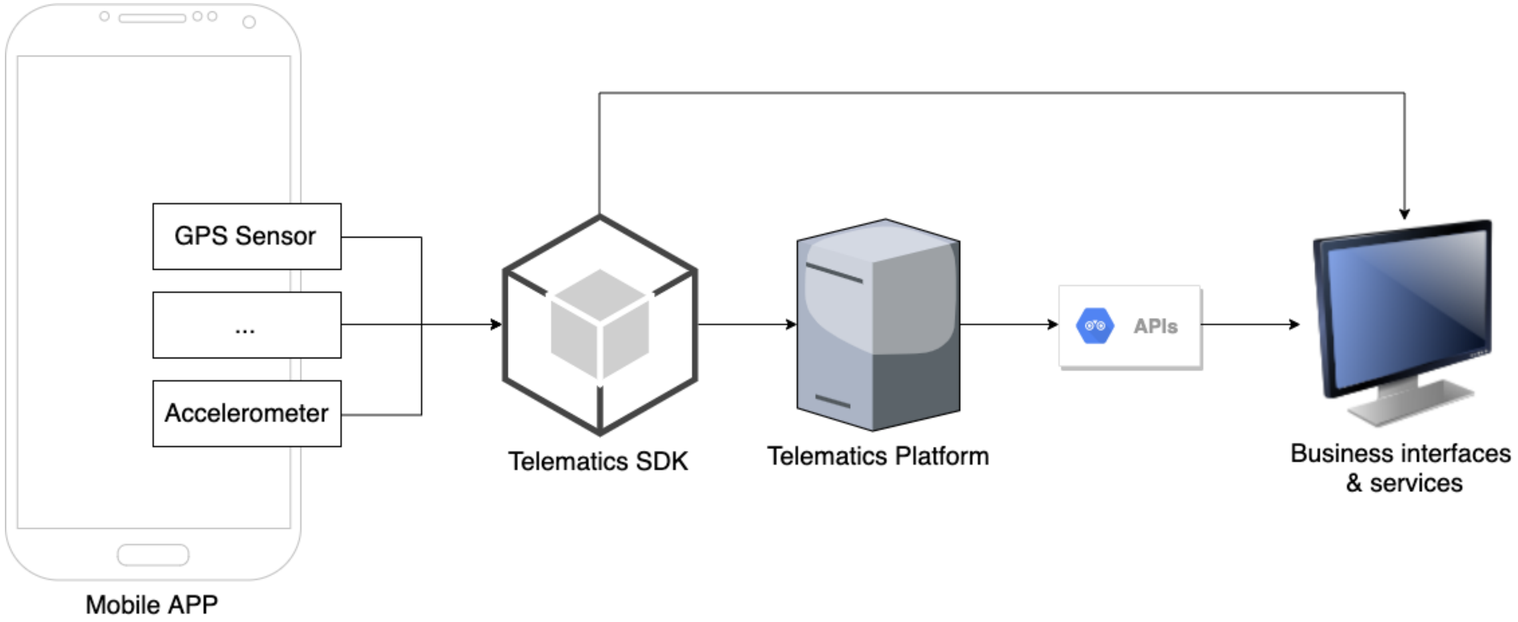
Игорь Набоков / iOS Developer

О чем доклад

- почему такая тема
- с чего я начал
- с какими проблемами столкнулся
- как эти проблемы решал
- пример публикации фреймворка

Почему

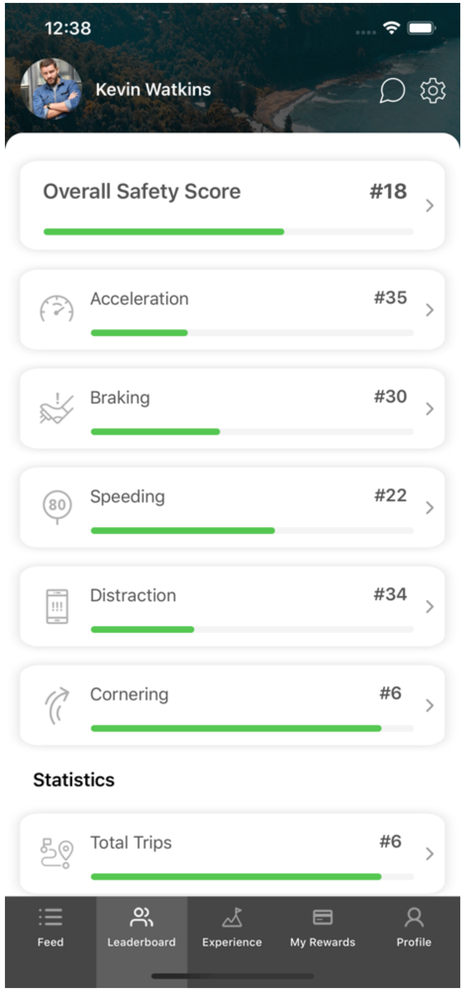
Что за библиотека телематики

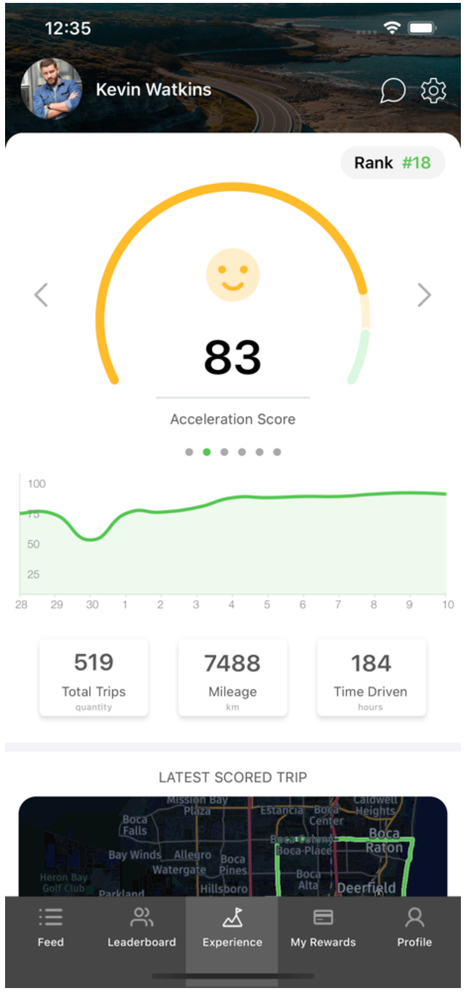


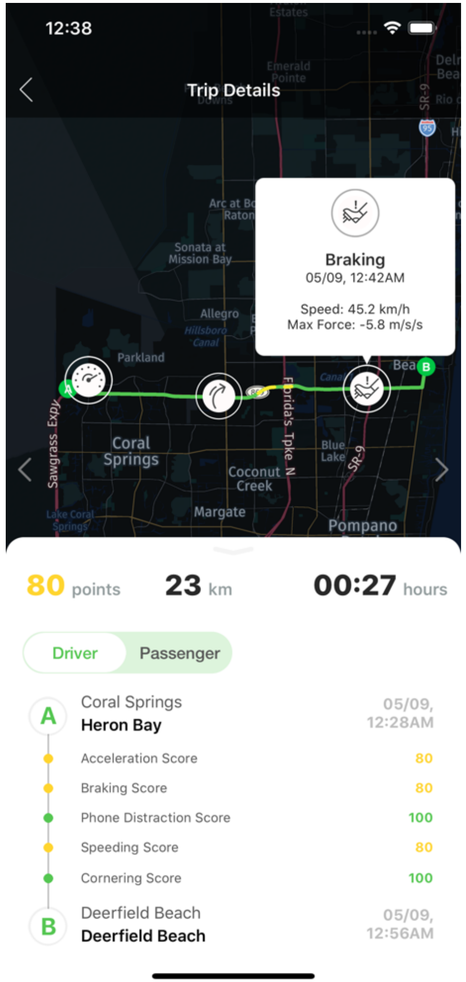
Примеры продуктов

- страхование авто (скидка за хорошего водителя)
- приложения каршеринга
- такси (водитель)
- флит-менеджмент (управление парком автомобилей)

Примеры продуктов







Популярные фреймворки

AFNetworking / Alamofire

Realm

Crashlytics

Google Maps SDK

Mapbox SDK

Особенности продуктового подхода

Особенности продуктового подхода

- конечные пользователи
- релизный цикл
- доставка

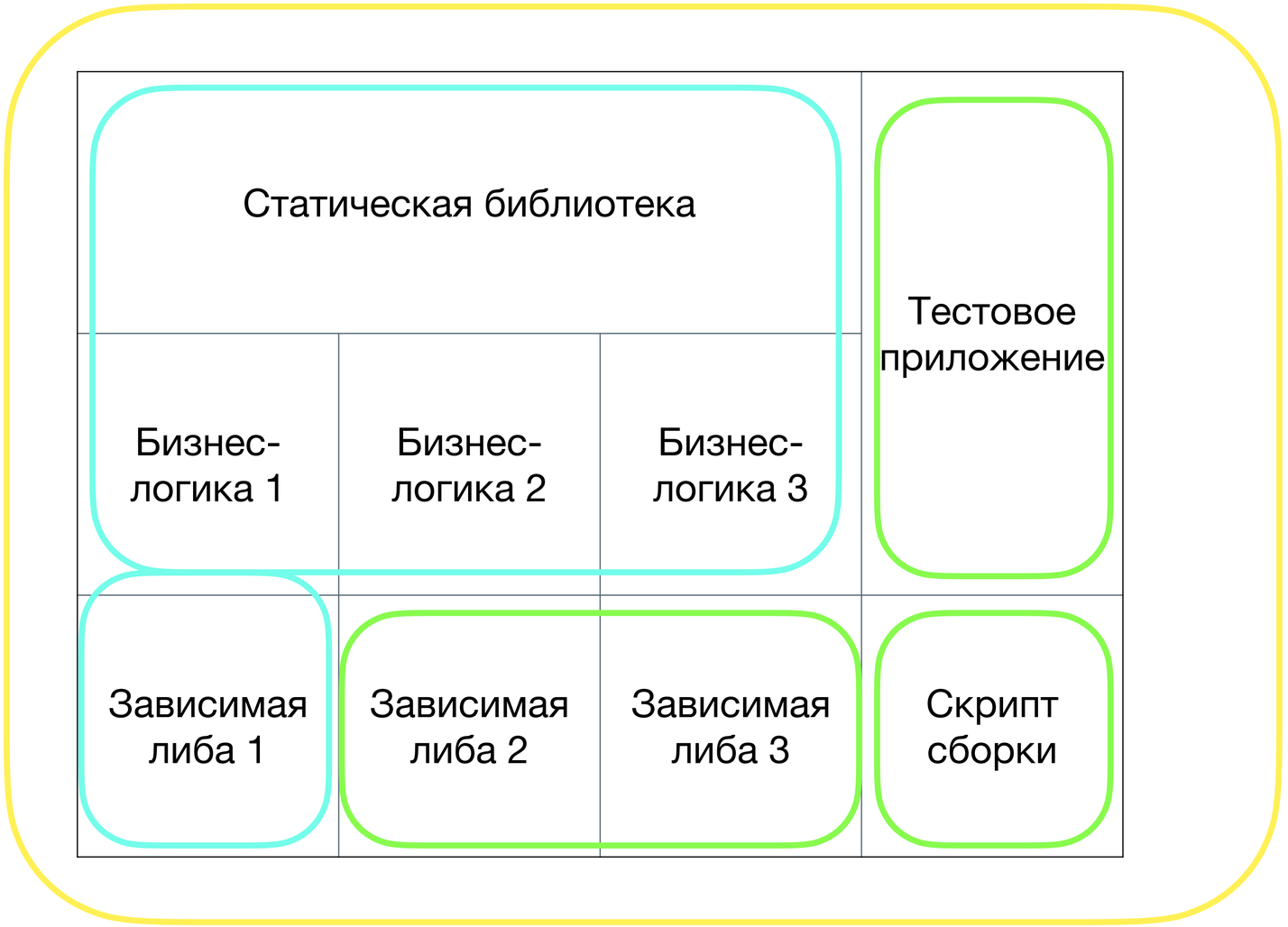
Немного теории

Виды фреймворков и библиотек

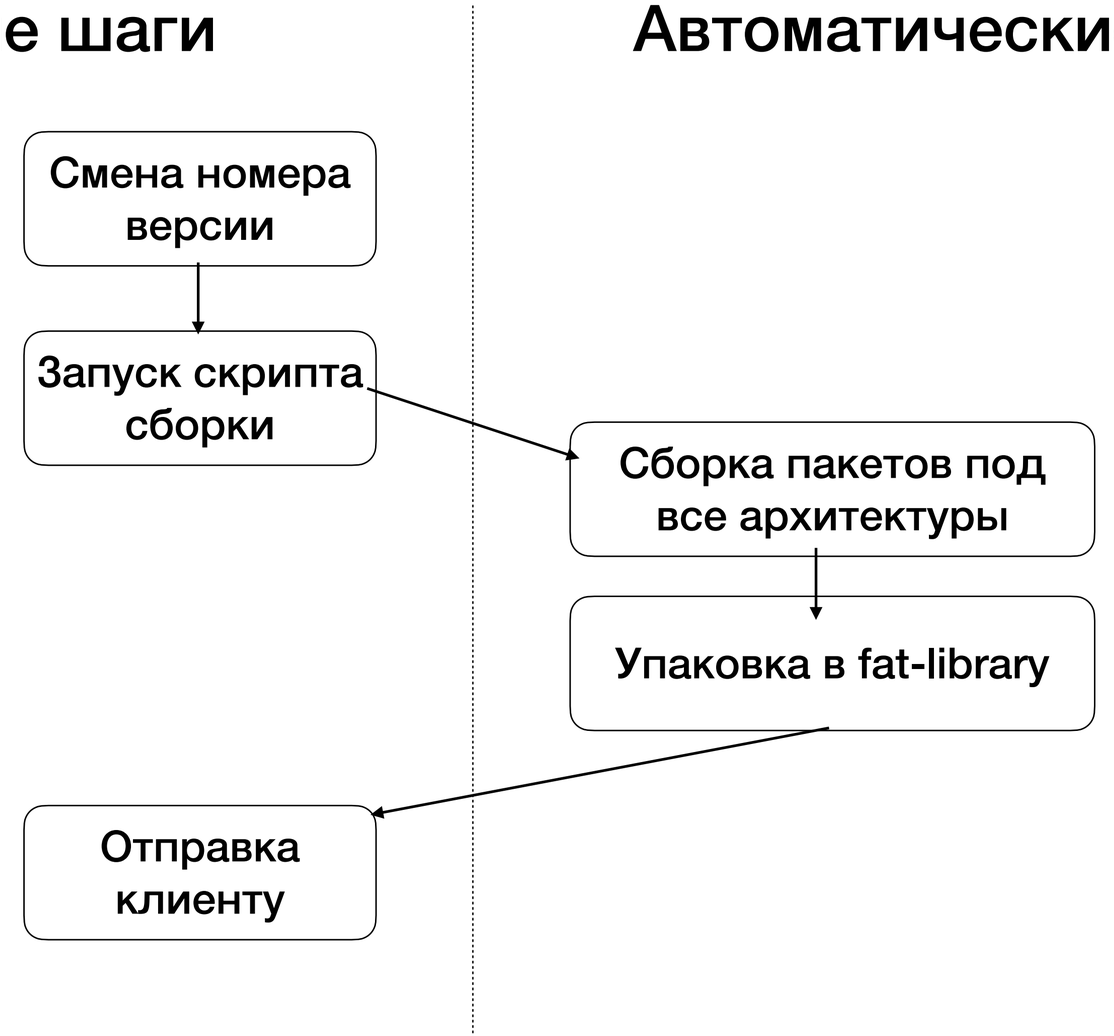
- framework vs library
- static vs dynamic
- fat library

Что было на входе

Примерная структура библиотеки



Примерная схема распространения



Проблемы

Проблемы

- сложность сборки
- сложность распространения и начала использования
- версионность
- конфликты зависимых библиотек

Решение проблем

Сложность распространения

Менеджер зависимостей

Наиболее очевидное решение сложности распространения

- как правило, уже используется в приложении
- позволяет подключить библиотеку, добавив всего 1 строчку в конфиг
- исключает проблемы ручных шагов
- решает проблему версионности
- позволяет использовать зависимости в библиотеках
- при необходимости, отключить библиотеку также легко как и установить

Выбор менеджера зависимостей

- 3 наиболее популярных:
  - Cocoapods
  - Carthage
  - Swift Package Manager
- начать с популярного
- оставлять возможность ручного добавления в проект
- учесть возможность использования скомпилированной библиотеки

Хранение скомпилированной библиотеки

На примере Cocoapods

- Репозиторий Cocoapods хранит только метаданные (podspec)
- Исходники хранятся в отдельном репозитории
  - самый популярный вариант
- точно также скомпилированные библиотеки могут лежать в отдельном месте, с доступом по прямой ссылке
  - zip-архив с доступом по прямой ссылке

Пример podspec

Все данные вымышлены

```
Pod::Spec.new do |s|

  s.name             = "MegaFramework"
  s.version          = "1.0.5"
  s.summary          = "This is a super mega framework"
  s.description      = "Framework for telematics data"
  s.homepage         = "https://google.com"
  s.license          = { :type => "Proprietary", :text => "All rights reserved." }

  s.author                 = { "Igor Nabokov" => "slimski@gmail.com" }

  s.ios.deployment_target = "9.0"
  s.ios.vendored_frameworks = 'MegaFramework.framework'

  s.source           = { :http => "https://storage.net/mega-ios/MegaFramework.zip" }


end
```

Где хранить?

- Существующее хранилище
  - ссылка заранее определена
  - количество трафика для upload
- репозиторий (GitHub, etc)
- Облачные хранилища:
  - персональные (google drive, dropbox, etc)
  - enterprise (Amazon, Azure, Google, Yandex)
- VPS или выделенный сервер

Сложность сборки

Решение сложности сборки

выделяем шаги

- сборка пакета
- номер версии
- публикация (рассылка по email)
- прогон тестов
- подготовка к публикации
- автоматическая публикация

Решение сложности сборки

управляем шагами

- разный порядок выполнения
- сводный отчет с подробностями
- контекст (test / prod)
- независимость от окружения

Fastlane

Возможности fastlane

- создание сценариев
- готовые действия
  - изменение версий,
  - сборка пакета
  - публикация в Cocoapods
- можно легко дописать новое действие

Другие варианты

- Нативные утилиты от Apple (xcodebuild, altool, agvtool, codesign итп)
- Xcode Server
- другие CI (Jenkins, CircleCI, Bamboo, итп)
- BuddyBuild

Кастомизация Fastlane

- выгрузка скомпилированного фреймворка в Azure
  - подходящий Action от MS с небольшими доработками:
    - azure/storage -> azure/storage/blob
    - отдельный метод для создания контейнера в Azure
- сценарии публикации с разными конфигурациями

Основной сценарий публикации

```
desc "build and publish current version to cocoapods"
  lane :make do |options|
    build_number = set_build
    prefix = options[:prefix]
    if prefix.nil?
      version = get_version_number(xcodeproj: framework_name, target:pulse_name)
    else
      version = get_version_number(xcodeproj: framework_name, target:pulse_name) + "-" + prefix + build_number
    end

    arc_name = project_name + "-" + version + ".zip"

    tests
    archive
    upload(arc_name: arc_name)
    update_pod(version: version, external:prefix.nil?)
    remove_artifacts
    commit
    push_to_git_remote
  end
```

```
desc "Set build number (options: number)"
lane :set_build do |options|
  set_info_plist_value(key: "BuildTime", value: Time.new.strftime("%FT%T%z"))
  increment_build_number(build_number: options[:number])
end

desc "Remove build artifacts"
lane :remove_artifacts do
  sh("rm -rf ./../*.xcarchive")
  sh("rm -rf ./../" + pulse_name + ".framework")
  sh("rm -rf ./../" + project_name + "-*.zip")
end

lane :tests do
  run_tests(scheme: pulse_name)
end
```

Проблемы

- сложность сборки
  скрипты fastlane
- сложность распространения и начала использования
  менеджер зависимостей
- версионность
  менеджер зависимостей
- конфликты зависимых библиотек
  менеджер зависимостей?

Конфликт зависимостей

- исключить как можно больше сторонних зависимостей
- рассмотреть возможность затащить код к себе
- использовать возможности менеджера зависимостей, но без строгой привязки к конкретному менеджеру

Дальнейшие улучшения

Что можно сделать еще

- документация
- демо-приложение
- хороший API

Доклад по API Аси Свириденко (Яндекс)

Переходим к практике

Спасибо за внимание

Все ссылки на использованные материалы:
https://telegra.ph/Ssylki-k-dokladu-po-iOS-frejmvorkam-05-25

Контакты автора:

Telegram: @slimski

Twitter: @IgorNabokov